Computational design of curved thin shells: from glass façades to programmable matter

by

Ruslan Guseinov

September, 2020

A thesis submitted to the
Graduate School
of the
Institute of Science and Technology Austria
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Committee in charge:

Herbert Edelsbrunner, Chair
Bernd Bickel
Chris Wojtan
Helmut Pottmann
Denis Zorin

Institute of Science and Technology

The thesis of Ruslan Guseinov, titled *Computational design of curved thin shells: from glass façades to programmable matter*, is approved by:

Supervisor: Bernd Bickel, IST Austria, Klosterneuburg, Austria

Signature: _____

Committee Member: Chris Wojtan, IST Austria, Klosterneuburg, Austria

Signature: _____

Committee Member: Helmut Pottmann, KAUST, Thuwal, Saudi Arabia

Signature: _____

Committee Member: Denis Zorin, NYU, New York, USA

Signature: _____

Defense Chair: Herbert Edelsbrunner, IST Austria, Klosterneuburg, Austria

Signature: _____

Signed page is on file

IST Austria Thesis, ISSN: 2663-337X

ISBN: 978-3-99078-010-7

I hereby declare that this thesis is my own work and that it does not contain other people's work without this being so stated; this thesis does not contain my previous work without this being stated, and the bibliography contains all the literature that I used in writing the dissertation.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution.

I certify that any republication of materials presented in this thesis has been approved by the relevant publishers and co-authors.

Signature: _____

Ruslan Guseinov
September, 2020

Signed page is on file

Abstract

Fabrication of curved shells plays an important role in modern design, industry, and science. Among their remarkable properties are, for example, aesthetics of organic shapes, ability to evenly distribute loads, or efficient flow separation. They find applications across vast length scales ranging from sky-scraper architecture to microscopic devices. But, at the same time, the design of curved shells and their manufacturing process pose a variety of challenges. In this thesis, they are addressed from several perspectives. In particular, this thesis presents approaches based on the transformation of initially flat sheets into the target curved surfaces. This involves problems of interactive design of shells with nontrivial mechanical constraints, inverse design of complex structural materials, and data-driven modeling of delicate and time-dependent physical properties. At the same time, two newly-developed self-morphing mechanisms targeting flat-to-curved transformation are presented.

In architecture, doubly curved surfaces can be realized as cold bent glass panelizations. Originally flat glass panels are bent into frames and remain stressed. This is a cost-efficient fabrication approach compared to hot bending, when glass panels are shaped plastically. However such constructions are prone to breaking during bending, and it is highly nontrivial to navigate the design space, keeping the panels fabricable and aesthetically pleasing at the same time. We introduce an interactive design system for cold bent glass façades, while previously even offline optimization for such scenarios has not been sufficiently developed. Our method is based on a deep learning approach providing quick and high precision estimation of glass panel shape and stress while handling the shape multimodality.

Fabrication of smaller objects of scales below 1 m, can also greatly benefit from shaping originally flat sheets. In this respect, we designed new self-morphing shell mechanisms transforming from an initial flat state to a doubly curved state with high precision and detail. Our so-called CurveUps demonstrate the encodement of the geometric information into the shell. Furthermore, we explored the frontiers of programmable materials and showed how temporal information can additionally be encoded into a flat shell. This allows prescribing deformation sequences for doubly curved surfaces and, thus, facilitates self-collision avoidance enabling complex shapes and functionalities otherwise impossible. Both of these methods include inverse design tools keeping the user in the design loop.

Моему дедушке Михаилу Ильичу (1919–2010),
энтузиасту философии и науки и светлому уму,
который воспитывал во мне любознательность
с раннего детства.

To my grandfather Mikhail Ilyich (1919–2010),
a philosophy and science enthusiast and a bright mind,
who fostered my curiosity since my early childhood.

Acknowledgements

The research presented in this thesis has been accomplished thanks to many people who supported me professionally and morally during my doctorate program.

First of all, I would like to thank my supervisor Bernd Bickel for being very carefully involved into all aspects of my growth as a researcher, helping me to set and reach ambitious goals, and openly sharing his experience.

A big thank you to all the committee members for accepting this role and for providing valuable feedback. Special thanks to Chris Wojtan who generously shared his time and ideas on a regular basis during many meetings. I am also very thankful to Helmut Pottmann for our exciting collaboration on one of the methods presented in this thesis. And, of course, I thank all the other collaborators for the joy of our collective work during which I had a chance to learn not only science but also communication. It has always been a pleasure to spend extended amounts of time in close collaborations, in particular with Konstantinos Gavriil visiting IST Austria and during my visit to Caltech where I worked with Connor McMahan and Chiara Daraio.

I thank the Computer Graphics research team of IST Austria, especially Eder Miguel and Jesús Rodríguez Pérez, for providing a lot of great feedback, sharing their knowledge, and having lots of fun together.

During the work on this thesis, I received substantial support from IST Austria's scientific service units. A big thank you to Todor Asenov and other Miba Machine Shop team members for their help with fabrication of experimental prototypes. In addition, I would like to thank Scientific Computing team for the support with high performance computing.

Financial support was provided by the European Research Council (ERC) under grant agreement No 715767 - MATERIALIZABLE: Intelligent fabrication-oriented Computational Design and Modeling, which I gratefully acknowledge.

I would like to thank my friends and family who always carefully supported me, especially I am very grateful to my parents.

Finally, I thank my wife Lisa. Your faith in me is truly priceless to me.

About the Author

Ruslan Guseinov got a *Specialist* degree (equivalent to MSc) in Applied Informatics at the Moscow Institute of Physics and Technology in 2010 and after his graduation worked as a Systems Analyst in a private company. Later his interest shifted to R&D and he started a new job as a Researcher and Software Engineer working on computational geometry algorithms in application to diamond cutting optimization. In 2014 his desire to improve his researcher skills and culture, led him to join the PhD program of IST Austria. There Ruslan's interests further evolved and he devoted his research to creation and computational design of new metamaterials suitable for shape morphing and to computational design in architecture. He published his works in high-impact journals, namely *Nature Communications* and *Transactions on Graphics* and presented his research at several venues including SIGGRAPH. During his PhD program, Ruslan has given several talks to high school students making an overview of his research and addressing questions of a young researcher's lifestyle.

List of Collaborators and Publications

◦ R. Guseinov, E. Miguel, and B. Bickel. Curveups: Shaping objects from flat plates with tension-actuated curvature. *ACM Trans. Graph.*, 36(4):64, 2017, https://doi.org/10.1145/3072959.3073709

◦ R. Guseinov, C. McMahan, J. Pérez, C. Daraio, and B. Bickel. Programming temporal morphing of self-actuated shells. *Nat. Commun.*, 11(1):237, Jan 2020, https://doi.org/10.1038/s41467-019-14015-2

Raw data is publicly available through IST Austria Research Explorer https://doi.org/10.15479/AT:ISTA:7154.

Source code and processed data are publicly available on https://github.com/russelmann/temporal-morphing-ncomms.

R.G. designed target shape geometries, fabricated, tested, and scanned shells, developed the inverse design tool, and performed simulations. He contributed to the design of the research, mechanical modeling, conducting the experiments, performing data modeling and fitting for bracket specimens.

◦ K. Gavriil*, R. Guseinov*, J. Pérez, D. Pellis, P. Henderson, F. Rist, H. Pottmann, and B. Bickel. Computational design of cold bent glass façades. *ACM Trans. Graph.*, 39(6):208, 2020, https://doi.org/10.1145/3414685.3417843.

K.G. and R.G. contributed equally to this work. R.G. developed glass panel shape optimization and acquired the simulation data. He contributed to the formulation of the panel shape representation and to the development of the deep learning approach.

Table of Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Design of curved surfaces has had a great importance in human history for ages. Perhaps, they were originally looked up by the artists, inventors, and craftsmen in nature. Ever since then, the aesthetics of organically bending, twisting, and stretching shapes have been bringing joy to the spectators. But it is more recent that we have a fundamental understanding of the reasons for such shapes to prevail long before they could have been appreciated by a critic.

The theories of solid mechanics, aero- and hydrodynamics, along with other domains of Physics and Mathematics explain remarkable properties of smoothly curved shapes. To name a few, they possess great toughness [MSA+18], low aerodynamic drag [UB18], or smoothly refract the light [STTP14]. Many of the fundamental discoveries related to curved surfaces were made in the age of industrialization which is characterised by the emergence of mass production. At that point, merely the first steps in that direction were taken and there used to be very little flexibility in manufacturing. Ironically, this lead to an immense deviation of the produced shapes from the ones found in nature and early manual crafting.

Modern progressive society has definitely overcome the stagnation of fabrication versatility of that period. We witness rapid development of many creative technologies of 3D printing [GDR+16], molding [AMG+19], nanofabrication [AHB18], etc. which are already being actively used in prototyping and production of goods. New fabrication techniques show how familiar tools can be used for novel construction approaches, such as the cold bent glass technique considered in this thesis. Recent advanced technologies enable creation of programmable *smart matter*. In particular, there are examples of methods for self-directed deformations of initially flat geometries including biomimetic 4D printing [GMN+16], self-folding light-responsive polymers [LSDG17], and more recent pneumatic elastomers [SRBR19] along with many others.

Overall, we consider in this thesis a kind of fabrication approaches when an initially flat sheet is transformed into a curved target shape. Despite they are often nontrivial in execution, they have a number of notable advantages over the alternatives, such as 3D printing or molding. The *flat-to-curved* approaches provide ways for energy saving and low-cost fabrication. The objects having a flat state as an intermediate phase in their fabrication can be very efficiently stored and transported before they take the final curved form. And, more importantly, flat-to-curved fabrication opens doors to new combined

ways of manufacturing. While a self-morphing shell is flat, it is possible to apply some of the advanced 2D fabrication technologies on it and bring them to the final curved surface due to the shell actuation. This can be done, for example, for printing electronic circuits [FRY+17] or sensors [GQM+17] which are extremely challenging to be executed directly on a curved surface.

This thesis is motivated by the gap coming up between the fabrication hardware advances and development of software realizing their potential. We present three methods dealing with fabrication of curved surfaces from flat sheets using modern fabrication techniques. The first one is devoted to computational design of cold bent glass façades and exhibits a machine learning approach to quickly predict and process physically realizable shapes for interactive shape modeling. Two other methods present novel programmable smart materials encoding shape geometry and temporal morphing. We work with materials with subtle mechanical properties which form inhomogeneous structures with large numbers of degrees of freedom and may have complex time-dependent behavior. Additionally, it is often not obvious from the computational point of view which level of abstraction to choose and how to decompose the final goal into intelligible parts. Overcoming these difficulties associated with highly nontrivial design spaces, we developed methods for modeling, simulation, and inverse design of various shell-like structures which cannot be tackled by the classical engineering approaches and software packages. Alongside with the computational contributions, we designed new structural mechanisms enabling our fabrication goals.

From the geometrical point of view, our methods are targeted at doubly curved, or non-developable, shapes which represent the most general case of smooth surfaces. According to Gauss' *Theorema Egregium* [Gau28], they cannot be flattened isometrically and require in-plane stretching or compression leading to the associated fabrication challenges. Such surfaces are also often referred to as "organic" or "freeform" shapes. For all the presented methods, special cases of the developables have been previously studied but they significantly restrict the design space as well as the associated functionality and appearance.

All the presented methods decompose a global surface into tiles, be it a single glass panel bent by the external loads or a 3D printed element interacting with its neighbors. This a common approach to enable construction of large objects and, at the same time, partially split the design problem into simpler components similar to the way how finite element method (FEM) breaks down the simulated media. This approach has been proven to work well independent of the scales and functionalities.

The following sections present our three methods: computational design of cold bent glass façades, encoding geometric information in doubly curved self-morphing shells, and encoding temporal information in doubly curved self-morphing shells.

1.1 Externally deformed shells

Curved glass façades allow the realization of aesthetically stunning looks for architectural masterpieces, as shown in Fig. 1.1. The curved glass is usually made with hot bending, a process where the glass is heated and then formed into a shape using a mold or using tailored bending machines for spherical or cylindrical shapes. While being able to unleash these stunning designs from being restricted to flat panels, this process is

laborious and expensive and, thus, an economic obstacle for the realization of exciting concepts such as the NHHQ skyscraper project by Zaha Hadid Architects (Fig. 4.7). As a cost-effective alternative, in recent years, architects have started exploring cold bending [Bee15]. Here, planar glass sheets are deformed by mechanically attaching them to a curved frame. Cold bending introduces a controlled amount of strain and associated stress in the flat glass at ambient temperatures to create doubly curved shapes [Dat17]. Compared with hot bent glass, it has the advantage of higher optical and geometric quality, a wide range of possibilities regarding printing and layering, the usage of partly tempered or toughened safety glass, and the possibility of accurately estimating the stresses from deformation [BIVIC07, FK11]. Furthermore, it reduces energy consumption and deployment time because no mold, heating of the glass, nor elaborate transportation are required.

However, designing cold bent glass façades comes with a challenging form-finding process. How can we identify a visually pleasing surface that meets aesthetic requirements such as smoothness between panels while ensuring that the solution is physically feasible and manufacturable? Significant force loads can occur at the connection between the glass and frame, and it is essential that the deformation of the glass stays within safe limits to prevent it from breaking.

In Chapter 4, we introduce our interactive, data-driven approach for designing cold bent glass façades. Starting with an initial quadrangulation of a surface, our system provides a supporting frame and interactive predictions of the shape and maximum stress of the glass panels. Following a designer-in-the-loop optimization approach, our system enables users to quickly explore and automatically optimize designs based on the desired trade-offs between smoothness, maximal stress, and closeness to a given input surface. Our workflow allows users to work on the 3D surface and the frame only, liberating the designer from the need to consider or manipulate the shape of flat panels – the optimal shape of the flat rest configuration of the glass panels is computed automatically.

At a technical level, we aim to determine the minimum energy states of glass panels conforming to the desired boundary without knowing their rest configuration. Based on extensive simulations of more than a million panel configurations with boundary curves

Figure 1.1: Examples of curved glass façades. Left: Fondation Louis Vuitton, Paris, by Frank Gehry (Photo[1]: Francisco Anzola / CC BY 2.0 / cropped). Right: Emporia, Malmö, by Wingårdh Arkitektkontor (Photo[2]: Susanne Nilsson / CC BY 2.0 / cropped).

[1] https://www.flickr.com/photos/fran001/49628199916/

[2] https://www.flickr.com/photos/infomastern/16240316513/

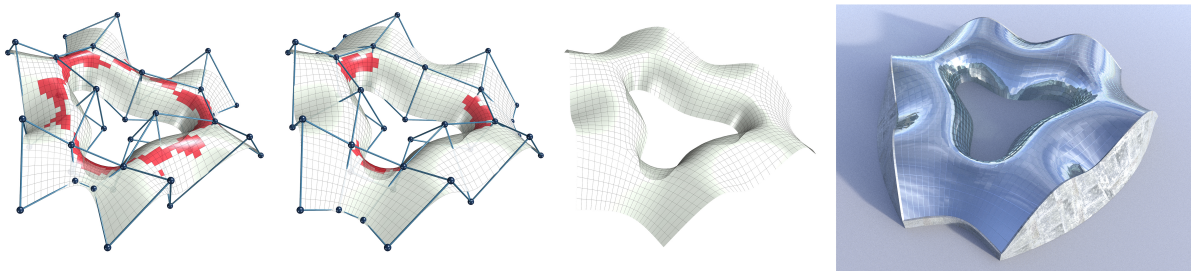


Figure 1.2: Material-aware form finding of a cold bent glass façade. From left to right: initial and revised panel layouts from an interactive design session with immediate feedback on glass shape and maximum stress (red color indicates panel failure). The surface design is then optimized for stress reduction and smoothness. The final façade realization using cold bent glass features doubly curved areas and smooth reflections.

relevant for our application domain, we observed the existence of several (in most cases up to two) stable states for many boundary curves. Identifying both *minimum energy states without knowing the rest configuration* and potentially *multiple stable states* is a non-trivial problem and cannot be easily computed using standard simulation packages. Furthermore, as a prerequisite for enabling *interactive design* for glass façades, we need to solve this problem for hundreds of panels within seconds.

Recently, deep neural networks have been successfully applied in similar workflows of interactive design and optimization [MKG+19]. Nevertheless, the existence of multiple stable states does not allow mathematically expressing panel geometry and maximal stress as functions of the given boundary frame and requires special treatment. To account for this, we develop a learning-based method utilizing a Mixture Density Network, a deep neural network architecture in junction with a Gaussian mixture model, that accurately predicts alternative shapes and maximum stresses of a glass panel given its boundary.

Training data for the network is acquired by running more than a million of physics-based shape optimization routines which is computationally at the scale of years of CPU time. The application of cloud computing let us accomplish data acquisition within days. Predictions of the trained network not observed originally are re-simulated and used for the database enrichment. Our model is differentiable, fast enough to interactively optimize and explore the shape of glass façades consisting of hundreds of tiles, and tailored to be easily integrated into the design workflow of architects. As a proof-of-concept, we have seamlessly integrated our system in a commonly used architectural design tool Rhino. We have carefully validated the accuracy and performance of our model by comparing it to real-world examples, and demonstrate its applicability by designing and optimizing multiple intricate cold bent glass façades.

1.2 Self-morphing shells

Self-morphing shell prototypes target applications in a wide range of scales from architecture to micro devices. Such objects may, for example, facilitate the building construction process [GWH+19] or enable implantable electronics [DQH19]. The potential of self-transforming structures with complex geometries has inspired researchers to explore novel materials, multi-material fabrication techniques, and material programmability as enabling technologies. The underlying transformation mechanism is usually the result of a complex

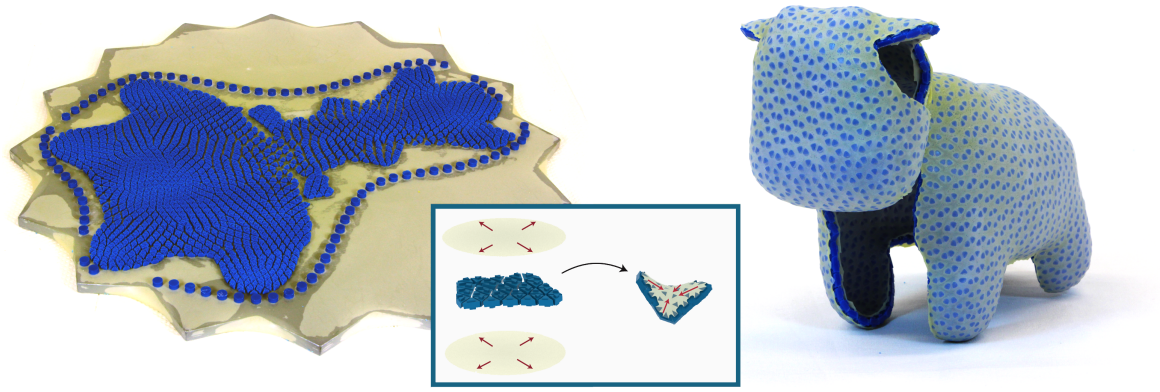


Figure 1.3: CurveUps are fabricated flat and actuated into doubly curved 3D shapes using pre-stretched elastic membranes.

interaction between materials exerting forces, anisotropic material behavior and distribution, and mechanical stability. Designing such structures is an active research challenge in computational fabrication, as it requires finding a physically valid configuration that satisfies functional constraints while facing a huge number of degrees of freedom.

We explore a type of transformation that starts from a flat initial configuration and evolves towards a desired three-dimensional shape. Self-morphing objects transforming specifically from a flat state can be stacked in volumetrically efficient arrangements, which simplifies transportation and storage. While flat sheets are easy to fabricate and store, many structural and functional applications across scales rely on changing surface curvatures (e.g., tunable mirrors [Cla08, LT03] and parabolic antennae [PTH+15]). Moreover, there is a wide range of advanced 2D fabrication technologies which can be used to directly inscribe electrical circuits, sensors, etc. into the shell in its original flat state. After the actuation, all the installed extra parts would be placed already on the curved surface, where direct installation would be a lot more complicated.

The first method in this section introduces how shape can be encoded into a self-morphing shell, and the second method extends this ability with the encodement of temporal information. Both shell types are laminates with contractile layers as the source of energy for the transformation and anisotropic tessellated layers encoding the information and guiding the morphing. Transformations between flat geometries and desired curved surfaces requires methods for prescribing local deformations. An overview of our approaches in this respect is presented below.

1.2.1 Encoding geometric information in material

In the first self-morphing method, we propose an actuation mechanism based on an anisotropic distribution of *disconnected* rigid tiles enclosed between two contractile layers. These outer layers drive the actuation process by pulling the tiles together. As the tiles collide with their neighbors, they join into a continuous shell forming the target geometry. This is a robust and cost-efficient transformation mechanism enabling the reproduction of a wide range of shapes at various scales provided that adequate materials are used.

Our particular realization of the mechanism uses uniformly pre-stretched latex sheets as the contractile layers which is a widely available standard material with excellent deformation properties. This is possible due to the fact that the initial configuration is

flat. Tiles are 3D printed in shapes resembling a frustum connected via pins to the elastic sheets (see Fig. 1.3).

By adjusting the distribution and shape of the tiles, the resulting local curvature can be influenced and thereby the resulting global shape defined. The tiles are rigidly attached to the pre-stretched sheets, which exert contracting forces on them once released. These forces can be adjusted by the layout of the tiles as well as by the parameters that control the attachment location. This construction enables shaping nearly smooth surfaces without developability restrictions. As a result, a complex self-transforming structure can be fabricated with simple rapid prototyping technologies.

We describe our computational approach for designing shapes from flat plates with tension-actuated curvature in Chapter 5. Our method starts with a desired input shape and automatically computes a fabricable configuration that best approximates the shape. At the core of our method is an optimization approach that solves a complicated layout and shape optimization problem: how do we best place the elements in a 2D layout, what is the optimal shape of these elements, and how do we connect them to the elastic sheet to obtain the required forces to reach a physically valid and stable state that closely approximates the desired input shape? Due to the nonlinear relationship between the element configurations and the resulting forces in the target configuration, this is a very challenging inverse problem, highly dependent on the initial guess. Motivated by simple construction rules observed in practice, we introduce a highly effective approximation to compute an initial guess. This initial guess often serves as a valid solution, and in case of any violation of the shell's stability condition, we then propose a physics-based model with a dedicated optimization method to locally fine-tune the configuration of the elements.

We verify our approach by both simulating the resulting structures and fabricating several example models, ranging from simple shapes such as elliptic or hyperbolic patches to design studies of popular models from architecture and computer graphics.

1.2.2 Encoding temporal information in material

Most proposed solutions in shape-morphing evolve towards a target geometry without considering time-dependent actuation paths. To achieve more complex geometries and avoid self-collisions, it is critical to encode a spatial and temporal shape evolution within the initially flat shell. Recent realizations of time-dependent morphing are limited to the actuation of few, discrete hinges and cannot form doubly curved surfaces. In Chapter 6, we demonstrate a method for encoding temporal shape evolution in architected shells that assume complex shapes and doubly curved geometries. The shells are non-periodic tessellations of pre-stressed contractile unit cells that soften in water at rates prescribed locally by mesostructure geometry. The ensuing mid-plane contraction is coupled to the formation of encoded curvatures.

Our realization of such mechanism can be interpreted at a high level as the CurveUps' tiles bridged by brackets, which are the core ingredient encoding temporal information. Originally they are stiff enough to keep the structure flat despite the stress exerted by the membrane. Once submerged into water, brackets become compliant over time at rates dependent on their thickness, which leads to their gradual compression under the membrane's load. We built a data-driven model of brackets by performing a series of experiments with samples subject to a variety of loading cases in water. We use this

model in an inverse design tool in order to encode desired actuation times into bracket thicknesses. Our inverse design tool accurately predicts the morphing process. This allows the user to make design decisions on shape geometry and actuation rates, for example, in order to avoid self-collisions. Fig. 1.4 shows our most complex shell which is very challenging to reach from the initial flat state as it is prone to multiple self-collisions during the morphing process.

In addition to the realization of encoded temporal morphing, our structures exhibit remarkable load bearing capability. Once the actuation is finished, they can be taken out from water and, as they dry, they regain higher stiffness. To demonstrate this property, we designed a mobile phone stand Fig. 1.5 holding an item twice its own weight. This property is specific to our self-morphing shells: they have a manually triggered (by submerging into water) transition phase during which they soften and perform the transformation, while before and after this phase they remain stiff.

1.3 Summary of contributions

Methods presented in this thesis contribute to the computer graphics and mechanical engineering communities in a number of ways. This section summarizes the most important of them while the details can be found further below in Chapters 4 to 6.

- We present an data-driven model for cold bent glass façades predicting and optimizing multimodal glass panel shape and maximal stress at interactive rates. This model can be seamlessly integrated into common architectural design software such as Rhino and enables interactive design of cold bent glass façades. The user observes nearly immediate optimal glass shape predictions together with the evaluation of feasibility of the panels. Additionally, our tool allows to automatically optimize a

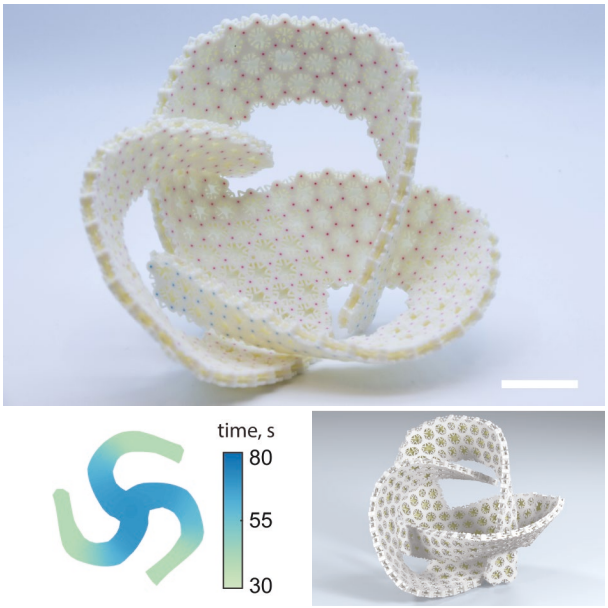


Figure 1.4: Spatio-temporally programmed shell (top), its actuation time landscape (bottom-left), and its corresponding simulated shell (bottom-right). This shell has a complex self-interweaving shape prone to multiple collisions in the course of its morphing process. Scale bar, 3 cm.

Figure 1.5: Our shells have remarkable load bearing capabilities. We designed a mobile phone stand holding an item twice its own weight.

given panelization eliminating not realizable panels while preserving design aesthetics expressed in term of fairness constraints.

- Two new self-actuated mechanisms for shells going from a flat to a doubly curved state are introduced. Both of them are laminates with contractile layers providing energy for the deformation process and anisotropically tessellated layers steering it. Both mechanisms allow the prescription of local deformations. The first one encodes target geometry with high precision. The second one additionally encodes temporal morphing which allows the realization of extremely complex shapes prone to self-collisions.

- To enable programming temporal morphing of shells, we developed an approach for data-driven modeling of time-dependent effective mechanical properties of structural elements (brackets) in isolation from the complete shell mechanism. This model can be applied both to predict morphing of the shell and to inversely design its transformation sequence.

- For each of the novel self-morphing mechanisms we developed algorithms to successfully encode global shape information. This includes target shape and morphing across the whole shell. These algorithms are applied in the dedicated custom-made inverse design tools, both keeping the user in the design loop. In the first tool, the user may introduce cuts to realize geometries otherwise infeasible. In the second tool, the user specifies the desired actuation rates across the shell, observes the predicted morphing and applies edits to achieve their goals, such as self-collision avoidance of the shell.

- Our latter self-morphing method enables fabrication of load bearing freeform self-morphing shells. They are originally flat and stiff. Immersion into water triggers the actuation process. After the actuation process, the shell can be taken out of water. As it cools down, it regains stiffness in the deformed target shape.

CHAPTER 2

Related work

Our work relates to several disciplines ranging from computational design of deformable objects to the foundations of self-actuating materials. In this chapter, we overview prior work by aggregating it into groups of similar core contributions.

2.1 Fabrication-aware surface design

Interactive design and shape optimization are areas that have a considerable history in engineering [CK08], architecture [ABVW14], and computer graphics research [BFR17, BCMP18], including tools for designing a wide variety of physical artifacts, such as furniture [UIM12], cloth [WS19], robotics [MTN+15], and structures for architecture [EKS+10].

Motivated by the digitalization of manufacturing, there is an increased need of computational tools that can predict and support optimizing the physical performance of an artifact during the design process. Several approaches have been developed to guarantee or improve the structural strength of structures [SVB+12, UMK17]. Focusing on shell-like structures, Musialski et al. [MAB+15] optimized their thickness such that it minimizes a provided objective function. More recently, Zhao et al. [ZXZ+17] proposed a stress-constrained thickness optimization for shells, and Gilureta et al. [GUPZ19] computed a rib-like structure for reinforcing shells, that is, adding material to the shell to increase its resilience to external loads. Considering both aesthetic and structural goals, Schumacher et al. [STG16] designed shells with an optimal distribution of artistic cutouts to produce a stable final result. Although we share the general goal of structural soundness, in our problem setting, we cannot change the thickness or material distribution. Additionally, even just determining the feasibility of a desired bent glass shape requires not only solving a forward simulation problem, but also an inverse problem because the rest shape of the glass panel is a priori unknown. Finding an optimal rest shape is often extremely important. Schumacher et al. [SZB18] investigated sandstone as building material that is weak in tension, thus requiring computing an undeformed configuration for which the overall stress is minimized. Similarly, glass panels have a low tensile strength and are subject to very high compression loads during the assembly process, which motivates the need for identifying minimal energy panels.

In our application to glass façades, having an accurate estimation of the stress is critical to predict panel failure and interactively guide designers towards feasible solutions. The

need to bridge the gap between accuracy and efficiency motivates the use of a data-driven approach.

Several methods have recently been proposed to design doubly curved objects from flat configurations [MPI+18, KLPCP18, PKLI+19] based on significantly elastic materials. All these methods are not targeted to be used within an interactive design pipeline, which relates them closer to our method for computational design of self-morphing shells. In a broader scope, paper folding is one of the most common techniques for generating surfaces. Previous works in this area have studied origami patterns [DO07, MGE07, DVTM16] and folding based on curved creases [KFC+08, MI11]. Recently, Tang et al. [TBWP16] added interactivity with their design tool for exploration of the space of developable surfaces. Our shape-morphing methods build on research in this area to compute unfolded designs of 3D models, but it explores the design space of non-developable surfaces.

To reach non-zero Gaussian curvatures from initially flat shells, bending must be coupled to in-plane stretches, according to Gauss' *Theorema Egregium* [Gau28]. Many other methods and tools have been developed that leverage a wide variety of material and structural properties to design fabricable doubly curved surfaces. Skouras et al. use stretching [STBG12] and bending [STK+14] of thin sheets to create inflatable structures with complex shapes. Similarly, Garg et al. [GSFD+14] exploit the shearing ability of regular wire grids to design aesthetic wire mesh sculptures. More recently, Konaković et al. [KCD+16] present a computational method for surface design that introduces a cutting pattern analogous to a triangular linkage to allow spatially varying stretching of originally inextensible flat sheets, similarly, the out-of-plane deformations of kirigami sheets are defined by the architecture of cut patterns [WGX+17, ZYN+15]. Kinematic frustration has recently been embraced for changing the curvature of initially flat shells [CMR+18]. These methods employ computational optimization strategies to find nontrivial configurations that approximate surfaces of double curvature. Ou et al. [OSV+16] introduce a universal bending mechanism that creates programmable, air-pressure-activated shape changes from a flat sheet with custom-shaped air pouches. Their design approach supports user-adjusted geometries of air pouches and a forward simulation of its transformation. Our shape-morphing approaches share similar goals but introduce the concept of self-actuation. We neither require an additional form to shape a plastic material nor any external stimuli such as air pressure. Furthermore, we explicitly solve an inverse problem to find a configuration that is physically stable.

2.2 Tile-based design

Mechanical structures used in all methods presented in this thesis can be seen as tilings of building blocks, be it glass panels or tiny 3D printed elements. The exploration of tile and element-based design spaces has been thoroughly studied. Examples include surface design using interlocking 3D [CPMS14, SCGT15] and planar [HBA12, SP13] elements, beadwork-based designs [IIM12], self-supporting masonry structures [VHWP12, DPW+14], or panel-based surfaces [PSB+08, EKS+10]. Often, fabrication of these structures requires finding an appropriate assembly sequence to produce the final surface. In contrast, our methods rely on the actuation mechanism to automatically perform the assembly, thus simplifying both the design and fabrication processes.

2.3 Computational design of façades

Covering general freeform surfaces with planar quadrilateral panels is a fundamental problem in architectural geometry and has received much attention [GSC+04, LPW+06, LXW+11, MDBL17, PEVW15]. The difficulties lie in the close relationship between the curvature behavior of the reference surface and the possible panel layouts. Problems occur especially in areas of negative curvature and if the design choices on the façade boundaries are not aligned with the curvature constraints imposed by planar quad meshes (Fig. 4.15). Using triangular panels, the problems are shifted toward the high geometric complexity of the nodes in the support structure [PEVW15]. Eigensatz et al. [EKS+10] formulated relevant aspects for architectural surface paneling into a minimization problem that also accounts for re-using molds, thereby reducing production costs. Restricting the design to simple curved panels, Pottmann et al. [PSB+08] presented an optimization framework for covering freeform surfaces by single-curved (developable) panels arranged along surface strips. However, glass does not easily bend into general developable shapes, limiting the applicability of this technique for paneling with glass.

A recent alternative for manufacturing doubly curved panels is cold bending of glass. A detailed classification and description of the performance of cold bent glass can be found in [Dat17]. Eversmann et al. [EIL16] explored simulations based on a particle-spring method and a commercially available FE analysis tool. Furthermore, they compared the resulting geometries to the measurements of the physical prototypes. For designing multi-panel façade layouts, Eversmann et al. [ESIL16] calculated the maximum Gaussian curvature for a few special types of doubly curved panels. This defined a minimal bending radius for exploring multi-panel façade layouts. Berk and Giles [BG17] developed a method for freeform surface approximation using quadrilateral cold bent glass panels. However, they limited their fabricability studies to two modes of deformation. Although conceptually simple, we found these approaches too limiting for general curved panels and, thus, have based our approach on a data-driven method.

2.4 Machine learning for data-driven design

Finite element methods (FEM) are widely used in science and engineering for computing accurate and realistic results. Unfortunately, they are often slow and, therefore, prohibitive for real-time applications, especially in the presence of complex material behavior or detailed models.

Dimensionality reduction is a powerful technique for improving simulation speed. Reduced space methods, for example, based on modal analysis [PW89, BJ05], are often used to construct linear subspaces, assuming that the deformed shape is a linear combination of precomputed modes. Simulations can then be performed in the spanned subspace, which, however, limits its accuracy, especially in the presence of non-linear behavior. Non-linear techniques such as numerical coarsening [CLSM15] allow for the reduction of the models with inhomogeneous materials, but usually require precomputing and adjusting the material parameters or shape functions [CBW+18] of the coarsened elements. Recently, Fulton et al. [FMD+19] proposed employing autoencoder neural networks for learning nonlinear reduced spaces representing deformation dynamics. Using a full but linear simulation, NNWarp [LSW+18] attempts to learn a mapping from a linear elasticity simulation to its nonlinear counterpart. Common to these methods is that they usually

precompute a reduced space or mapping for a *specific* rest shape but are able to perform simulations for a wide range of Neumann and Dirichlet boundary constraints. In our case, however, we are facing a significantly different scenario. First, we need to predict and optimize the behavior of a whole range of rest shapes, which are defined by manufacturing feasibility criteria (in our case, close to, but not necessarily perfect, rectangular flat panels). Second, our boundary conditions are fully specified by a low-dimensional boundary curve that corresponds to the attachment frame of the glass panel. Instead, we propose directly infer the deformation and maximal stress from the boundary curve.

Recently, data-driven methods have shown great potential for interactive design space exploration and optimization, for example, for garment design [WSFM19], or for optimized tactile rendering based on a data-driven skin mechanics model [VCO20]. An overview of graphics-related applications of deep learning can be found in Mitra et al. [MKG+19]. In the context of computational fabrication, data-driven approaches were used, for example, for interactively interpolating the shape and performance of parameterized CAD models [SXZ+17] or learning the flow for interactive aerodynamic design [UB18]. Although these methods are based on an explicit interpolation scheme of close neighbors in the database ([SXZ+17]) or Gaussian processes regression ([UB18]), in our work, we demonstrate and evaluate the potential of predicting the behavior and solving the inverse problem of designing a cold bent glass façade using neural networks. This entails the additional challenge of dealing with multistable equilibrium configurations that, to the best of our knowledge, has not been addressed before in a data-driven computational design problem.

2.5 Self-actuating materials

Previous works have developed different self-actuating materials and applied them to generate 3D shapes. Based on unidimensional components with discrete elements encoding the prescribed behavior, Raviv et al. [RZM+14] produce grids that deform from an initially planar configuration to a curved surface using hydrophilic materials for actuation. Gladman et al. [GMN+16] generalize this approach, encoding information in a continuous domain. They generate biologically inspired deforming structures using unidimensional composite hydrogel elements that swell anisotropically when immersed in water and interact in complex ways when arranged in networks. In [WMW+15], self-actuation has been demonstrated through nematic-to-isotropic phase changes in liquid crystal elastomers. Similarly, Kim et al. [KHB+12] use photopatterning polymer films to create temperature-responsive gel sheets that can transform from a flat configuration to a prescribed 3D shape. A variety of other 4D-printed systems can also be used to achieve desired shapes by coupling locally prescribed in-plane kinematics to changes in curvature [KTB+18, SAK+19, ABB+17, BKR+17, BvRL+19]. Moving into 2D domains, methods based on self-folding origami [HAB+10b, TFM+14, AMT+14, KWD+15, NEB+14, PKWB18] are able to exploit the information encoded in the crease patterns to create self-folding sheets. While the design space of these approaches is usually constrained to developable surfaces, hinge-based designs, or near constant Gaussian curvature designs, our approach uses an initially flat layer of disconnected rigid tiles, which allows us to avoid hinges and go beyond developable surfaces.

A key element in self-actuating materials is the actuation mechanism. Its goal is to provide the energy necessary to drive the prescribed change in properties. Previous works have

explored a wide variety of actuation mechanisms, from temperature-driven Shape Memory Alloys (SMA) [HAB+10b, FTS+13] and gel sheets [KHB+12] to hydrophilic materials that change properties when immersed in water [RZM+14, GMN+16], including pneumatic actuation [SIC+11] or light-responsive materials [RDC+12].

2.6 Temporal morphing of shells

The proposed solutions referenced in the previous sections demonstrate successful encodement of the geometric information in material, nevertheless they do not provide any means to control deformation rates during morphing processes. Consequently, self-collisions may occur in attempts to realize more complex geometries.

The ability to locally control the shape evolution in time drastically expands the design space of self-morphing shells. More broadly, an intrinsic capacity for pre-programmed temporal responses allows designing materials that can perform complex tasks, like self-deployment and locomotion, without the need for external controllers or power supplies [KMD+19]. While time-dependent folding has been demonstrated in structures that are wired to power sources and electronic control devices [HAB+10a, FTD+14], a small number of architected shells made of materials with intrinsic actuation capabilities have incorporated temporal programming through the sequential folding of discrete hinges [MYI+15, TGA+18, LSDG17]. However, none of these examples allow for changes in Gaussian curvature. They realize sequential folding of few discrete hinges [LSDG17], must be fabricated in their target shape prior to manual programming [MYI+15], or rely on ad-hoc empirical designs that do not account for characterizations of their materials' time-dependent constitutive responses [TGA+18].

CHAPTER 3

Thin sheet modeling

In all of the methods presented in this thesis, we model thin sheets using purely elastic physical models. Such models account only for reversible deformations which can be completely undone by removing all of the external loads. We found elastic models suitable both for modeling glass panels (Chapter 4) and thin rubber sheets (Chapters 5 and 6) and as these materials barely exhibit any irreversible deformations (such as plastic or viscous) in the scenarios occurring in the following chapters.

Obviously, thin glass and rubber sheets have very distinctive deformation behaviors. Glass is a solid and brittle material, which has significant resistance to bending even in a form of very thin sheets, whereas rubber membrane's deformation is often governed almost completely by the stretching and shearing forces. This fact lets us neglect the bending component of deformation in the second case.

In this section, we first introduce a continuous strain model for thin shells and different approaches for its discretization. Then, we present the physical models built on top of these discretizations used further in this thesis. Further details on the simulation, coupling, and shape optimization based on these models can be found in the following chapters.

3.1 Thin sheet strain model

A thin sheet of thickness h is geometrically represented by a mid-surface with extrusions in two opposite normal directions of magnitude $h/2$, where h is assumed to be much smaller than the minimal radius of curvature of the mid-surface. We consider the sheets in their deformed and undeformed states with the latter one always planar. We assume that there is no deformation associated with the normal lines to the mid-surface, i.e. they remain normal to the mid-surface and straight.

We denote the undeformed state quantities with capital letters and the deformed state quantities with lower-case letters, for example, rest state coordinates $\mathbf{X} \in \mathbb{R}^2$ and deformed coordinates $\mathbf{x} \in \mathbb{R}^3$. In this chapter, we denote mid-plane and bending quantities with a bar and a hat, correspondingly, for example, mid-plane strain $\bar{\mathbf{E}}$ and bending strain $\hat{\mathbf{E}}$. However, we change this notation in the following chapters since we do not refer to mid-plane and bending components explicitly.

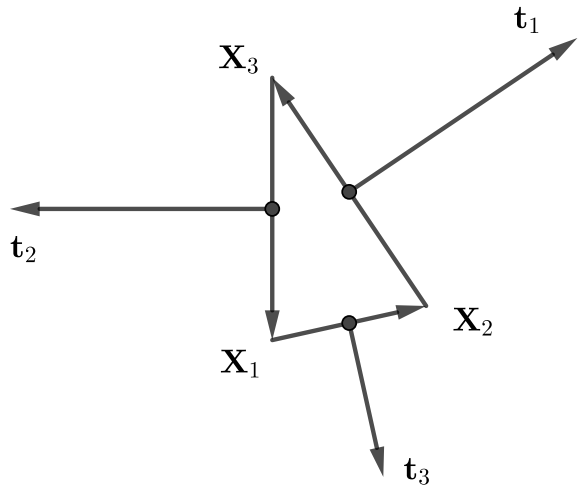


Figure 3.1: Vectors $\mathbf{t}_i$ used in strain computations are edge vectors rotated $\pi/2$ clockwise.

3.1.1 Continuous strain

Denote the mapping from undeformed to the deformed state $\chi : \mathbf{X} \to \mathbf{x}$, and the displacement field $\mathbf{u}(\mathbf{X})$. Then the mid-surface deformation gradient $\bar{\mathbf{F}} = \nabla_u \mathbf{x}$ and the Green-St-Venant strain tensor is the following:

$$\bar{\mathbf{E}} = \frac{1}{2}\big(\bar{\mathbf{F}}^T\bar{\mathbf{F}} - \mathbf{I}\big).$$

Note that this strain tensor can be approximated by $\bar{\mathbf{E}} \approx (\bar{\mathbf{F}}^T + \bar{\mathbf{F}})/2 - \mathbf{I}$ measuring the difference in squared distances between the deformed and undeformed states:

$$\mathbf{U}^T\bar{\mathbf{E}}(\mathbf{X})\mathbf{U} \approx \big\|\chi(\mathbf{X}+\mathbf{U}) - \chi(\mathbf{X})\big\|^2 - \|\mathbf{U}\|^2.$$

Under linearity approximation, following derivations from [GSH+04], the volumetrically defined deformation gradient can be written out as follows:

$$\mathbf{F}(\mathbf{X}, z) = \big(\mathbf{I} + z\hat{\mathbf{E}}(\mathbf{X})\big)\bar{\mathbf{F}}(\mathbf{X}), \tag{3.1}$$

where z represents the offset relative to the mid-plane, and Green's strain tensor

$$\mathbf{E}(\mathbf{X}, z) = \bar{\mathbf{E}}(\mathbf{X}) + z\hat{\mathbf{E}}(\mathbf{X}), \tag{3.2}$$

where $\hat{\mathbf{E}}$ is bending strain. Since in our case the undeformed state is planar, $\hat{\mathbf{E}}$ is equal to the shape operator of the deformed mid-surface.

3.1.2 Membrane strain discretization

We assume piecewise constant strains over FEM triangles. We separately discretize in-plane and bending strains from Equation (3.2).

Denote l_i length of the i^{th} edge of a triangle at rest and $\tilde{l}_i$ its length in the deformed state. Then, membrane strain is expressed as follows:

$$\bar{\mathbf{E}} = \frac{1}{16A^2}\sum_i \epsilon_i(\mathbf{t}_j \otimes \mathbf{t}_k + \mathbf{t}_k \otimes \mathbf{t}_j), \tag{3.3}$$

where A is area of the triangle, $\epsilon_i = \tilde{l}_i^2 - l_i^2$ (i^{th} edge strain), and $\mathbf{t}_i$ are undeformed edge's in-plane outward normals of lengths l_i (in other words, edge vectors rotated $-\pi/2$). See Fig. 3.1 for notation.

3.1.3 Bending strain discretization

A common way to discretize the shape operator in computer graphics is the edge-based approach. One of the principal curvatures is assumed to be equal to zero and oriented along the edge and another one is normal to the edge. This discretization is not able to follow smooth mid-surface's principal curvature lines under mesh refinement and remains highly dependent on the mesh triangulation. Another, more precise, approach is the following triangle-based discretization:

$$\hat{\mathbf{E}} = \sum_i \frac{\psi(\theta_i)}{2Al_i} \mathbf{t}_i \otimes \mathbf{t}_i,$$

where θ_i is the signed dihedral angle associated with the edge i, and $\psi(\theta)$ is some monotonic function in θ. It has to have derivative equal to 1 near zero and greater than zero for any $-\pi < \theta < \pi$. The following examples are commonly used: $\psi(\theta) = \theta$, $\psi(\theta) = 2\sin(\theta/2)$, or $\psi(\theta) = 2\tan(\theta/2)$.

This formulation is the simplest known way to capture bending directions in a continuous sense, nevertheless, it has been proven to be noisy for irregular triangulations [GGRZ06]. The flaw of this model lies in insufficient amount of DoFs of the linear triangular discretization. It has been corrected by introducing extra DoFs per edge defining deviation of mid-edge normals from the adjacent triangle-averaged direction [GGRZ06]:

$$\hat{\mathbf{E}} = \sum_i \frac{\theta_i/2 + s_i\phi_i}{Al_i} \mathbf{t}_i \otimes \mathbf{t}_i,$$

where ϕ_i is the deviation from the normal and s_i is the sign of this deviation depending on which of the two adjacent triangles this shape operator refers to.

3.2 Elastic shell model

We split our elasticity model into two parts: membrane energy and bending energy. Both components are integrated over the panel thickness to obtain energy densities in the domain of the triangles. The Saint Venant–Kirchhoff membrane energy is defined via Lamé parameters $\lambda = \frac{Y\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{Y}{2(1+\nu)}$ (for Young's modulus Y and Poisson's ratio ν) as follows:

$$\bar{W} = h\Big(\frac{\lambda}{2}\big(\operatorname{Tr}\bar{\mathbf{E}}\big)^2 + \mu \operatorname{Tr}\big(\bar{\mathbf{E}}^2\big)\Big). \tag{3.4}$$

For the bending energy density sometimes the following simple expression is used: $\hat{W}_{\text{DS}} = C(\operatorname{Tr}\hat{\mathbf{E}})^2$. This expression captures only the change in mean curvature and, thus, it not suitable for us. As an alternative, we use the following expression from Koiter's shell model [Koi66]:

$$\hat{W}_{\text{K}} = \frac{\mu h^3}{12}\Big(\frac{\lambda}{\lambda+2\mu}\big(\operatorname{Tr}\hat{\mathbf{E}}\big)^2 + \operatorname{Tr}\big(\hat{\mathbf{E}}^2\big)\Big). \tag{3.5}$$

The total elastic energy density across the triangle with rest area A_0 is the following:

$$W_E = A_0\big(\bar{W} + \hat{W}_{\mathrm{K}}\big). \tag{3.6}$$

3.3 Elastic membrane model

The thin membrane model neglects the bending component of the deformation. This leads to a compact strain expression enabling relatively simple expression for nonlinear material models. We use the incompressible Neo-Hookean material model since it captures the behavior of rubber precisely enough, even for large strains. In this model, the normal lines to the mid-surface remain straight but may experience stretching or compression. The Right Cauchy-Green strain tensor for a shell can be written as

$$\bar{\mathbf{C}} = 2\bar{\mathbf{E}} + \mathbf{I} = \bar{\mathbf{F}}^T\bar{\mathbf{F}} = \lambda_1\mathbf{N}_1\mathbf{N}_1^T + \lambda_2\mathbf{N}_2\mathbf{N}_2^T, \tag{3.7}$$

where λ_1 and λ_2 are the squared principal stretches while $\mathbf{N}_1$ and $\mathbf{N}_2$ are their corresponding eigenvectors.

Due to the volumetric incompressibility and the absence of transverse shearing, we can write down the Cauchy Green tensor as follows:

$$\mathbf{C} = \begin{bmatrix} \bar{\mathbf{C}} & 0 \\ 0 & (\lambda_1\lambda_2)^{-1} \end{bmatrix} \tag{3.8}$$

and the strain energy density as follows:

$$W_{\mathrm{M}} = \kappa(\operatorname{Tr}\mathbf{C} - 3) = \kappa(\lambda_1 + \lambda_2 + \frac{1}{\lambda_1\lambda_2} - 3), \tag{3.9}$$

where κ is the stiffness coefficient.

Using Constant Strain Triangles, we calculate the elastic forces as:

$$\mathbf{f}_i^{\mathrm{e}} = -\sum_{e\in\mathcal{F}_i}\frac{\partial W_{\mathrm{M}}^e}{\partial\mathbf{x}_i}V_e = -\sum_{e\in\mathcal{F}_i}\left(\frac{\partial W_{\mathrm{M}}^e}{\partial\lambda_1^e}\frac{\partial\lambda_1^e}{\partial\mathbf{x}_i} + \frac{\partial W_{\mathrm{M}}^e}{\partial\lambda_2^e}\frac{\partial\lambda_2^e}{\partial\mathbf{x}_i}\right)hA_e,$$

where $\mathcal{F}_i$ is the set of faces incident to vertex i, h is the thickness of the membrane at rest, A_e is the area of element e in the rest configuration, $V_e = hA_e$ is the volume of element e in the undeformed configuration and $\mathbf{x}_i$ are the membrane vertices.

CHAPTER 4

Computational design of cold bent glass façades

This chapter presents a method for interactive design of cold bent glass façades, a cost-efficient technique for building doubly curved architectural panelizations. In such constructions, glass panels are bent in-place and installed into curved frames remaining stressed, which makes it challenging to model them in a way allowing for computational design at interactive rates. We describe our approach based on a Mixture Density Network trained on a large number of shape optimization procedures.

4.1 Overview

We propose a method for the interactive design of freeform surfaces composed of cold bent glass panels that can be seamlessly integrated in a typical architectural design pipeline. Fig. 4.1 shows an overview of the design process. The user makes edits on a base quad mesh that is automatically completed by our system to a mesh with curved Bézier boundaries. Our data-driven model then interactively provides the deformed shape of the cold bent glass panels in the form of Bézier patches conforming to the patch boundaries and the resulting maximal stress. This form-finding process helps the designer make the necessary decisions to avoid panel failure. At any point during the design session, the user can choose to run our simulation-based optimization method to automatically compute a suitable panelization while retaining some desirable features such as surface smoothness and closeness to the reference design.

In Sec. 4.2, we show how the base mesh controlled by the user is extended through special cubic Bézier curves to the set of patch boundaries. Each patch is delimited by planar boundary curves of minimum strain energy. These special Bézier patch boundaries are convenient for modeling glass panels because they facilitate the construction of supporting frames while providing a smooth approximation to the desired design.

Bézier boundaries do not convey any information on the deformed or undeformed configuration of the panel. Our method uses simulation to compute both configurations of the panel such that certain conditions are met, which are derived from manufacturing constraints. First, current panel assembly does not guarantee C^1 continuity at the boundary between neighbor panels because it is very hard to enforce normals along the frame

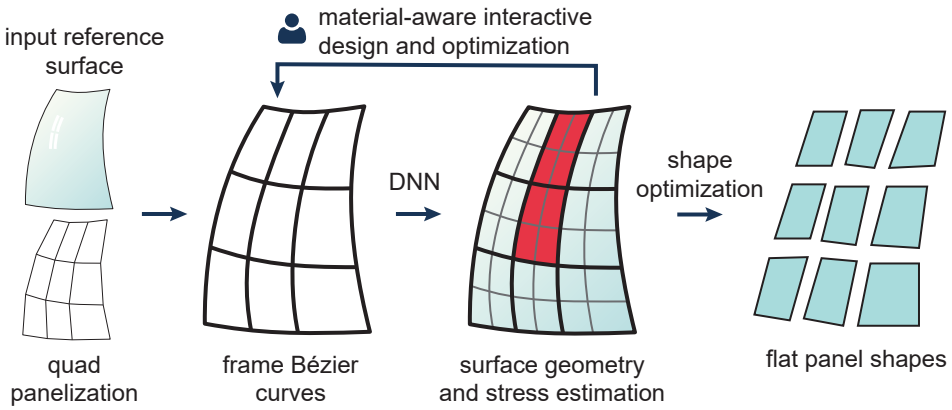


Figure 4.1: Overview of our design tool workflow. The user makes edits on a quadrilateral base mesh and gets immediate feedback on the deformed shape and maximal stress of the glass panels. When needed, an optimization procedure interactively refines the surface to minimize safety and fairness criteria. If desired, any target reference surface may be used to initialize the process.

in practice. Second, glass panels have a low tensile strength and are prone to breaking during the installation process in the presence of large tangential forces. Following these criteria, we let the panel be defined by the boundary curve of the frame and compute both the deformed and undeformed shapes of the panel such that the resulting total strain energy is minimal. In this way, we ensure our panelization has at least C^0 continuity and that the assembly of the panels requires minimal work, thus reducing the chances of breakage. In Sec. 4.3, we describe in detail the physical model and the computation of minimal energy panels.

Panel shape optimization provides us with a mapping between our design space of Bézier boundary curves and theoretically realizable cold bent panels, in both undeformed and deformed configurations. Our material model also accurately estimates the maximum stress endured by the glass. The user is free to interactively edit the base mesh while receiving immediate feedback on the maximum stress, but this neither ensures the panels will not break, nor does it foster the approximation of a target reference surface. To achieve this goal, we solve a design optimization problem: Bézier boundary curves are iteratively changed to minimize closeness to an input target surface (and other surface quality criteria) while keeping the maximum stress of each panel within a non-breaking range. In Sec. 4.5, we describe in detail our formulation of the design optimization.

However, accurately computing the minimal energy panels is computationally very challenging, which makes physical simulation infeasible for being directly used within the design optimization loop. Furthermore, the mechanical behavior of glass panels under compression often leads to multiple stable minimal energy configurations depending on the initial solution. This complicates the optimization even more: not only does the problem turn into a combinatorial one, but there is no algorithmic procedure that can efficiently count and generate all existing static equilibria given some boundary curve. We address this challenge by building a data-driven model of the physical simulation. First, we densely sample the space of the Bézier planar boundary curves and compute the corresponding minimal energy glass panels together with an estimation of the maximum stress. Then, we train a Mixture Density Network (MDN) to predict the resulting deformed shape and maximum stress given the boundary of the panel. The MDN explicitly models

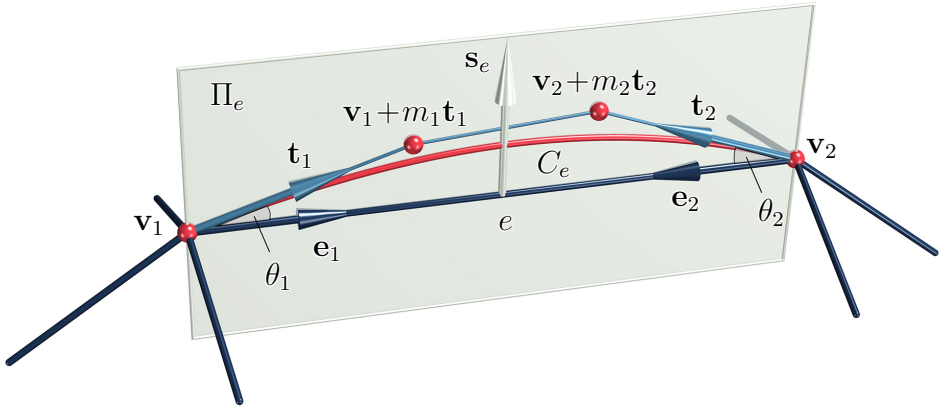


Figure 4.2: Parameterization of a panel boundary curve from a pair of tangent directions $\mathbf{t}_1$, $\mathbf{t}_2$ corresponding to dual halfedges. The final boundary curve (red) is computed by minimizing a linearized bending energy.

multistability and also allows us to discover alternative stable equilibria that can be used to enrich the training set. In Sec. 4.4, we elaborate on the characteristics of our regression network and our sampling and training method. The trained regression network can finally be used to solve the inverse design optimization problem. Once the user is satisfied with the design, our shape optimization procedure generates the rest planar panels, which are ready to be cut and assembled into a beautiful glass façade.

4.2 Geometry representation

A panelization of an architectural surface is built upon a quadrangular base mesh $\mathcal{M} = (V, E, F)$, where the vertices V determine the panel corner points and each quad face in F is filled by one curved panel. In practice, the user interacts with the design tool by making edits to $\mathcal{M}$ through any parametric mesh design method, in our case a Catmull-Clark subdivision from a coarser mesh (see Fig. 1.2). This helps achieve fair base meshes and gives a reasonable control for edits. However, any other mesh design scheme could be potentially used.

Each edge in E is then automatically replaced by a planar cubic Bézier curve defining the boundaries of the panel, and the inner control points are predicted using our regression model. In this section, we describe the details for getting from $\mathcal{M}$ to the union of curved panels. Moreover, we show how to express the panels with a minimal number of parameters, which are later used for the data-driven model.

4.2.1 Panel parameterization

We model each glass panel as a bicubic Bézier patch $\mathbf{S} : [0,1]^2 \to \mathbb{R}^3$, which is defined by 16 control points $\mathbf{c}_{ij}$, where $i,j \in \{0,1,2,3\}$. The corner points $\mathbf{c}_{00}$, $\mathbf{c}_{03}$, $\mathbf{c}_{33}$, $\mathbf{c}_{30}$ are vertices in $\mathcal{M}$.

Panel boundary

Each edge e of $\mathcal{M}$ is associated with a patch boundary curve $\mathbf{C}_e$. To describe its construction, we focus on a single edge e with vertices $\mathbf{v}_1, \mathbf{v}_2$, and we denote the unit

vectors of the half-edges originating at $\mathbf{v}_i$ by $\mathbf{e}_i$ (see Fig. 4.2). We opted for planar boundary curves representing panel's edges; thus, we first define the plane Π_e that contains $\mathbf{C}_e$. We do this by prescribing a unit vector $\mathbf{s}_e \in \mathbb{R}^3$ that lies in Π_e and is orthogonal to e. Note that this parameterization is non-injective (vector $-\mathbf{s}_e$ represents the same plane Π_e), but its ambiguity can be resolved using the compact representation in Sec. 4.2.2. The two inner control points of the cubic curve $\mathbf{C}_e$ lie on the tangents at its end points. Tangents are defined via the angles θ_i they form with the edge. Hence, the unit tangent vectors are

$$\mathbf{t}_i = \mathbf{e}_i \cos\theta_i + \mathbf{s}_e \sin\theta_i,$$

In view of our aim to get panels that arise from the flat ones through bending, we further limit the cubic boundary curves to those with a minimal (linearized) bending energy, as described in [YC04]. For them, the two inner control points are given by $\mathbf{v}_i + m_i\mathbf{t}_i$, $i = 1, 2$, with

$$m_1 = \frac{(\mathbf{v}_2 - \mathbf{v}_1) \cdot [2\mathbf{t}_1 - (\mathbf{t}_1 \cdot \mathbf{t}_2)\mathbf{t}_2]}{4 - \left(\mathbf{t}_1 \cdot \mathbf{t}_2\right)^2},$$

and m_2 is obtained analogously by switching indices 1 and 2.

The boundary of $\mathbf{S}$ is thus fully parameterized by the 4 corner vertices $\mathbf{c}_{ij}$, $i, j \in \{0, 3\}$, the 4 edge vectors $\mathbf{s}_e$, and the 8 tangent angles θ (2 per edge). This parameterization of the panels is used in the regression model and the design tool implementation described in Sec. 4.4 and Sec. 4.5 respectively.

Panel interior

The interior control points $\mathbf{c}_{ij}$, $i, j \in \{1, 2\}$ express the shape of a panel enclosed by a given boundary. We found that within the admissible ranges of the boundary parameters, any optimal glass panel (see a detailed description in Sec. 4.3) can be very closely approximated by fitting the internal nodes of the Bézier patch. Moreover, we need to regularize the fitting, because for a given Bézier patch, it is possible to slide the inner control points along its surface while the resulting geometry stays nearly unchanged.

We denote the vertices of the target panel shape $\mathbf{x}_i$ and the corresponding vertex normals $\mathbf{n}_i$. For every $\mathbf{x}_i$, we find the closest points $\mathbf{y}_i$ on the Bézier surface and fix their coordinates in the parameter domain. The fitting is then formulated as follows:

$$\min_{\mathbf{c}_{ij}} \quad \left\| \left(\mathbf{y}_i(\mathbf{c}_{ij}) - \mathbf{x}_i\right) \cdot \mathbf{n}_i \right\|^2 A_i + w_\mathrm{B} \sum_k E_k^2(\mathbf{c}_{ij}), \quad i, j \in \{1, 2\},$$

where A_i are Voronoi cell areas per panel vertex, E_k are the lengths of all control mesh edges incident to the internal nodes, and w_B is the regularizer weight that we set to 10^{-5}. To achieve independence of rigid transformations, we express the inner control points in an orthonormal coordinate frame adapted to the boundary. The frame has its origin at the barycenter of the four corner points. Using the two unit diagonal vectors

$$\mathbf{g}_0 = \frac{\mathbf{c}_{33} - \mathbf{c}_{00}}{\|\mathbf{c}_{33} - \mathbf{c}_{00}\|}, \quad \mathbf{g}_1 = \frac{\mathbf{c}_{30} - \mathbf{c}_{03}}{\|\mathbf{c}_{30} - \mathbf{c}_{03}\|},$$

the x-axis and y-axis are parallel to the diagonal bisectors, $\mathbf{g}_1 \pm \mathbf{g}_0$, and the z-axis is parallel to $\mathbf{b} = \mathbf{g}_0 \times \mathbf{g}_1$, which we call the face normal.

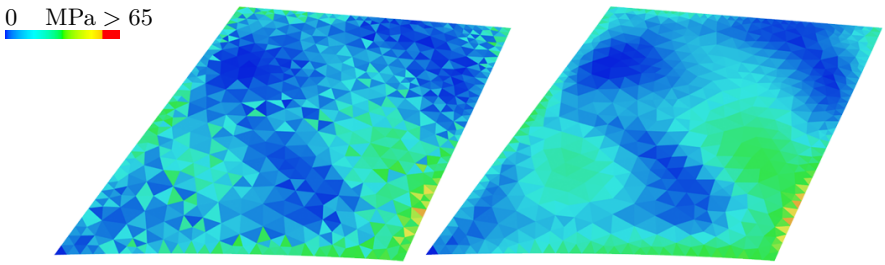


Figure 4.3: Comparison between the stress distribution produced with the typical shape operator used in, e.g., [PNdJO14] (left), and ours suggested in [GGRZ06] (right). The latter is much smoother and results in a more reliable estimation of the maximal stress.

4.2.2 Compact representation

The panel boundary is used as an input to a neural network to predict the shape and stress of the minimal energy glass panel(s) conformal to that boundary. Thus, it is beneficial to reduce the input to the essential parameters, eliminating rigid transformations of the boundary geometry.

We consider $\mathbf{d} \in \mathbb{R}^6$ to be the vector of the six pairwise squared distances of vertices $\mathbf{c}_{ij}$, $i, j \in \{0, 3\}$. Given $\mathbf{d}$, we can recover two valid mirror-symmetric embeddings of the 4 corner points. Assuming that the order of the vertices is always such that

$$\det(\mathbf{c}_{03} - \mathbf{c}_{00}, \mathbf{c}_{30} - \mathbf{c}_{00}, \mathbf{c}_{33} - \mathbf{c}_{00}) \geq 0$$

holds, the embedding is unique up to rigid transformations. We assume such a vertex ordering from now on. The plane Π_e for each edge is then characterized by its oriented angle γ_e with the face normal $\mathbf{b}$. Finally, we define $\mathbf{p} \in \mathbb{R}^{18}$ as the concatenation of the distance vector $\mathbf{d}$, the 4 edge plane inclinations γ_e, and the 8 tangent angles θ (2 per edge). The vector $\mathbf{p}$ is used as an input to the neural network defined in Sec. 4.4.

4.3 Panel shape optimization

Our method leverages mechanical simulation to create a large dataset of minimal energy panels that conform to cubic Bézier boundaries. Given some boundary curves, we are interested in finding deformed glass configurations that are as developable as possible. Non-developable panels result in high tangential forces that complicate the installation of the panel and increase the chances of breakage. By finding the pair of deformed and undeformed shapes of the panel that minimize the strain energy subject to a fixed frame, we ensure the work required for its installation is minimal, helping to reduce the tangential force exerted at the boundary. This dataset is used to train and test a model that predicts the deformed state and maximum stress of such panels, which is suitable for rapid failure detection and inverse design. In this section, we describe the simulation method used for the computation of the deformed and undeformed states of a minimal energy glass panel.

To sufficiently accurately predict glass shape and stress under small strains, we apply the Koiter's shell model with the discrete formulation based on the discrete shape operator suggested in [GGRZ06] which has been previously considered by Weischedel [Wei12]. Contrary to simpler thin shell bending models commonly used in computer graphics, e.g., [PNdJO14], this model more faithfully captures principal strain curves and outputs

smoother stress distributions (Fig. 4.3). Refer to Chapter 3 for the definitions of terms related to the shell model that we use. In this section, we describe how we find the minimal energy configuration corresponding to some given Bézier boundaries.

4.3.1 Minimal energy panels

Given a parametric design of a façade composed of a quadrangular mesh with Bézier curves at the edges, we aim to find a suitable panelization using cold bent glass. Although deforming a glass panel to conform to cubic boundaries is feasible, the fragility of this material imposes non-trivial constraints on the maximum amount of stress tolerated by the panels. Thus, we designed a method to compute the glass panel design with the lowest possible strain energy that still will fit our installation constraints. Note that in practice, the existing assembly methods do not preserve normals across neighboring panels; thus, we restrict our problem to guarantee only C^0 continuity. By computing the fabricable panel with the lowest possible total strain energy, we minimize the net work required to install the panel and notably reduce the local tangential stress suffered by the glass.

Overall, our pipeline takes as an input the 16 control points of the Bézier boundaries and automatically computes both the deformed $\mathbf{x}$ and undeformed $\mathbf{X}$ configurations of a planar glass panel that conforms to the boundary and has minimal energy. This is done in two steps.

Initialization

At first, we generate a regular mesh that uniformly discretizes the parameter domain of the surface (a unit square) and lift the vertices to an initial Bézier patch defined by the boundaries. In our pipeline, such a patch can be obtained in two ways:

- Generated by our prediction model, when shape optimization is used to enrich the database or to compute the undeformed shape of the final design panels.
- Initialized as a surface patch with zero twist vectors at the corners (a quad control mesh has parallelograms as corner faces) when shape optimization is used to build the initial database.

The lifted mesh is conformally flattened with minimal distortion. We uniformly resample the boundary of this mesh targeting a total number of edges M_b and triangulate the interior using Delaunay triangulation with the bounded maximal triangle area. Finally, the vertices of this mesh are mapped back to the parameter domain and lifted to the initial Bézier patch. As a result, we obtain an initial configuration for a deformed glass panel conforming to Bézier boundaries and its corresponding undeformed configuration.

Minimization

The initial solution is not in static equilibrium and has arbitrarily high stresses. We compute the minimal energy configuration by minimizing the discrete strain energies defined in Eq. 3.4 and Eq. 3.5 over deformed $\mathbf{x}$ and undeformed $\mathbf{X}$ configurations. We refer to the vector of all the deformed nodes at the boundary and the internal nodes as $\mathbf{x}_\mathrm{b}$ and $\mathbf{x}_\mathrm{i}$, respectively. To reduce the complexity of the problem and keep a high-quality triangulation of the undeformed configuration, we assume internal nodes at the rest

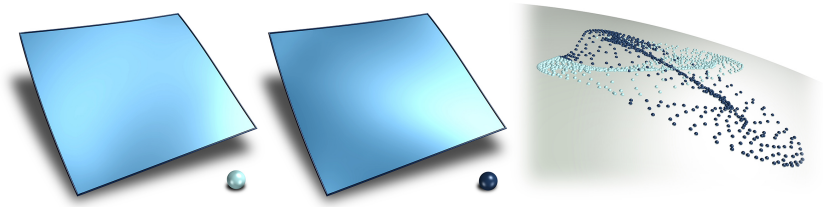


Figure 4.4: Comparison between two alternative stable equilibria for a given Bézier boundary. The two resulting panels produce radically different Gauss maps (right) leading to very distinguishable reflection effects.

configuration $\mathbf{X}_\mathrm{i}$ are computed through Laplacian smoothing of the boundary vertices $\mathbf{X}_\mathrm{i} = \mathbf{L}\mathbf{X}_\mathrm{b}$. Then, the aforementioned minimization problem results in the following:

$$\min_{\mathbf{i},\phi,\mathbf{X}_\mathrm{b}} W(\mathbf{x}, \phi, \mathbf{X}_\mathrm{b}) + R(\mathbf{X}_\mathrm{b}), \tag{4.1}$$

where W is the sum of all strain energy terms, ϕ are the mid-edge normal deviations, and R is a regularization term removing the null space due to the translation and rotation of the undeformed configuration. In particular, it is formulated as a soft constraint: the centroid of the boundary nodes is fixed to the origin, and one of the nodes has a fixed angle with the x-axis. Note that we only consider undeformed boundary nodes $\mathbf{X}_\mathrm{b}$ as DoFs of the optimization; after each solver iteration, we project the internal nodes' coordinates $\mathbf{X}_\mathrm{i}$ through Laplacian smoothing. In addition, the boundary nodes of the deformed configuration remain fixed and conforming to Bézier boundaries.

As can be seen in Fig. 4.4, minimizing Eq. 4.1 does not always produce a unique solution. For a given boundary, glass panels can potentially adopt multiple stable equilibria corresponding to locally optimal shapes that depend on the initialization of the problem. Although for some boundary curves there is a clearly preferred shape that is more energetically stable than the rest, in other cases, several stable equilibria are valid solutions that might be practically used in a feasible panelization. Furthermore, the maximum stress levels differ a lot between stable configurations. Multistability imposes two challenges for building a data-driven model of glass panel mechanics. First, we do not know in advance the number of local minima that exist for a given boundary nor how energetically stable these configurations are in practice; second, we do not know how to initialize the minimization problem to obtain such solutions. Both challenges motivated the use of a MDN as a regressor for the shape and corresponding stress of the glass panels. In Sec. 4.4, we describe our regression model and the methodology we followed to enrich the database by discovering new stable equilibria through an iterative process.

4.3.2 Failure criterion

To estimate whether the panel is going to break, we compute the maximal engineering stress across all the elements of the discretization. We estimate the stress of an element by computing the first Piola-Kirchhoff stress tensor $\mathbf{P} = \mathbf{F}\mathbf{S}$. Here, $\mathbf{F}$ is the deformation gradient of the element, and $\mathbf{S}$ is the corresponding second Piola-Kirchhoff stress tensor. In a similar fashion to Pfaff et al. [PNdJO14], we compute the total stress of a panel using our estimation of the combined bending and membrane strain introduced in Eq. 3.2:

$$\mathbf{S}\Big(\mathbf{E}(\mathbf{X}, z)\Big) = \lambda \operatorname{Tr}(\mathbf{E})\mathbf{I} + 2\mu\mathbf{E}. \tag{4.2}$$

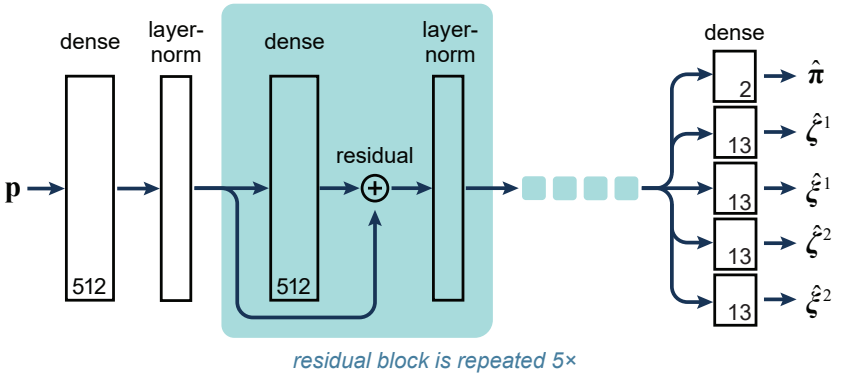


Figure 4.5: Architecture of our data-driven model. The input is a panel boundary $\mathbf{p}$; the model predicts means $\hat{\boldsymbol{\zeta}}^k$, variances $\hat{\boldsymbol{\xi}}^k$, and component weights $\hat{\boldsymbol{\pi}}$ for a two-component Gaussian mixture over the shape and stress of the minimal-energy surface. Numbers in dense layers indicate the number of output units.

The maximal engineering stress is then evaluated as the maximum absolute singular value of $\mathbf{P}$ across all elements. That is, for each element, we compute $\mathbf{S}\big(\mathbf{E}(\mathbf{X}, \pm h/2)\big)$, where the bending contribution to the stress is at its maximum, and pick the highest absolute singular value. The global maximal stress value is generally at most C^0-continuous with respect to the panel boundary curves, which makes its direct usage in a continuous optimization undesired. Instead, we compute an L_p-norm of maximal principal stress per element. In practice, we found that $p = 12$ suffices. We denote the resulting value σ and refer to it as the "maximal stress" for brevity.

Taking our assumptions, it is important to note that neither the overall shape nor the maximal stress value changes for a given panel under uniform scaling. This implies that only the ratio of the thin shell dimensions and the panel thickness matters. For simplicity, we choose 1 mm as our canonical thickness for the simulations and scale the obtained results accordingly for every other target thickness.

4.4 Data-driven model

We require a model that can efficiently predict the shape and stress of the minimum-energy panels for a given boundary. The simulation described in Sec. 4.3 calculates these quantities, but is too slow to incorporate in an interactive design tool. Our data-driven model aims to predict the deformed shape and corresponding maximum stress of the panels more efficiently. Moreover, we use it to calculate the derivatives with respect to the input boundary, which is required for gradient-based design optimization.

Therefore, we use a statistical model that maps panel boundaries to the shapes and stresses of minimal-energy conforming cold bent glass surfaces. Sec. 4.4.1 describes the model and training process. The training requires a large dataset of boundaries and the resulting panel shapes and stresses; in Sec. 4.4.2, we describe the space of boundaries we sample from and how the shape optimization and stress computation of Sec. 4.3 is applied to them. To improve the results further, we augment the dataset to better cover the regions of the input space where the predictions do not match the training data because of multistability of the glass panels (Sec. 4.4.3) and retrain the model on this enriched dataset. We will release our dataset and pretrained model publicly.

4.4.1 Multi-modal regression model

Our prediction model takes as an input a vector $\mathbf{p} \in \mathbb{R}^{18}$ representing a panel boundary. As noted in Sec. 4.3.1, several different surfaces may conform to a given boundary, corresponding to different local minima of the strain energy. Therefore, predicting a single output yields poor results, typically the average over possible shapes. Instead, we use a *mixture density network* (MDN)—a neural network model with an explicitly multi-modal output distribution [Bis06]. For a given boundary, each mode of this distribution should correspond to a different conforming surface.

Whereas training a neural network to minimize the mean squared error is equivalent to maximizing the data likelihood under a Gaussian output distribution, an MDN instead maximizes the likelihood under a Gaussian mixture model (GMM) parameterized by the network. Therefore, it must output the means and variances of a fixed number K of mixture components, as well as a vector $\hat{\boldsymbol{\pi}}$ of component probabilities. In the rest of the paper, all variables with hats denote predictions from our data-driven model, as opposed to values from the physical simulation. In our model, each component is a $(12 + 1)$-dimensional Gaussian with diagonal covariance, corresponding to the four interior control points of the shape, $\mathbf{c}_{ij} \in \mathbb{R}^3$, $i, j \in \{1, 2\}$, and stress, σ, of one possible conforming surface. We denote the mean of the concatenated shape and stress of the k^{th} mixture component by $\hat{\boldsymbol{\zeta}}^k$ and the variance by $\hat{\boldsymbol{\xi}}^k$; both are output by the neural network and hence depend on the input boundary $\mathbf{p}$ and network weights $\mathbf{w}$.

Model architecture

We use a densely connected model with six layers of 512 exponential linear units (ELU) [CUH15], with residual connections [HZRS16] and layer-normalization [BKH16] at each hidden layer (Fig. 4.5). We trained tens of models using combinations of these hyperparameters' values and selected the one that performed the best on the held-out validation set. In simulation, we observed that a given boundary could potentially admit more than two stable states. However, these cases were extremely rare; therefore, we set $K = 2$. This suffices for capturing the vast majority of stable states observed in our dataset, resulting in a low validation error. Hence, the output layer has 54 units, with no activation for the means $\hat{\boldsymbol{\zeta}}^k$, exponential activation for the variances $\hat{\boldsymbol{\xi}}^k$, and a softmax taken over the mixing probabilities $\hat{\boldsymbol{\pi}}$.

Model training

The model is trained to minimize the negative log-likelihood of a training set $\mathcal{T}$ under the GMM:

$$\mathcal{L}(\mathcal{T}; \mathbf{w}) = -\sum_{(\mathbf{p},\boldsymbol{\zeta})\in\mathcal{T}} \log \left\{ \sum_{k=1}^{K} \hat{\pi}^k(\mathbf{p}; \mathbf{w})\, \mathcal{N}\Big(\boldsymbol{\zeta} \,\Big|\, \hat{\boldsymbol{\zeta}}^k(\mathbf{p}; \mathbf{w}),\, \hat{\boldsymbol{\xi}}^k(\mathbf{p}; \mathbf{w})\Big) \right\} \tag{4.3}$$

where $\boldsymbol{\zeta}$ is a true output for panel $\mathbf{p}$, i.e. the concatenation of shape and stress from one simulation run, and $\mathcal{N}$ represents a diagonal Gaussian density. We also add an L2 regularization term with strength 10^{-4} on the weights $\mathbf{w}$, to discourage over-fitting.

We use the stochastic gradient method Adam [KB15] to minimize the above loss function with respect to the network weights $\mathbf{w}$. We use a batch size of 2048, learning rate of 10^{-4}, and early stopping on a validation set with patience of 400 epochs. We select the best

model in terms of the validation loss obtained during the training process. A single epoch takes approximately 30 seconds on a single NVIDIA Titan X graphics card, and in total, training takes around 20 hours.

Model output

For brevity, in the remainder of the paper, for a given panel boundary $\mathbf{p}$ and for a possible state $k \in \{1, 2\}$, we write:

- $\hat{\mathbf{S}}_{\mathbf{p}}^{k} : [0, 1]^2 \to \mathbb{R}^3$ for the Bézier surface patch that is defined by the boundary $\mathbf{p}$ and the predicted interior control nodes from the mean of the k^{th} component (i.e., the leading 12 elements of $\hat{\boldsymbol{\zeta}}^k$).
- $\hat{\sigma}_{\mathbf{p}}^{k}$ for the stress value (i.e., the last element of $\hat{\boldsymbol{\zeta}}^k$).
- $\hat{\pi}_{\mathbf{p}}^{k}$ for the k^{th} component probability $\hat{\pi}^k(\mathbf{p}; \mathbf{w})$.

Furthermore, we write $\hat{\mathbf{S}}_{\mathbf{p}}$ and $\hat{\sigma}_{\mathbf{p}}$ (i.e., without the k superscript) to refer to the *best* prediction for boundary $\mathbf{p}$, which is determined by two factors:

1. if any of the component probabilities $\hat{\pi}_{\mathbf{p}}^{k}$ is greater than 95% we discard the alternative and define the corresponding shape/stress prediction as best, or
2. otherwise, the best is determined depending on the application, either as the lower stress, the smoother shape, or the closer shape to a reference surface.

We discard components with $\hat{\pi}_{\mathbf{p}}^{k} < 0.05$ since the modes with a near-zero probability imply a low level of confidence in the corresponding prediction.

4.4.2 Dataset construction

To train our prediction model, we require a dataset of boundaries that is representative of our target application. These are then paired with the shapes and stresses of the conforming surfaces with minimal energy. Recall from Sec. 4.2 that a panel boundary may be parameterized invariantly to rigid transformations by corner pairwise squared-distances $\boldsymbol{d}$, edge-plane inclinations $\boldsymbol{\gamma}$, and halfedge tangent directions $\boldsymbol{\theta}$. We generate boundaries by sampling these parameters from the ranges and distributions described in Appendix A.1. Note that the physical model for the deformed shape and stress is invariant under the scaling of all geometric magnitudes; we choose our sampling ranges so that it would be possible to scale the results to panel length-to-thickness ratios commonly used in cold bent glass façades (e.g., 150–600). By applying the shape optimization described in Sec. 4.3 to these boundaries, we obtain fine discrete meshes representing the deformed cold bent panels. We obtain a Bézier representation of such panels by keeping the sampled boundaries and fitting interior control points to match the simulated surface using the method described in Sec. 4.2.1. Plus, in our representation, any non-flat panel geometry can be equivalently represented in four alternative ways, depending on vertex indexing, and can be mirrored. Therefore, we transform each simulated panel into eight samples by permuting the vertex indexes and adding their mirror-symmetric representations.

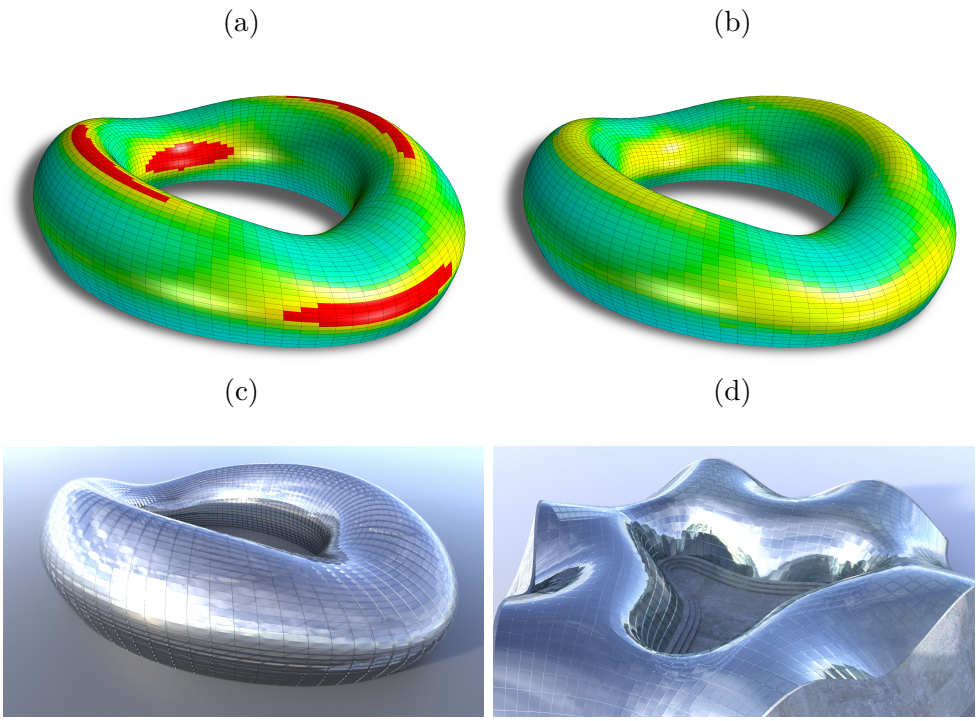


Figure 4.6: An initial design includes panels exceeding stress limits (a). It is optimized for stress-reduction (b) and rendered (c). (d) A different façade designed with our tool.

In total, we simulated approximately 1.5 million panels, which corresponds to 12 million samples after vertex permutations and adding mirror-symmetric panels. We reserved 10% of these samples as a validation set for tuning the optimization hyperparameters and network architecture. To acquire such large amounts of data requiring massive computations, we employed cloud computing.

4.4.3 Dataset enrichment

When a given boundary has multiple conforming panels, the physical simulation returns only one of these determined by the twist-free Bézier patch initialization. Conversely, our data-driven model always predicts $K = 2$ states, though one may have a very small mixture weight $\hat{\pi}^k$, indicating it is unlikely to be a valid optimal panel. We observed that after training, the model often predicts shapes for boundaries in its training set that differ from those returned by the simulation—however, re-simulating these boundaries with a different initialization recovers a solution close to that predicted by the data-driven model. This observation suggests a method to extend the dataset with new samples to improve prediction error.

Specifically, we use the prediction from the model as an initialization for the simulator, which is then likely to converge to a stable surface that was not reached from the default initialization. The resulting surface can be added to the training set, so after retraining, the model will give an even more accurate prediction in the same region of parameter space. We apply the data-driven model to every panel in the training set, and collect the predicted shapes $\hat{S}^k_{\mathbf{p}}$, $k \in \{1, 2\}$, where $\pi^k > 5\%$. For each of these, we calculate the maximum deviation of the internal control nodes along any dimension, from the true shape in the training set. We then retain the 200k panels ($\sim$15% of the original training set) for which this deviation is the largest. For each such boundary, we re-run

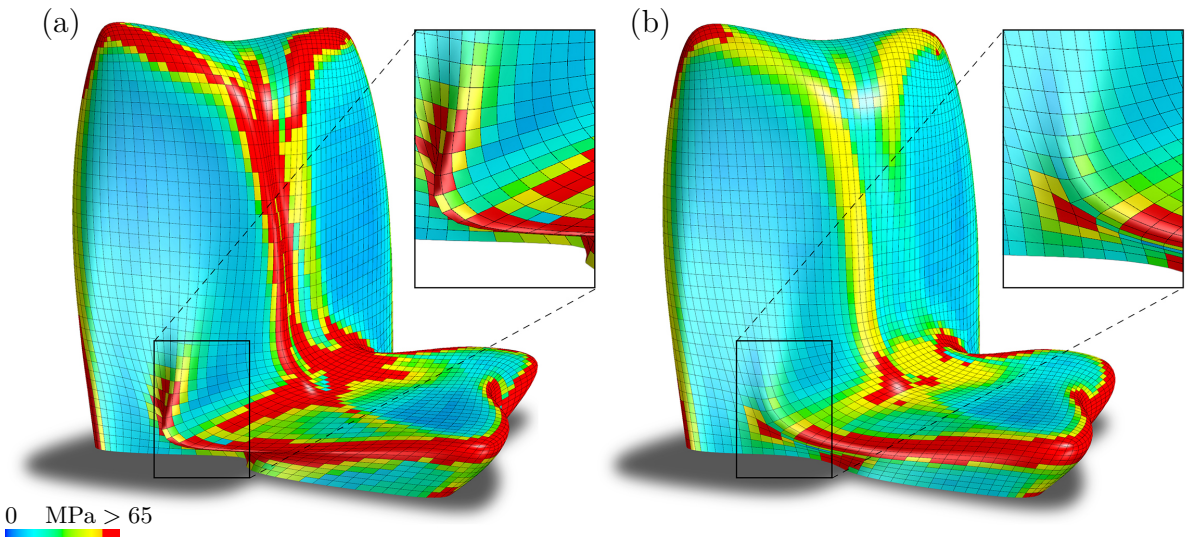


Figure 4.7: Optimization of the NHHQ skyscraper design (Zaha Hadid Architects). We first optimize for smoothness of the overall design, and then optimize selected high stress areas for stress reduction. (a) Stress on panels computed on the original shape and panel layout. Red panels exceed the threshold of 65 MPa. (b) Stress on panels after optimization. The inset shows an area with clearly visible shape change. We decrease the number of panels exceeding 65 MPa from 1517 to 874.

the simulation, using the predicted shape $\hat{S}^k_{\mathbf{p}}$ as the initialization. Finally, we select all panels which have at least 2 mm difference along any dimension of any internal control node compared to the panel obtained originally and add these to the training set. The resulting, enriched training set is used to retrain the model.

4.5 Interactive design

In this section, we show how we arrive at a practical interactive design tool for freeform surface panelization using cold bent glass panels. We aim to produce a tool compatible with the standard design workflow of an architectural designer. At every moment during the editing process, the user gets immediate feedback on the physical properties of the panelization (i.e., shape and stress predictions for the panels). Upon request, an automated process running at interactive rates uses an optimization to "guide" the design. Figs. 1.2 and 4.6 show two different doubly curved glass surfaces that have been interactively designed from scratch using our tool. Although it is generally desired to create designs free from breaking panels, in a real project, one might like to assume the cost of hot-bending a small proportion of the panels. Therefore, there is a practical trade-off between the smoothness and aesthetics of a design and its manufacturability. We consider this option by explicitly weighting various design criteria in the formulation of our inverse design problem.

4.5.1 Optimization formulation

Depending on the specific application domain, the desired properties might vary. This translates into the minimization of a composite target functional $\mathcal{E}$:

$$\mathcal{E} = w_\sigma \mathcal{E}_\sigma + w_\mathrm{s}\mathcal{E}_\mathrm{s} + w_\mathrm{f}\mathcal{E}_\mathrm{f} + w_\mathrm{p}\mathcal{E}_\mathrm{p} + w_\mathrm{c}\mathcal{E}_\mathrm{c}. \tag{4.4}$$

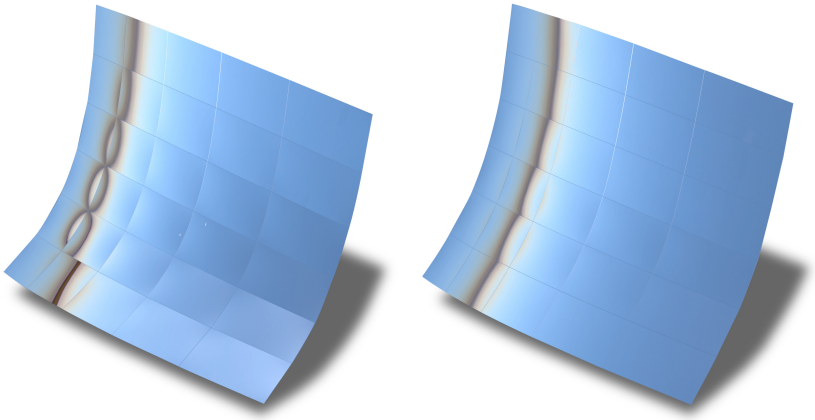


Figure 4.8: Effect of optimization on visual smoothness. On the left, a selection of cold bent panels computed on a given layout. On the right, the same panels after optimization of the layout for the kink angle and bending stress reduction.

Overall, the total energy $\mathcal{E}$ depends on the vertex positions $\mathbf{v} \in V$, the edge plane vectors $\mathbf{s}_e$ defining the plane Π_e associated with e, the tangent angles θ_i, and some auxiliary variables associated with inequality constraints. Each weighted contribution to $\mathcal{E}$ represents a desired property of the final design, which we discuss in detail in the following sections.

Panel stress $\mathcal{E}_\sigma$

The most important property is the manufacturability of the final design. Failure in a specific type of glass is modeled by estimating the maximal stress present in the glass panel and comparing it to the maximum allowed stress value.

The MDN from Sec. 4.4 acts as a stress estimator. We constrain the predicted stress value $\hat{\sigma}_\mathbf{p}$ for a given boundary $\mathbf{p}$ to be less than a stress bound $\sigma_{\max}$. We assign $\sigma_{\max}$ to a value lower than the stress value at which failure occurs, taking into account a safety factor and the estimator error. The inequality constraints $\hat{\sigma}_\mathbf{p} \leq \sigma_{\max}$ are converted to equality constraints by introducing an auxiliary variable $u_\mathbf{p} \in \mathbb{R}$ per panel boundary $\mathbf{p}$, and formulating the manufacturability energy as

$$\mathcal{E}_\sigma = \sum_\mathbf{p} (\hat{\sigma}_\mathbf{p} - \sigma_{\max} + u_\mathbf{p}^2)^2. \tag{4.5}$$

Fig. 4.10 shows the effect of limiting the maximum stress of the design for a section of the façade of the Lilium Tower.

Smoothness $\mathcal{E}_\mathrm{s}$

Here, we collect some terms in the final objective function that aim in various ways to obtain as smooth as possible panelizations. As shown in Fig. 4.8, this is essential for achieving the stunning look of curved glass façades because it greatly affects the reflection pattern. The smoothness term is the sum of two individual functionals, i.e. $\mathcal{E}_\mathrm{s} = \mathcal{E}_1 + \mathcal{E}_2$.

Kink angle smoothing $\mathcal{E}_1$

It is generally not possible to get smooth connections along the common boundary curves of panels, but we can try to minimize the kink angle. For each pair of faces f_i, f_j sharing a common edge e, we consider their respective predicted panels $\hat{\mathbf{S}}_i$, $\hat{\mathbf{S}}_j$ and minimize the angle between their surface normals $\mathbf{n}_i$, $\mathbf{n}_j$ evaluated at the parameter $t = 0.5$ of the shared curve

$$\mathcal{E}_1 = 0.1 \sum_{e \in E_I} (1 - \mathbf{n}_i \cdot \mathbf{n}_j)^2, \tag{4.6}$$

where E_I is the set of interior edges of $\mathcal{M}$, and 0.1 is a suitable importance weight within the smoothness term. Note that $\hat{\mathbf{S}}_i$, $\hat{\mathbf{S}}_j$ are shape predictions for the respective boundary curves of the two faces f_i, f_j. Thus, optimization involves computing the Jacobian of the MDN output w.r.t. the input boundaries. Fig. 4.9 shows the effect of including the kink smoothing term in the design of the NHHQ façade.

Curve network smoothing $\mathcal{E}_2$

Each edge in the dominant mesh polylines of $\mathcal{M}$ determines a cubic patch boundary curve, and the sequence of these curves should also be as smooth as possible. At each connection of two edges, the corresponding tangents should agree, and thus, the inwards directed unit tangent vectors satisfy $\mathbf{t}_i = -\mathbf{t}_j$, or equivalently $\mathbf{t}_i \cdot \mathbf{t}_j + 1 = 0$. This tangent continuity constraint explains the first part in the smoothness term

$$\mathcal{E}_2 = \sum (\mathbf{t}_i \cdot \mathbf{t}_j + 1)^2 + \sum \left[\mathbf{s}_e \cdot (\mathbf{n}_i + \mathbf{n}_{i+1})\right]^2. \tag{4.7}$$

The second part concerns the planes Π_e. We consider an edge e with endpoints $\mathbf{v}_i$, $\mathbf{v}_{i+1}$. The discrete osculating plane at $\mathbf{v}_i$ is spanned by $(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})$ and has a unit normal $\mathbf{n}_i$. Likewise, $(\mathbf{v}_i, \mathbf{v}_{i+1}, \mathbf{v}_{i+2})$ defines a discrete osculating plane with normal $\mathbf{n}_{i+1}$ at $\mathbf{v}_{i+1}$. We want Π_e to be the bisecting plane between these two, i.e. $\mathbf{s}_e \cdot (\mathbf{n}_i + \mathbf{n}_{i+1}) = 0$. Of course, the sums are taken over all occurrences of the described situations.

Finally, in practice, a few other parts are added to the smoothness term $\mathcal{E}_\mathrm{s}$, which concern special cases. At combinatorially singular vertices of $\mathcal{M}$, we constrain the tangent vectors to lie in a tangent plane. Plus, there are various symmetry considerations that are used at the boundary, but those could easily be replaced by other terms with a similar effect.

Mesh fairness $\mathcal{E}_\mathbf{f}$

So far, we have dealt with the smoothness of the panelization to a given mesh $\mathcal{M}$. Because we also allow the mesh $\mathcal{M}$ to change during the design, we need to care about its fairness. This is done in the standard way using second-order differences of consecutive vertices along dominant mesh polylines,

$$\mathcal{E}_\mathrm{f} = \sum (\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1})^2. \tag{4.8}$$

Proximity to reference mesh $\mathcal{E}_\mathbf{p}$

When designing a panelization for a given reference geometry, it is not sufficient to have the mesh $\mathcal{M}$. One will usually have a finer mesh $\mathcal{M}_\mathrm{ref}$ describing the reference geometry (Fig. 4.1). To let $\mathcal{M}$ change but stay close to the reference surface, we need a term that allows for the gliding of $\mathcal{M}$ along $\mathcal{M}_\mathrm{ref}$. This is done in a familiar way: to let a vertex $\mathbf{v_i}$

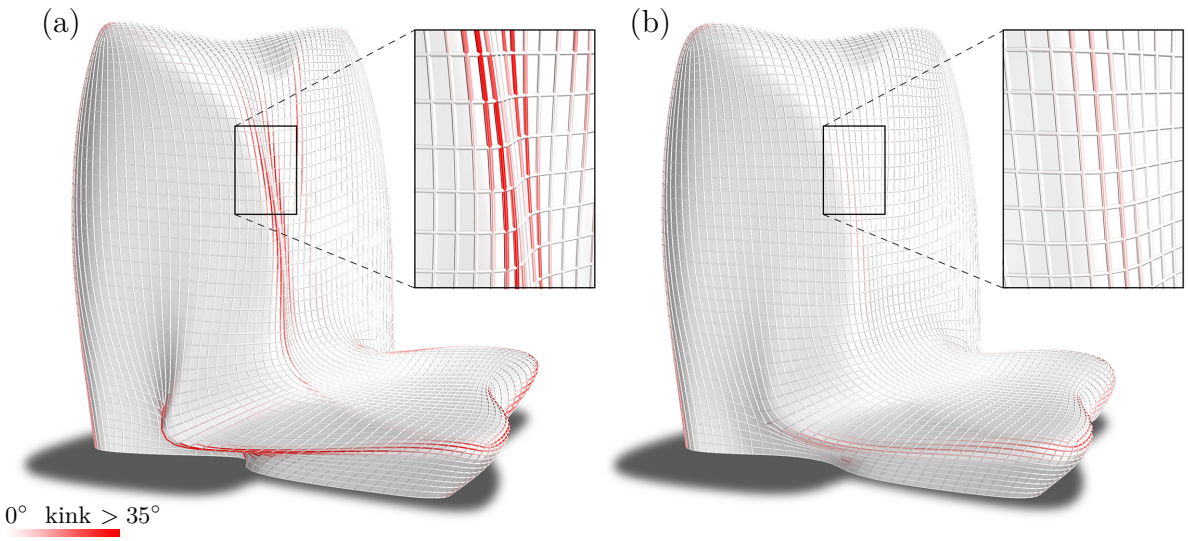


Figure 4.9: Comparison of the kink angle between panels in the NHHQ model, before (a) and after (b) running our design optimization algorithm. As a result, the mean kink angle is lowered from 3.7° to 2.7°, while the maximum was reduced from 60.9° to 36.0°.

stay close to $\mathcal{M}_{\text{ref}}$, we consider its closest point $\mathbf{v}_i^*$ on $\mathcal{M}_{\text{ref}}$ and the unit surface normal $\mathbf{n}_i^*$ at $\mathbf{v}_i^*$. In the next iteration, $\mathbf{v_i}$ shall stay close to the tangent plane at $\mathbf{v}_i^*$, which is expressed via

$$\mathcal{E}_{\text{p}} = \sum_{\mathbf{v}_i \in V} \left[(\mathbf{v_i} - \mathbf{v}_i^*) \cdot \mathbf{n}_i^* \right]^2 . \tag{4.9}$$

Design space constraints $\mathcal{E}_{\text{c}}$

Since we want the neural network to produce reliable estimates, we need to ensure the panel boundary curves remain within the range used for training (Sec. 4.4.2). This is achieved as the sum of two constraint functionals $\mathcal{E}_{\text{c}} = \mathcal{E}_3 + \mathcal{E}_4$. First, we constrain the tangent angles to $|\theta_i| = \angle(\mathbf{t}_i, \mathbf{e}_i) \leq 4.9°$ for all angles θ_i of halfedges $\mathbf{e}_i$ with tangent vectors $\mathbf{t}_i$. We again convert the inequality constraints to equality constraints by introducing auxiliary variables u_i,

$$\mathcal{E}_3 = \sum (\theta_i^2 - (4.9°)^2 + u_i^2)^2. \tag{4.10}$$

Second, we are working under the assumption that the vectors $\mathbf{s}_e$ are unitary and orthogonal to their respective edges e, which results in

$$\mathcal{E}_4 = \sum_e [(\mathbf{s}_e \cdot \mathbf{e})^2 + (\mathbf{s}_e^2 - 1)^2]. \tag{4.11}$$

4.5.2 Optimization solution

The minimization of $\mathcal{E}$ results in a nonlinear least-squares problem that we solve using a standard Gauss-Newton method. The derivatives are computed analytically, and, since each distinct term of $\mathcal{E}$ has local support, the linear system to be solved at each iteration is sparse. We employ Levenberg-Marquardt regularization and sparse Cholesky factorization using the TAUCS library [Tol03].

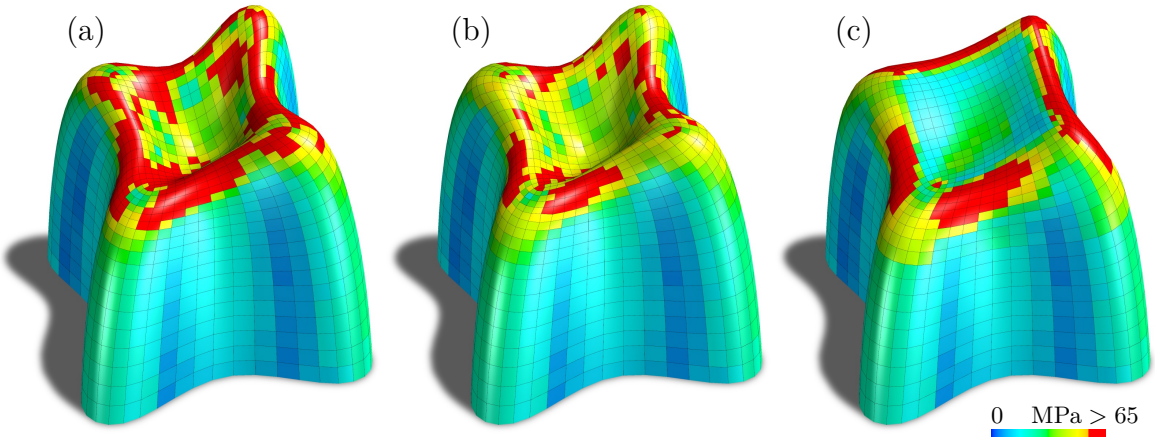


Figure 4.10: Optimization of the Lilium Tower (model by Zaha Hadid Architects) for different target properties. (a) Stress values for the initial panel layout. (b) Optimizing the design only for stress reduction and proximity to the original design leads to more panels within the stress threshold, but also to a non-smooth curve network. (c) Allowing the design to deviate from the input and including fairness, produces a smoother result with reduced stress. Number of panels exceeding 65 MPa amounts to respectively 293, 131, and 225.

Initialization

The edge plane vector $\mathbf{s}_e$ of an edge e is initialized so that Π_e is the bisecting plane of two discrete osculating planes, as in the explanation for Eq. 4.7. The angles θ_i are initialized so that they are at most 5° and so that the tangents lie as close as possible to the estimated tangent planes of the reference geometry. After initializing all other variables and computing an estimated stress value per face panel, the auxiliary variables are initialized such that they add up to the inequality constraint bound or zero otherwise (i.e., the inequality constraint is not satisfied). The shape $\mathbf{S_p}$ of each panel is initialized with the MDN prediction using the initial boundary parameters $\mathbf{p}$. In case there are two possible shapes, we use the one that provides the best solution considering application-dependent criteria (e.g., stress reduction). When looking for the smoothest fit, we pick the one minimizing $\sum(1 - \mathbf{n}_i \cdot \mathbf{n}_e)^2$, i.e., a measure of angle deviation between each edge normal (sum of two adjacent face normals orthonormalized to $\mathbf{e}$) and the surface normal $\mathbf{n}_i$ evaluated at the parameter $t = 0.5$ of the edge curve.

Optimization weights

The weights associated with the target functional $\mathcal{E}$ act as handles for the designer to guide the output of the optimization toward the desired result. We do not opt for a fixed weight configuration because the ideal balance is not uniquely defined, but is instead governed by project-dependent factors such as budget and design ambition.

In all our experiments, we found it sufficient to assign the weights either to zero or to the values $\{10^{-2}, 10^{-1}, 10^{0}\}$. Fig. 4.10 shows one example of the different effects possible when changing the property importance. In practice, and as a rule of thumb for a standard optimization where we prioritize stress reduction and smooth panels (in that order), we use weight values $w_\sigma = 1$, $w_\mathrm{s} = w_\mathrm{c} = 10^{-2}$, and $w_\mathrm{p} = w_f = 10^{-1}$. We also reduce the fairness importance at the i-th optimization iteration by scaling its weight by 0.9^i.

4.6 Results

4.6.1 Experimental validation

We experimentally validated our simulation results and design workflow. For practical reasons, the experiments were done at a small scale using borosilicate thin glass of about 180×130 mm^2 and 0.35 mm thickness.

For the validation of the simulation results (see Fig. 4.12), high precision frames were machined from cast aluminum. The glass panel is pressed down on a 2 mm wide smooth support frame by a dense array of stainless steel finger springs which are cushioned by 0.5 mm polytetrafluoroethylene (PTFE). The support frame matches a thin boundary strip of the simulated glass panel. To test the accuracy of the predicted shape, we selected and 3D-scanned a panel with a high estimated maximal stress (98 MPa), which is beyond our safety limit but still manufacturable (see Fig. 4.12). The obtained surface was registered to the output of our shape optimization routine, and we observed a worst-case deviation of 0.12 mm. Note that we registered an offset surface from the optimal mid-surface to account for the glass thickness.

The frames for the design model created with our tool are illustrated in Fig. 4.11 and were built from laser cut and welded 1.2 mm thick stainless steel sheet metal. The glass, which is cushioned by tape, is pressed down on to the frame by L-shaped stainless steel fixtures spot welded to the frame. The presented design model is negatively curved and consists of nine individual panels, each about 200×170 mm^2 in size. The expected stress levels range from 20 to 62 MPa. As predicted, all panels are fabricable and intact.

During bending, our panels usually do not need to go through more extreme deformations than the final one, meaning that we do not expect higher stresses while bending. Because normally panels have a dominant bending direction, we observed that it is possible to “roll” the glass onto the frame accordingly during assembly. Clamping the glass to the frame fixes the normals at the boundary, which makes one of the alternative shapes preferable.

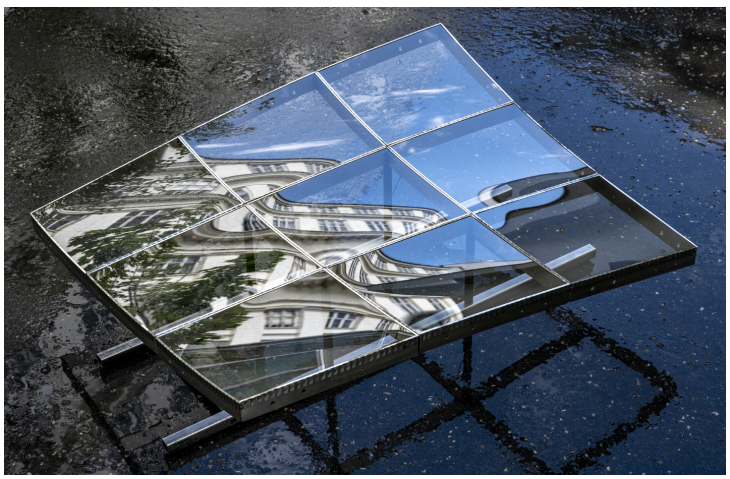


Figure 4.11: Realization of a doubly curved surface using 3x3 cold bent panels.

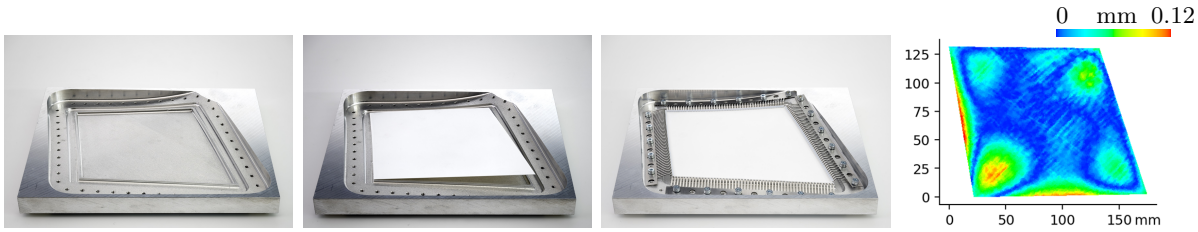


Figure 4.12: A doubly curved panel of a thickness of 0.35 mm with off-plane corner deviation of 6.9 mm. A white coating has been applied to it for 3D-scanning. Right: deviation from the simulation by at most 0.12 mm.

4.6.2 Validation of data-driven model

We evaluate the shape prediction on panels with maximal stress below 65 MPa, which results in a mean average error (MAE) of ∼0.5 mm. Note that this is significantly less than the assumed 1 mm thickness of the glass. We evaluate stress on panels whose true maximal stress is in the range 50–65 MPa (our region of interest); our predictions have MAE of ∼2.9 MPa. Moreover, the 67^{th} percentile error is 2.5 MPa, and the 88^{th} percentile error is 5 MPa. In addition, we evaluate how often our model correctly predicts whether or not the actual maximal stress value (in contrast to the L_p-norm) exceeds the 65 MPa threshold. From our test set, we obtained below 1% of false negatives (when the model incorrectly predicts the panel is feasible) and 15% of false positives.

4.6.3 Applications

From a manufacturing point of view, the simplest solution to clad architectural surfaces is the use of planar panels. However, this simplification sacrifices the visual smoothness of the surface. Moreover, planar panels impose a restriction on the panels layout, and in negatively curved areas, there is often no other choice than to follow the principal

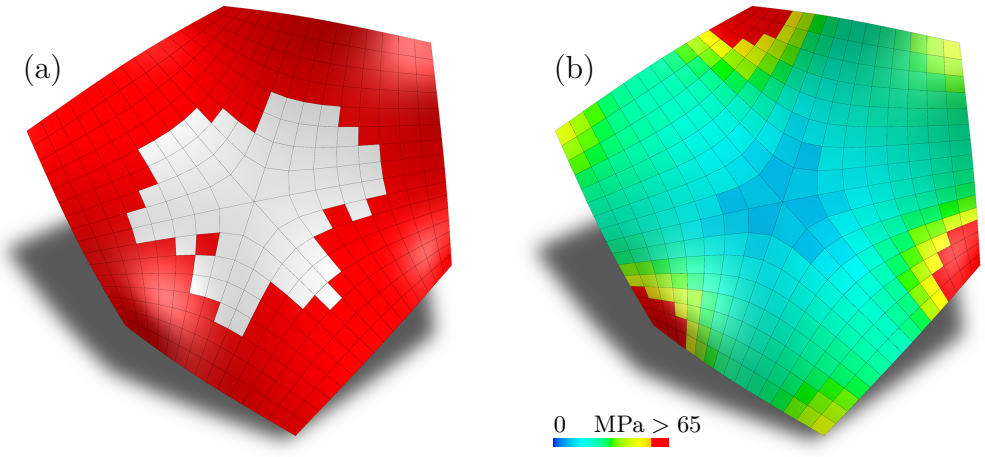


Figure 4.13: Bent glass capabilities. (a) A quadrilateral mesh where the red faces exceed a deviation of a planarity of 0.02 (measured as the distance between the diagonals divided by average edge length) and, therefore, not suitable for a flat glass panelization. (b) A cold bent panelization with corresponding face stresses. The stress values for the six central panels have been computed via simulation because they were outside the MDN input domain. According to a stress limit of 65 MPa, most of the panels optimized are feasible. The resulting cold bent panelization is shown in Fig. 4.14.

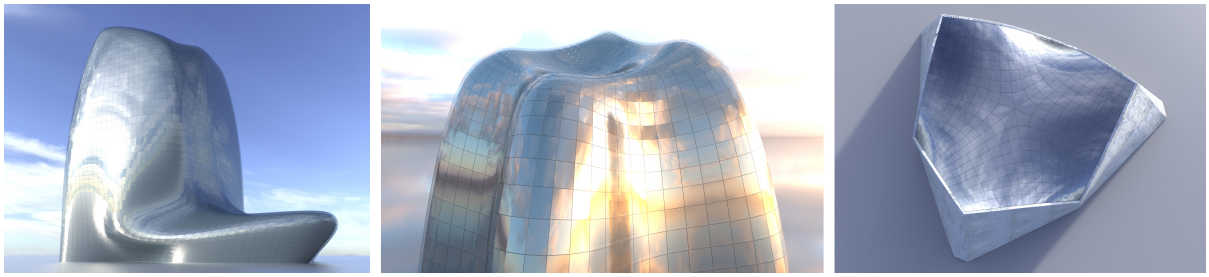


Figure 4.14: Dominant cold bent glass realizations of the NHHQ model (left). The Lilium Tower (center) after optimization for smoothness and stress reduction. The surface from Fig. 4.13 as an architectural design (right). Panels exceeding the maximum stress (check Figs. 4.7, 4.10, 4.13) are realized with hot bending.

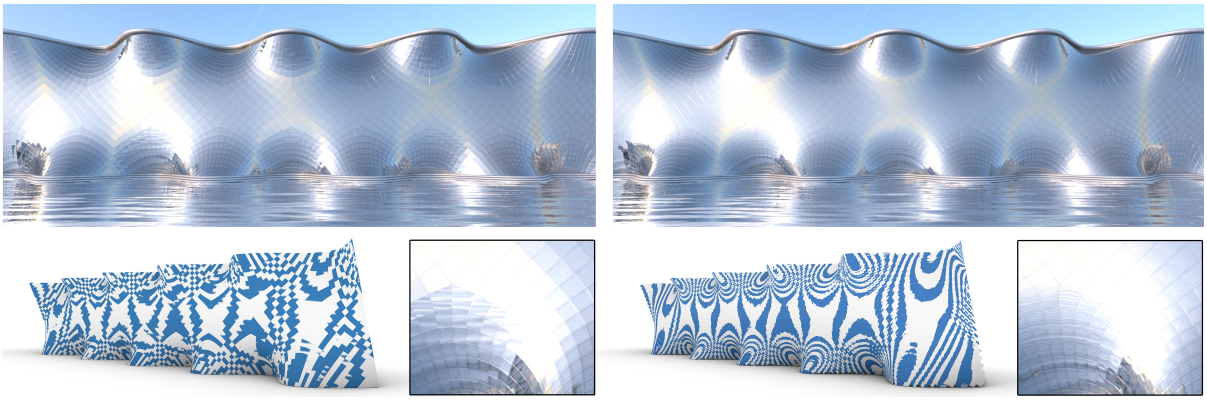


Figure 4.15: Doubly curved surface panelized using a planar quad mesh following the principal curvature network (left). This is the smoothest possible panelization of this surface achievable with flat panels [PKD+19]. The solution using cold bent glass panels designed with our method (right) shows much smoother results. We apply the shading technique of zebra striping (reflections of an infinite array of parallel light strips) for both solutions. The resulting patterns are shown in the bottom images; clearly smoother stripes are indicators of higher visual smoothness.

curvature directions of the surface. On the other hand, a panelization with doubly curved panels is often prohibitive because of the high production cost of custom molds. Cold bent glass can be then a suitable solution. In Fig. 4.15, we compare the visual appearance of the smoothest possible panelization achievable with planar panels with a cold bent one, while in Fig. 4.13, we show a panelization layout that is mostly feasible with cold bent glass, but not with planar panels. In the following, we illustrate how users can employ our workflow for architectural panelization and design.

Façade panelization

In this case, the input is a quadrilateral mesh that encodes both the design shape and the panel layout. Once the edges of the input mesh are smoothed via cubic Bézier curves, we can predict the panels' shapes and stresses. Those panels exceeding the failure criterion shall be realized with custom molds. At this point, the user can optimize the shape for the reduction of stress and kinks between panels, and tune the weights described in Sec. 4.5.2 for choosing an appropriate compromise between fidelity with the original shape, number of custom molds needed, and visual smoothness (see Figs. 4.7, 4.9, and

4.10). To show cold bent glass' capabilities in façade panelization, we tested this workflow on the challenging NHHQ and Lilium Tower models by Zaha Hadid Architects, which were never realized. The initial quadrangulations for these two designs are planar quad meshes which were created by the architect. The results are shown in Fig. 4.14.

Façade design

Besides the panelization of a given shape, our workflow is very well suited as an interactive design tool. In this case, the user can interactively modify the quad mesh that represents the panel layout and gets immediate feedback on which panels can be produced with cold bent glass, while exploring different designs. Initial values are computed as in Sec. 4.5.2 and we use the mesh vertex normals for the estimated tangent planes at the vertices. The estimation times are compatible with an interactive design session. Once the user is satisfied with a first approximate result, the panelization can be further optimized, as described in Sec. 4.6.3, to improve the smoothness and reduce panel stresses. In this step, we can further reduce the number of panels that are not feasible for cold bending. Figs. 1.2, 4.6, and 4.13 show some sample architectures designed with this procedure. Furthermore, the accompanying video demonstrates the interactive feedback capabilities of our system for efficient form finding and exploration of the constrained design space while keeping the designer in the loop.

All interactive design sessions were performed on an Intel® Core™ i7-6700HQ CPU at 2.60 GHz and NVIDIA GeForce GTX 960M. The MDN is implemented in `TensorFlow` and is run on the GPU. For 1k panels, the prediction time is 0.1 seconds while the optimization averages 3 seconds per iteration. We usually deal with less panels because we target the selected high-stress areas of the overall design. A total of 10–20 iterations are enough for the desired results. In comparison, our shape optimization, as described in Sec. 4.3, implemented in `C++` and using the `IPopt` optimization library with code-generated derivatives takes around 35 seconds on average for a single panel with $\sim 10^3$ elements. Note that this routine is not fully optimized for speed because it is not required during the interactive phase but mainly used for acquiring training data.

4.7 Discussion

In this chapter, we have introduced an interactive, data-driven approach for material-aware form finding of cold bent glass façades. It can be seamlessly integrated into a typical architectural design pipeline, allows non-expert users to interactively edit a parametric surface while providing real-time feedback on the deformed shape and maximum stress of cold bent glass panels, and it can automatically optimize façades for fairness criteria while maximal stresses are kept within glass limits. Our method is based on a deep neural network architecture and multi-modal regression model. By coupling geometric design and fabrication-aware design, we believe our system will provide a novel and practical workflow, allowing to efficiently find a compromise between economic, aesthetic, and engineering aspects.

Identifying such a compromise usually involves multiple competing design goals. Although we have demonstrated the applicability of our system for several design criteria, it would be interesting to extend the design workflow by adding capabilities, for example, for strictly local edits, marking some panels as a priori hot bent or specifying kink edges. Because

of our differentiable network architecture, in theory, it should be trivial to incorporate additional criteria into our optimization target functional or even employ a different numerical optimization algorithm if desired.

Similar to all data-driven techniques, we should only expect accurate predictions from our network if similar training data was available. Surprisingly, we noted that we were able to discover stable states that we initially did not find with the traditional optimization approach, and used these to enrich our database. Each boundary in the training set is associated with a single stable surface output by the simulator. Nevertheless, the network may correctly predict the existence of a second stable state for that boundary, because its predictions implicitly incorporate information from similar panels in the training set, where the second state is seen. However, we cannot guarantee that our database contains all relevant stable states and that all of them will be predicted. Identifying *all* stable states and optimally sampling the database using this information would be an interesting avenue for future work. For fabrication, in our experiments, reproducing the desired particular state was trivial and emerged when intuitively attaching the glass to the frame.

In the presence of more than one potential state, our system currently selects in each iteration per panel the state that best fits our application-dependent criteria. An alternative would be to compute a global, combinatorial optimal solution among all potential states. However, because of the combinatorial complexity, this would result in a much harder and probably computationally intractable optimization problem. We also considered solving the combinatorial problem by using a continuous relaxation but ultimately did not find evidence in our experiments that would indicate the need for such an approach as we observed stable convergence to satisfactory results. However, identifying the global minimum would nevertheless be an interesting research challenge.

In the following two chapters, we turn from the shells deformed by the external loads to shells transforming themselves with the actuation process driven by the internal forces. Additionally, such shells provide ways to encode the target geometry and deformation rates, which are introduced in Chapters 5 and 6 correspondingly.

CHAPTER 5

Encoding geometric information in self-morphing shells

In this chapter, we present a novel self-morphing shell actuation mechanism, called *CurveUps*, based on contractile membranes and irregular tessellation of rigid tiles. When relaxed, the membranes bring the rigid tiles together, driving the shape change. While our approach is based on a uniformly stretched elastic sheets and does not rely on external stimuli, our approach might theoretically be extensible with some of the other existing actuating materials. We present our two-step inverse design approach to replicate target doubly curved geometries. The first step quickly provides a coarse solution which is locally refined during the second step based on physical simulation.

5.1 Overview

The goal of our method is to approximate an input 3D model from an initially *flat* tile-based assembly that can be shaped into a *stable 3D configuration* without the need for any connectors or manual assembly. Our shells are made of flat, rigid tiles, which have a frustum-like shape. When aligned in 3D, they form a shell where all faces between neighboring elements are in contact. The individual elements are only connected via two thin elastic sheets, which represent the front and back covers of the shell. The elastic sheets are pre-stretched in the initial flat configuration and, once released, exert contraction forces between the elements. Finding an optimal design for such a flat assembly is very challenging, as the shape in 3D has to be stable under the contracting forces: the forces resulting from the contraction of the elastic sheet must balance the contact forces of the elements. To solve this problem efficiently, we propose a two-step optimization procedure. Fig. 5.1 illustrates the workflow. First, we compute an initial flat configuration and its corresponding 3D shape. For this, we uniformly remesh the input surface into triangles with the target edge length L that is well suited for our output device. Then, we thicken the triangles, which provides the initial shape of our elements, and pack them into a 2D layout. We use an approximate model that represents resulting forces between elements as linear springs, with empirically determined rules for a favorable 2D layout formulated as soft constraints and hard constraints that guarantee fabricability. This model also takes the shape and placement of connectors into account, which greatly influences the resulting forces. As shown in our results section, this model provides a high-quality approximation,

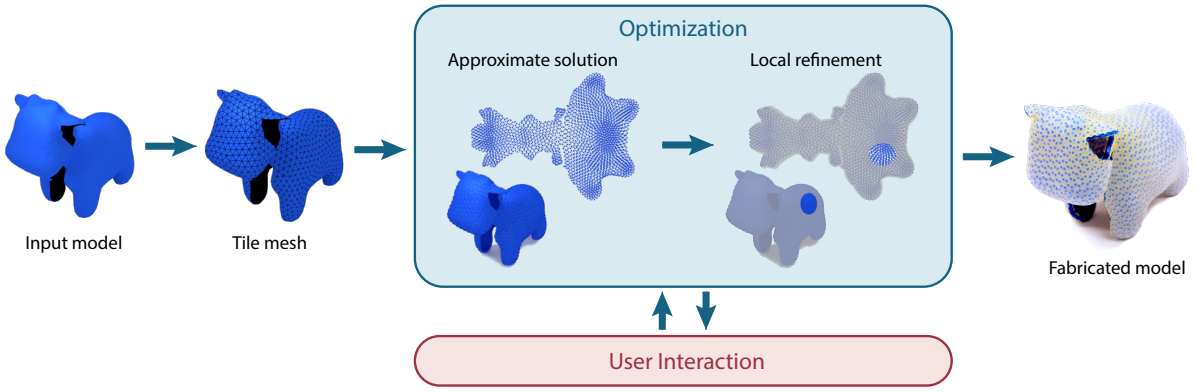


Figure 5.1: Overview of our workflow: the user provides a target mesh, the system builds the initial tile layout and finds an approximate configuration of tiles while allowing the user to make cuts in the 2D layout. The approximate solution is then refined locally using the physical model. Finally, the structure is fabricated as a flat piece that is structurally stable in its actuated configuration.

which in most cases is already ready for fabrication. However, to guarantee physical validity, we switch to a more accurate finite element model with a Neo-Hookean material model for the elastic forces and compute a configuration that simultaneously optimizes the 2D layout, the shape of the individual elements, and the connectors to the elastic sheet. Finally, the obtained tile distribution is ready for fabrication and assembly.

5.2 Model

In this section, we formally introduce our model and design parameters. We start by introducing the geometry of the tiles. Then, we describe the representation used for the flat and actuated configurations. Finally, we explain the force balance as an essential feature of the actuated configuration.

Tiles and pins

Each tile consists of its body and the pins attached to it. The body is a *frustum* defined by a pair of triangles with pairwise parallel edges representing its front and back sides. Thus, the body can be either a prism or a truncated pyramid. The distance between the triangles is the tile body thickness H_{body}. Each tile body triangle has a pin attached to it, which is a right triangular prism of height H_{pin} with the base edges parallel to the tile triangle edges pairwise. In this construction, we parametrize each pin by the pairwise distances from its edges to the edges of the corresponding tile triangle. Hence, for each tile, there are six parameters, which are denoted as

$$\mathbf{q} = q_{it}^{s},$$

where $s \in \{\mathrm{f}, \mathrm{b}\}$ denotes the front and back sides of tiles, $i \in [1, M]$ denotes tiles, M is the total number of tiles, and $t \in \{1, 2, 3\}$ denotes the edges of the tiles' triangular faces. A detailed illustration can be seen in Fig. 5.2.

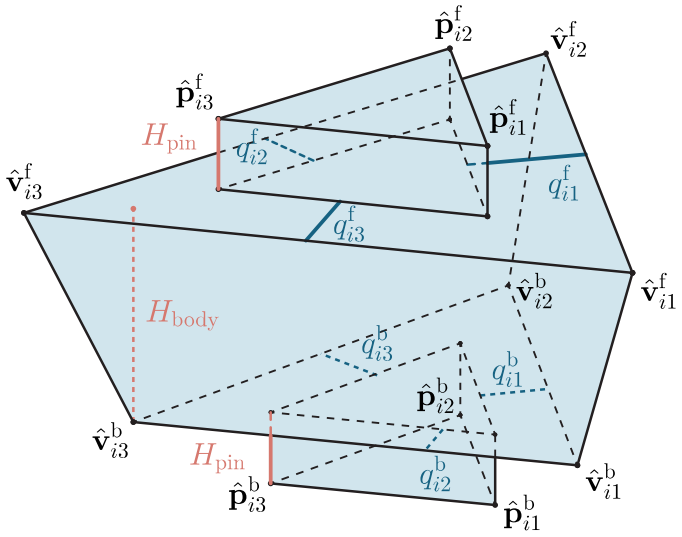


Figure 5.2: One tile with notation.

Flat configuration

In the flat configuration, the layout of triangular tiles is represented by the set of maps from the actuated to the flat configuration for each tile i:

$$\omega_i : \hat{\mathbf{v}}^{\mathrm{f}}_{it} \mapsto \bar{\mathbf{v}}^{\mathrm{f}}_{it},$$

where $\hat{\mathbf{v}}^{\mathrm{f}}_{it} \in \mathbb{R}^3$ and $\bar{\mathbf{v}}^{\mathrm{f}}_{it} \in \mathbb{R}^2$ are vertex coordinates of tile front faces in the actuated and flat configurations, respectively. In order to avoid deformation of the faces by the given mapping, we constrain the edge lengths of the triangles so that, effectively, each tile remains rigid and can be represented by only three degrees of freedom.

Actuated configuration

Tiles in the actuated configuration are represented by a triangle mesh $\mathcal{M}$, containing M faces. Each face defines the front face of a tile, while the opposite face is defined solely by the tile thickness H_{body} and dihedral angles formed with neighboring faces. This parameterization implies that the front faces of the tiles should perfectly match in the actuated configuration. We denote N as the number of vertices in $\mathcal{M}$ with coordinates $\mathbf{x}_j$ for $j \in [1, N]$. Note that following this construction might produce small intersections between tiles in the actuated configuration. We handle this problem by removing intersecting parts in a post-process prior to fabrication.

In summary, as illustrated in Fig. 5.3, the design space of our model is defined by the following:

- the vertices in the actuated configuration, $\mathbf{x} \in \mathbb{R}^{3N}$,
- the maps of faces of $\mathcal{M}$ from the actuated to flat configuration, ω_i, effectively in $\mathbb{R}^{3M}$,
- the pin parameters, $\mathbf{q} \in \mathbb{R}^{6M}$,

which gives a total number of degrees of freedom of $3N + 9M$.

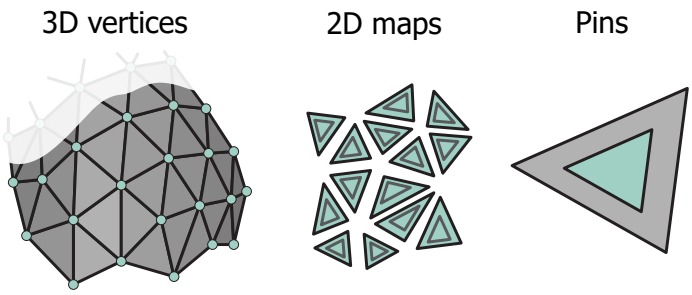


Figure 5.3: Degrees of freedom highlighted with green color (from left to right): vertex coordinates $\mathbf{x}$ of the triangle mesh $\mathcal{M}$, maps of triangles from the actuated to flat configuration ω_i, and pin parameters $\mathbf{q}$.

Mechanical equilibrium

During actuation, the pre-stretched elastic membranes push the tiles together. Eventually, tiles collide, producing contact forces that balance out the elastic forces. We use this equilibrium condition to compute the contact forces as a least squares problem:

$$\begin{aligned} &\text{minimize} && \sum \|\mathbf{f}^{\mathrm{c}}\|^2 \\ &\text{subject to} && \text{static equilibrium,} \end{aligned} \tag{5.1}$$

where static equilibrium is expressed as

$$\begin{aligned} \mathbf{f}_i^{\mathrm{c}} + \mathbf{f}_i^{\mathrm{e}} &= 0, \\ \mathbf{t}_i^{\mathrm{c}} + \mathbf{t}_i^{\mathrm{e}} &= 0, \end{aligned} \tag{5.2}$$

for each i-th tile, $\mathbf{f}_i^{\mathrm{c}}$ and $\mathbf{f}_i^{\mathrm{e}}$ represent the total contact and elastic forces, respectively, and $\mathbf{t}_i^{\mathrm{c}}$ and $\mathbf{t}_i^{\mathrm{e}}$ represent the total torques with respect to any given point generated by the contact and elastic forces, respectively. Since the choice of the point is arbitrary, in practice, we compute the torques relative to a preselected front face vertex for each tile. Given a point $\mathbf{r}_i$, the torques are computed as $\mathbf{t}_i^{\mathrm{c}} = \mathbf{r}_i \times \mathbf{f}_i^{\mathrm{c}}$ and $\mathbf{t}_i^{\mathrm{e}} = \mathbf{r}_i \times \mathbf{f}_i^{\mathrm{e}}$.

We compute the elastic forces per pin vertex by solving a quasi-static problem on the thin sheet, as described in Sec. 3.3. The contact forces are computed for each of the four contact points of each contact side of the tile (see Appendix B.1 for computation of the contact points). We enforce Newton's third law for each pair of contact forces applied to the same contact point shared between two neighboring tiles.

Note that while the front faces of the tiles always match perfectly, the back faces might need to be cut in order to avoid self-intersections with neighboring tiles (see Fig. B.1). Thus, we cut each tile at each back face vertex with a plane so that the tiles no longer intersect each other (see Appendix B.1).

While this model ensures physically valid contact forces, they might be not strong enough or inverted (meaning that they are pulling tiles apart instead of pushing them together), producing an invalid design. Thus, in order to find a set of model parameters that ensures a valid design, we formulate a constrained optimization problem, which is described in the next section.

5.3 Optimization

The goal of our system is to approximate an input 3D model with the surface generated by the tiles in the actuated configuration. We formulate this goal as a constrained nonlinear optimization problem, where soft constraints represent desired properties of the final assembly while hard constraints are used to capture fabrication and actuation limitations:

$$\begin{aligned} \underset{\mathbf{P}}{\text{minimize}} \quad & E_{\text{target}} + w_{\text{tight}} E_{\text{tight}} \\ \text{subject to} \quad & g_a(\mathbf{P}) \leq 0, h_b(\mathbf{P}) = 0, \end{aligned} \tag{5.3}$$

where E_{target} measures the deviation of the actuated mesh $\mathcal{M}$ from the target mesh $\mathcal{T}$, E_{tight} represents the tightness of bonds between tiles, $g_a(\mathbf{P})$ and $h_b(\mathbf{P})$ are the hard inequality and equality constraints, respectively, and $\mathbf{P}$ is a concatenation of all model parameters $\mathbf{x}_j$, $\bar{\mathbf{v}}^{\text{f}}_{it}$, q^s_{it}. In order to allow the user to control the behavior of the optimization, we introduce a weight, w_{tight}, which defines the tradeoff between structural tightness and deviation from the target shape.

Target energy

Our goal is to obtain a shape that closely resembles the target shape. The target energy measures the deviation of the surfaces and is measured as an area-weighted vertex to vertex distance:

$$E_{\text{target}}(\mathbf{x}) = \sum_j A_j \|\mathbf{x}_j - \mathbf{c}_j\|^2, \tag{5.4}$$

where $\mathbf{c}_j$ denotes the vertex coordinates of $\mathcal{T}$ and A_j is the sum of the areas of all triangles of $\mathcal{T}$ sharing the vertex j.

Tightness energy

The tiles should tightly press against each other in the final configuration. The tightness energy supports a minimum normal contact force magnitude. Assuming infinite friction between tiles, we measure this structural tightness as the sum of the magnitudes of the normal contact forces smaller than f_{min} for each pair of contact tiles in the actuated mesh, $\mathcal{M}$:

$$E_{\text{tight}}(\mathbf{P}) = \sum_k \sum_{l=1}^{4} \left(\max\left(0, f_{\text{min}} - \mathbf{f}^{\text{c}}_{kl} \cdot \mathbf{n}^{\text{c}}_{kl}\right)\right)^2, \tag{5.5}$$

where $\mathbf{n}^{\text{c}}_{kl}$ and $\mathbf{f}^{\text{c}}_{kl}$ are the l-th contact normal and contact force, respectively, of the k-th edge and f_{min} is the minimal required normal force magnitude.

Fabrication and actuation limitations

These are expressed as hard constraints. Specifically, we set lower and upper bounds on the pin parameters to ensure proper actuation:

$$p_{\text{min}} \leq p^s_{it} \leq p_{\text{max}},$$

where p_{min} and p_{max} are constant. The minimal value constrains the pins to stay within the limits of the base triangle they are attached to, while the maximal value prevents the pins from being too far away from the contacts between tiles.

We also set a lower bound $A^{\text{pin}}_{\text{min}}$ on the pin areas so that they can be reliably attached to the actuating elastic membranes:

$$g^{\text{pin area}}_{si} = 2A^{\text{pin}}_{\text{min}} - \left\|(\bar{\mathbf{p}}^s_{i2} - \bar{\mathbf{p}}^s_{i0}) \wedge (\bar{\mathbf{p}}^s_{i1} - \bar{\mathbf{p}}^s_{i0})\right\| \leq 0, \tag{5.6}$$

where we use the wedge product notation $\wedge$ as a "two-dimensional analog" of the cross product.

Additionally, for each pair of tiles in contact, we introduce constraints for collisions, taking into account fabrication limits on the minimal distance between disconnected printed objects:

$$g^{\text{gap}}_{skl} = d^{\text{gap}}_{\text{min}} - \frac{(\bar{\mathbf{v}}^s_{k12} - \bar{\mathbf{v}}^s_{k11}) \wedge (\bar{\mathbf{v}}^s_{k2l} - \bar{\mathbf{v}}^s_{k11})}{\|\bar{\mathbf{v}}^s_{k12} - \bar{\mathbf{v}}^s_{k11}\|} \leq 0, \quad l \in \{1, 2\}, \tag{5.7}$$

where the indexing is shown in Fig. 5.4.

Since the maps ω_i defined by the variables $\hat{\mathbf{v}}^{\text{f}}_{it}$ and $\bar{\mathbf{v}}^{\text{f}}_{it}$ might deform the shapes of the tiles, we constrain the rigidity of the tiles as follows:

$$h^{\text{rig}}_{it} = \|\hat{\mathbf{v}}^{\text{f}}_{it} - \hat{\mathbf{v}}^{\text{f}}_{it^{+1}}\| - \|\bar{\mathbf{v}}^{\text{f}}_{it} - \bar{\mathbf{v}}^{\text{f}}_{it^{+1}}\| = 0, \tag{5.8}$$

where $t^{+1} = (t \mod 3) + 1$.

Solving the optimization problem in Eq. 5.3 requires the evaluation of the contact forces, which in turn requires solving a quasi-static problem to find the equilibrium configuration of the elastic membranes, as shown in Eq. 5.1. This operation is computationally expensive and unaffordable for any moderate-sized problem in an interactive tool. Hence, in order to efficiently solve this problem, we propose a two-step optimization approach, where we first solve a coarse approximate optimization problem and then locally refine the obtained solution.

5.3.1 Coarse optimization

We have analyzed the behavior and properties of desirable solutions and synthesized physics-based energy terms that capture them. Then, we have used these energy terms to construct an approximation model that requires no expensive operation to be evaluated but closely represents the original optimization landscape. This allows us to find an approximate solution to the original optimization problem in a cost-efficient way.

The new optimization problem is formulated as:

$$\begin{aligned} \underset{\bar{\mathbf{v}},\mathbf{p}}{\text{minimize}} \quad E_{\text{flat}} = \; & w_{\text{r}}E_{\text{r}} + w_{\text{c}}E_{\text{c}} + w_{\text{pa}}E_{\text{pa}} + w_{\text{a}}E_{\text{a}} + \\ & + w_{\text{p}}E_{\text{p}} + w_{\text{i}}E_{\text{i}} + w_{\text{s}}E_{\text{s}}, \end{aligned} \tag{5.9}$$

where each term in the cost function E_{flat} contributes to obtaining desirable properties in the actuation, fabrication, or assembly processes.

Tile rigidity

In order to minimize the deformation of individual tiles with respect to the remeshed model, we introduce a tile rigidity term, E_{r}, that measures the deformation of tile faces:

$$E_{\text{r}} = L^{-2} \sum_{i,t} \left(h^{\text{rig}}_{it}\right)^2, \tag{5.10}$$

where h^{rig}_{it} is the tile rigidity constraint function defined in Eq. 5.8 and L^{-2} is a normalizer based on the edge length L to make the energy term dimensionless.

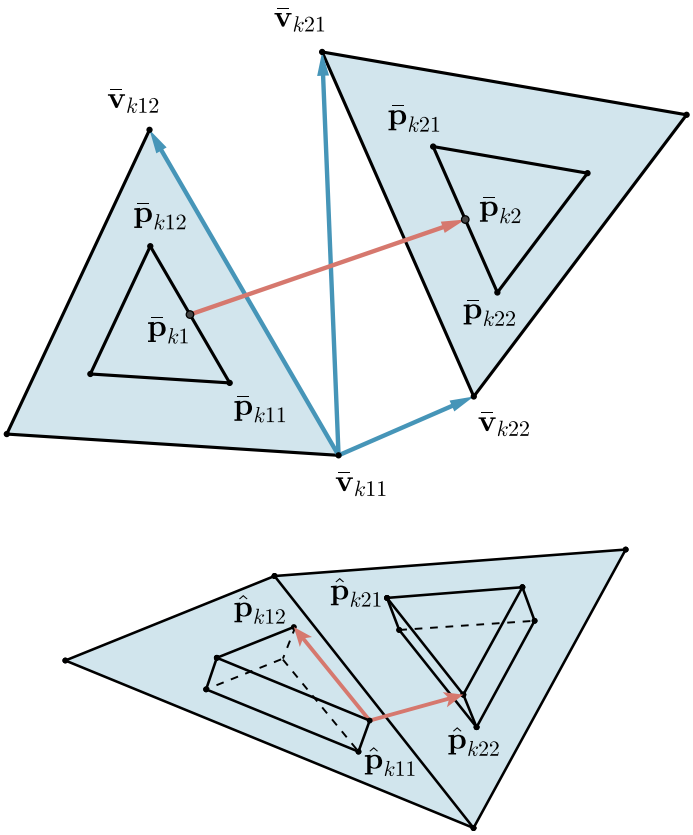


Figure 5.4: Two contact tiles in the flat (top) and actuated (bottom) configurations.

Collision avoidance

In order to avoid collisions between neighboring tiles in the flat layout, the term E_c is added:

$$E_\mathrm{c} = L^{-2} \sum_{k,s,l} \left(\max\left(0, g_{ksl}^{\mathrm{gap}}\right) \right)^2, \tag{5.11}$$

where g_{ksl}^{gap} is the inequality constraint defined in Eq. 5.7.

Pin area

We penalize pin areas below the defined minimum:

$$E_\mathrm{pa} = L^{-4} \sum_{s,i} \left(\max\left(0, g_{si}^{\mathrm{pin\,area}}\right) \right)^2, \tag{5.12}$$

where $g_{si}^{\mathrm{pin\,area}}$ is the inequality constraint defined in Eq. 5.6.

Tile alignment

We aim to place the tiles in a flat configuration such that the resulting elastic forces are aligned as much as possible with the contact surface normals. This decreases the friction contact forces, thus significantly improving the stability of the structure. As illustrated in Fig. 5.5, we introduce an alignment term, E_a, that penalizes tile misalignments:

$$E_\mathrm{a} = L^{-2} \sum_{k} \left(\|\bar{\mathbf{v}}_{k11}^{\mathrm{f}} - \bar{\mathbf{v}}_{k21}^{\mathrm{f}}\| - \|\bar{\mathbf{v}}_{k12}^{\mathrm{f}} - \bar{\mathbf{v}}_{k22}^{\mathrm{f}}\| \right)^2. \tag{5.13}$$

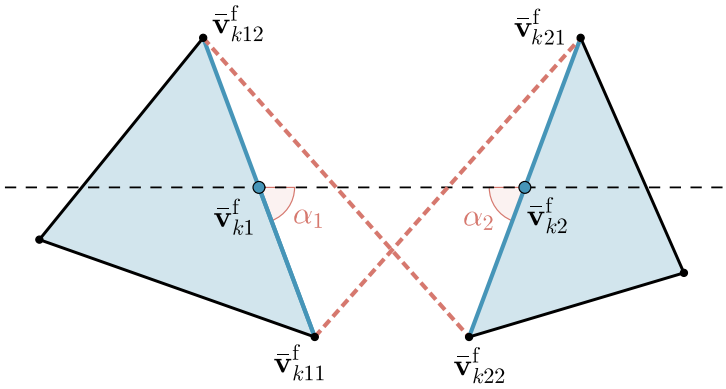


Figure 5.5: The alignment energy term aims to equalize the angles α_1 and α_2 between the edges of the tiles and the dashed line, which connects their centers $\bar{\mathbf{v}}^{\mathrm{f}}_{k1}$ and $\bar{\mathbf{v}}^{\mathrm{f}}_{k2}$. This is achieved by minimizing the difference between the lengths of the diagonals (red dashed segments).

The effectiveness of this term is illustrated in Fig. 5.6; it reduces the relative friction forces on average from 18% to 11% in the Half-sphere model.

Tile packing

In order to maximize the normal contact forces and decrease the size of the flat layout, we introduce a tile packing term, E_{p}, that helps pack the tiles in the flat layout as closely as possible:

$$E_{\mathrm{p}} = L^{-4} \sum_{k} \|\bar{\mathbf{v}}^{\mathrm{f}}_{k11} - \bar{\mathbf{v}}^{\mathrm{f}}_{k22}\|^4 + \|\bar{\mathbf{v}}^{\mathrm{f}}_{k12} - \bar{\mathbf{v}}^{\mathrm{f}}_{k21}\|^4. \tag{5.14}$$

We use the fourth order to severely penalize large distances between the contact tiles. This term also helps to reduce the computational cost of the elastic membrane simulation, which is determined by the number of degrees of freedom. For a given discretization resolution, this depends directly on the free area (not attached to any pin) in the elastic membranes. This term minimizes the free areas and the domains between tiles, therefore reducing the number of simulation degrees of freedom. As a side effect, it also allows us to prevent buckling of the membranes, which tends to produce invalid contact forces.

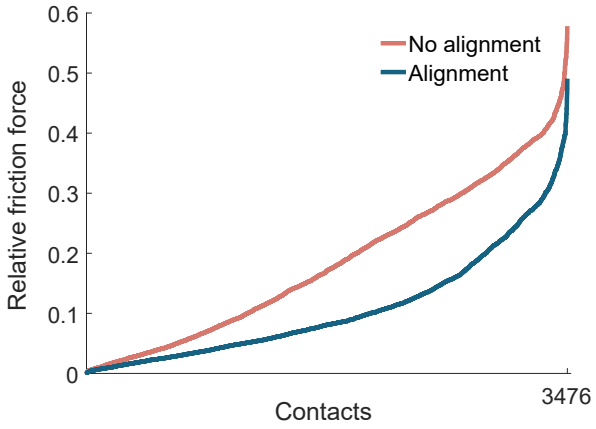


Figure 5.6: Relative friction force magnitude per contact in ascending order with and without the alignment energy term for Half-sphere.

Tile inversion

In order to prevent potential inversion of tile faces when transformed by the mappings ω_i, we introduce the following energy to penalize wrong orientation:

$$E_{\mathrm{i}} = L^{-4} \sum_i \left(\min \left(0, (\bar{\mathbf{v}}^{\mathrm{f}}_{i2} - \bar{\mathbf{v}}^{\mathrm{f}}_{i1}) \wedge (\bar{\mathbf{v}}^{\mathrm{f}}_{i3} - \bar{\mathbf{v}}^{\mathrm{f}}_{i1}) \right) \right)^2. \tag{5.15}$$

Spring-like approximation

Our goal is to find an equilibrium configuration with no pulling contact forces and elastic forces evenly distributed over the structure. This requires the evaluation of elastic forces in the actuated configuration, which is computationally expensive due to the required global physical simulation using nonlinear membrane deformation models. We have developed a spring-like approximation, which significantly reduces the computational cost and allows us to obtain the initial pin parameters that fulfill this goal.

We define the spring-like energy as:

$$E_{\mathrm{s}} = \sum_{k,s} \left(\frac{\|\bar{\mathbf{p}}^s_{k2} - \bar{\mathbf{p}}^s_{k1}\| \|\hat{\mathbf{p}}^s_{k12} - \hat{\mathbf{p}}^s_{k11}\|}{\tau \|(\hat{\mathbf{p}}^s_{k12} - \hat{\mathbf{p}}^s_{k11}) \times (\hat{\mathbf{p}}^s_{k22} - \hat{\mathbf{p}}^s_{k11})\|} \right)^2, \tag{5.16}$$

with τ being the uniform stretch ratio, $\bar{\mathbf{p}}^s_{k1} = \frac{1}{2}(\bar{\mathbf{p}}^s_{k11} + \bar{\mathbf{p}}^s_{k12})$ and $\bar{\mathbf{p}}^s_{k2} = \frac{1}{2}(\bar{\mathbf{p}}^s_{k21} + \bar{\mathbf{p}}^s_{k22})$. This energy term is the sum of squares of ratios A/B, where A is the distance between the pin centers in the flat configuration scaled by τ^{-1} and B is the distance between the pins in the actuated configuration. Minimizing such ratios in the least squares sense leads to the following observations:

- A is minimized, meaning that the rest area of the membrane between the tiles decreases and the centers of pins get closer to each other, which decreases the membrane shearing along the contact edge and thus friction;
- B is maximized, meaning that the membrane gets more stretched in the actuated configuration;
- the opposite sides of one pair of tiles in contact tend to be stretched equally due to the least squares formulation.

Note that although Eq. 5.9 includes the desired model properties as soft constraints, they can easily be turned into hard constraints by setting their weights to infinity. In practice, we exploit this feature by first solving an unconstrained optimization using finite weights for all terms, which rapidly provides an initial solution, and then refining the initial solution using hard rigidity, collision, and pin area constraints.

5.3.2 Local refinement

The designs obtained after solving the approximate model in Eq. 5.9 display the desired properties, but due to the limited precision of the approximation, invalid contacts may still be present. We could resolve them by running the full optimization problem in Eq. 5.3, but it would generate performance issues due to the computational cost of elastic membrane simulation.

Algorithm 5.1: Optimization Algorithm

Function *Coarse Optimization (***x**, ω_i*)*
  while *not Accepted* **do**
    Solve Eq. 5.9 for ω_i, **q**, only soft constraints;
    User places cuts
  end
  Solve Eq. 5.9 for ω_i, **q**, hard and soft constraints;
  Result: ω_i, **q**
Function *Local Refinement (***P**, $\mathbf{f}^e$*, c)*
  Define local domains as in Fig. 5.7;
  Solve Eq. 5.3 for local DoFs and membrane;
  Update **P** in Parameter update domain;
  Update $\mathbf{f}^e$ in Elastic update domain;
  Result: **P**, $\mathbf{f}^e$
Data: Mesh vertices **x** ; `/* Uniformly remeshed user input */`
ω_i := Init Conformal (**x**) ; `/* Initialization by conformal map */`
ω_i, **q** := Coarse Optimization (**x**, ω_i);
$\mathbf{f}^e$:= Global Elastic Forces for **P**;
$\mathbf{f}^c$:= Solve Eq. 5.1 for **P**, $\mathbf{f}^e$;
C := Set of contacts with nonzero terms of Eq. 5.5 for **x**, $\mathbf{f}^c$;
for *each* $c \in C$ **do**
  P, $\mathbf{f}^e$:= Local Refinement (**P**, $\mathbf{f}^e$, c)
end
Result: **P**

We solve this problem by exploiting the local dependence of elastic forces on model parameters. We have found that, in practice, for a given pair of tiles in contact, a change in their design parameters affects only the elastic forces on the neighboring tiles. We use this observation to run the elastic membrane simulations required in Eq. 5.3 on a smaller subdomain, significantly reducing the computational cost. It is important to note that contact forces do not show this local dependence and therefore need to be recomputed globally.

Specifically, in order to detect any possible invalid contacts, we start by evaluating the elastic and contact forces on the complete domain. Then, for every invalid contact, we define three subdomains. First, a *parameter update subdomain*, corresponding to the two tiles in contact. Second, an *elastic update subdomain*, which includes the previous subdomain and all neighboring tiles. And third, an *elastic simulation subdomain*, defined by the n-ring of surrounding tiles (including the previous subdomain), where in our examples $n = 4$. Fig. 5.7 illustrates the different subdomains.

We solve Eq. 5.3 for the design parameters associated with the *parameter update subdomain*. To update the elastic forces on pins, we simulate the elastic membranes in the *elastic simulation subdomain*, but we only update those in the *elastic update domain*, keeping the values of the elastic forces computed in the initial evaluation everywhere else. Then, we compute the contact forces by solving Eq. 5.1 on the complete domain (but using the locally updated elastic forces).

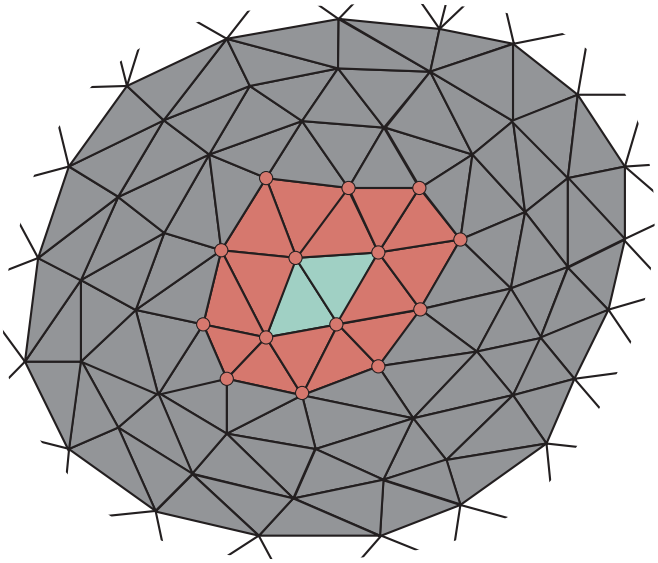


Figure 5.7: We define 3 subdomains for local refinement. First, the *parameter update domain* (green), where design parameters are updated. Second, the *elastic update domain* (red), where elastic forces acting on tile vertices are updated. And third, the *elastic simulation domain* (gray), where the quasi-static elastic problem is solved.

5.4 Implementation

In this section, we explain implementation details, including the computational design tool and the fabrication process.

5.4.1 Computational design tool

Simulation

We use the incompressible Neo-Hookean material model (see Sec. 3.3 for details), which is able to accurately capture the behavior of the elastic sheets, even for large deformations. We aim to generate a simulation mesh with minimal area to avoid unpredictable behavior of non-stretched sheet areas and minimize the computational cost. To compute the domain, we bridge each pair of actuated pins with quadrilateral regions and include all polygons enclosed by them (see Fig. 5.8 for an example). Then, we mesh the domain with triangles constraining minimal angles and maximal areas, obtaining a simulation-ready mesh.

Optimization

We solve two optimization problems: first, a global optimization, which uses the approximate model as explained in Sec. 5.3.1, and second, a series of local refinements (Sec. 5.3.2) using the accurate force model in Eq. 5.3.

In order to obtain the initial guess for the first, we build a least-squares conformal map of $\mathcal{T}$ [LPRM02] as a fast way to initialize the problem, and we scale it so that the area of each triangle fits the area of the corresponding triangle in $\mathcal{T}$. Then, we construct the tiles: the front face of each tile is placed inside the corresponding triangle of the conformal map, while the back face is defined by the front face, the tile thickness, and the dihedral angles

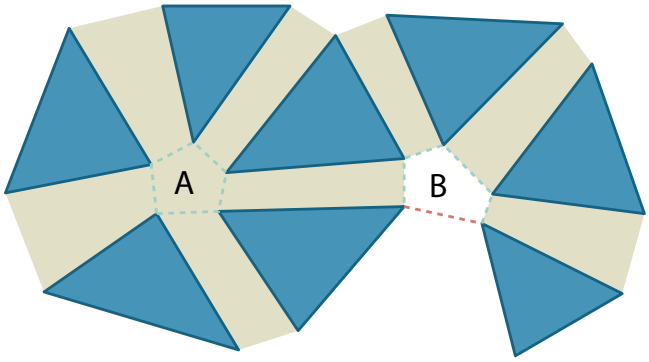


Figure 5.8: Defining the elastic sheet simulation domain based on pins (shown as blue triangles): region A is added, since it forms a full polygon, but region B is not added, since it is missing one side.

with neighboring elements. If necessary, we preprocess the input model and remesh it to obtain a regular, isotropic triangle mesh with given average edge length.

To solve the coarse optimization problems, we use the Active-set method of the `Knitro` optimization toolbox, and for the local refinements, we use the SQP method. Both methods require gradients of the cost function with respect to the design parameters. For all geometric variables, we use the algorithmic differentiation packages `CppAD` and `TOMLAB /MAD` to compute their derivatives. For the elastic forces and the contact forces, we compute analytic derivatives.

As mentioned in Sec. 5.2, we cut the tiles in order to avoid their intersections and compute the exact positions of the new contact points on the back side. Taking into account that the deviations from the target mesh are small, we avoid overloading our main optimization problem with additional internal dependencies by fixing the contact points' locations as their barycentric coordinates with respect to the corresponding front edge.

Another potential problem is the possibility of collisions between tile bases and the elastic sheet domains. We tackle this problem by choosing a sufficiently large value for the pin height.

User interaction

Not all input shapes are reproducible, and some might require modifications that impact the aesthetics of the model. To address this problem, our system provides an interactive user interface that keeps the user in the design loop. Given an input model, the system aims to compute a valid design. If no valid design can be found, the user is informed and visual feedback is provided. As shown in Fig. 5.9, problematic areas where contact forces are not within the required tolerances are highlighted, allowing the user to modify the input model accordingly with an external modeling tool or introduce cuts within our user interface. These cuts disconnect tiles that would otherwise be in contact, thus reducing the complexity of the required actuation. We display the updates performed by the optimization while the user can interactively place and refine cuts. This process allows the design space to be intuitively explored until the fabrication constraints are fulfilled and the user is satisfied with the obtained design.

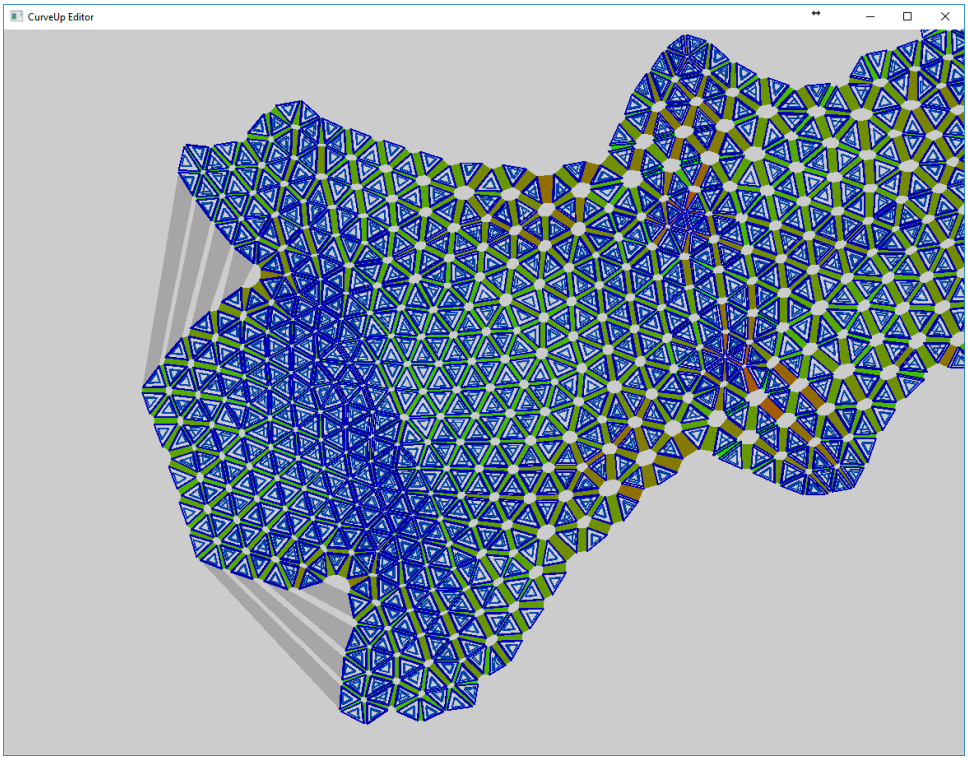


Figure 5.9: Our user interface visualizes the current configuration, highlights problematic areas, and allows for placing cuts interactively. Editing the Spot model: two cuts introduced by the user are highlighted in gray. Red colors indicate potentially problematic contacts according to the approximation.

5.4.2 Fabrication

Our fabrication process, as illustrated in Fig. 5.10, starts by 3D printing the rigid tiles and a support structure, which guarantees that the computed tile layout is kept during post-printing manipulation. We also add small friction bumps at the interfaces of neighboring tiles, which allows us to use the infinite friction assumption explained in Eq. 5.5. In our experiments, we use a Stratasys J750 3D printer and its support material, which can be easily removed using a water jet.

We use latex sheets as elastic membranes. Since the actuation depends on the relative forces on both membranes and these depend linearly on the thickness, we can safely use latex sheets of different thicknesses for each model. This is useful to minimize the risk of cracks and other defects in latex when a larger stretching factor is required. To uniformly stretch the latex sheets, we use a stretching plate with regularly distributed teeth, which produce enough friction to keep the sheet stretched without any additional fixation. In order to ensure uniform stretch and the specified stretching factor, we add markers to the latex sheet and manually stretch the latex to bring the markers to the teeth of the stretching device.

Next, a uniform layer of glue is applied over the tiles' pins on one side of the sheet, and one of the pre-stretched latex sheets is glued to them. The layout is now preserved by the latex sheet, which is kept stretched on the device, and we can safely remove the support material using a water jet. The structure is then dried, and the second pre-stretched latex sheet is glued on the other side. Finally, we detach the latex sheets from the stretching devices, obtaining the final actuated 3D shape, and cut away the latex surplus. Before

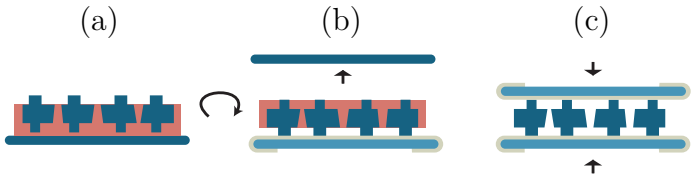


Figure 5.10: Illustration of our fabrication process. (a) 3D printout (red support, blue tiles). (b) Gluing one side to the stretched latex sheet and washing away the support. (c) Gluing the other side.

the surplus is removed, it might exert very large contraction forces at the boundary of the structure. In order to protect CurveUps from those forces potentially leading to the detaching of latex from the pins, we print a set of cylinders around the flat layout (see Fig. 1.3). The area of each cylinder is large enough to keep latex glued even when subject to very large latex surplus forces. After latex contraction, these cylinders are cut away together with the surplus.

5.5 Results

We have tested our design and fabrication approach with a broad variety of shapes, including mathematical shapes, such as the Half-sphere and a subpart of a Hyperboloid, architectural shapes, such as the Lilium Tower, artist-designed models, such as the Turtle, the Bump Cap, and the Mask, and more standard computer graphics models, such as Spot, the cow. All models were fabricated as a single patch in the flat configuration.

Tab. 5.1 shows details for each model as well as timings for the design computation and fabrication.

Half-sphere and Hyperboloid demonstrate the capabilities of our design system to produce simple models with positive and negative Gaussian curvatures, respectively. The Lilium Tower illustrates the behavior of the system when working with a more complex architectural shape. The Turtle, the Bump Cap, and the Mask are custom models that illustrate the capabilities of the design approach when used by a professional designer. Finally, Spot, is our most complex model. Its locally high curvature as well as challenging geometry for flattening, such as the horns, make it impossible to exactly reproduce as a single piece. We smooth the horns of the original model since their initial shape is too extreme for our purposes, and we manually remove the bottom part. Therefore, Spot's geometry exemplifies a use case where a non-artist is able to interact with our system, tuning an existing model and introducing cuts, to obtain a layout that can be properly actuated into the final 3D shape. Fig. 5.11 shows the input model, the optimized 2D layout, and the fabricated prototypes.

Fixed parameters

The minimum height of the tile body H_{body} and the height of the pins H_{pin} depend on the resolution of our fabrication device. For all of our fabricated objects, we have chosen $H_{\text{body}} \approx 3$ mm and $H_{\text{pin}} \approx 1$ mm, summing up to approximately 5 mm of total structure thickness. To assure a good bond between the pins and the latex sheets, we set the minimal pin area $A^{\text{pin}}_{\text{min}} \approx 4$ mm^2 and $p_{\text{max}} \approx 3$ mm. In order to leave some space for

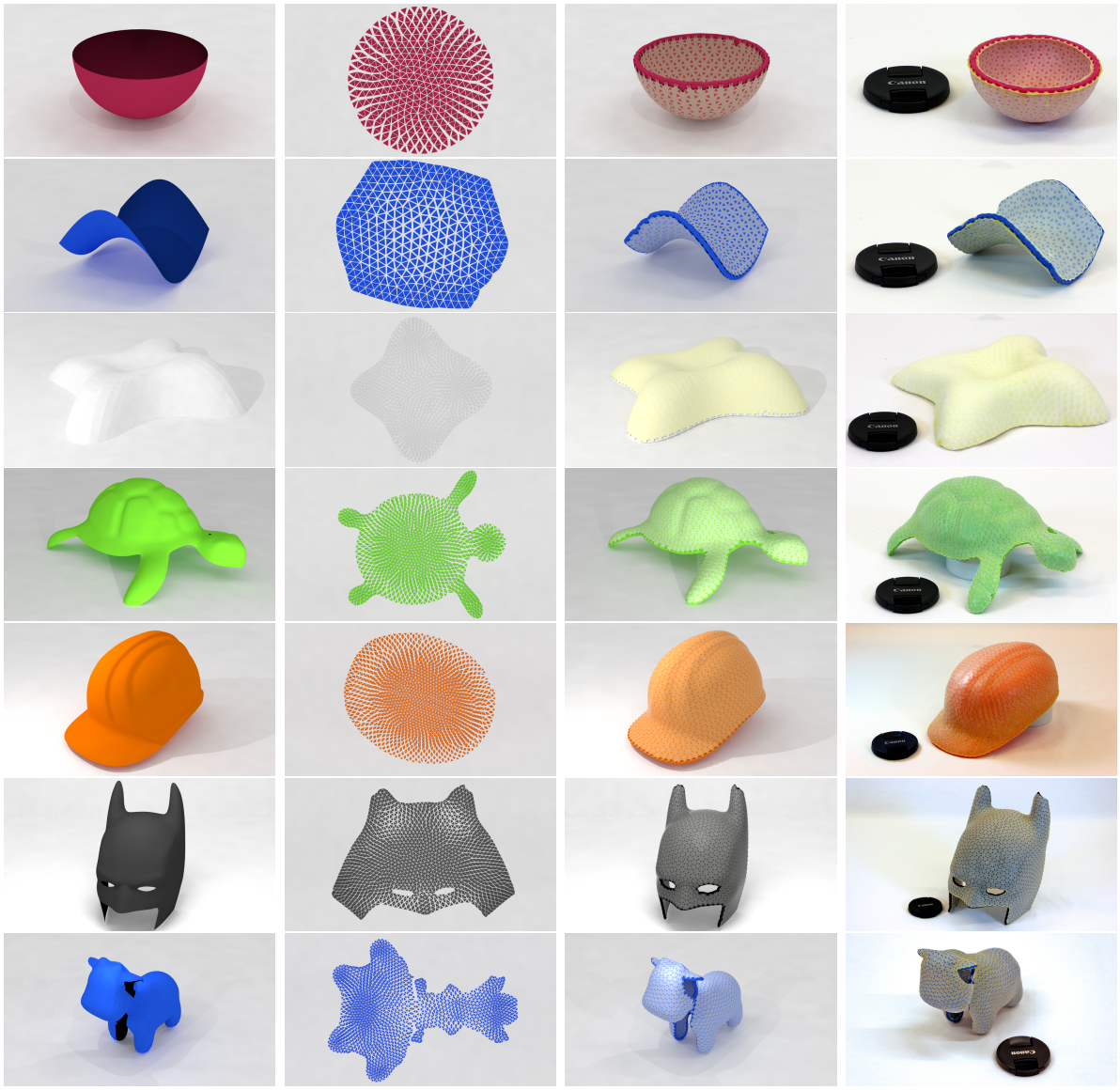


Figure 5.11: Input mesh, flat layout, actuated configuration, and fabricated result for the test models. From top to bottom: Half-sphere, Hyperboloid, Lilium, Turtle, Bump Cap, Mask, and Spot.

Model	#Tiles	$\hat{A}$	$\bar{A}/\hat{A}$	Soft	Hard	Glob.	Loc.	Fab.
H.-Sph.	594	145	1.63	40	31	59	-	71
Hyperb.	571	128	1.64	51	27	33	-	77
Lilium	1844	449	1.46	318	286	173	-	130
Turtle	2212	535	1.67	665	426	1162	1096	144
B. Cap	2531	606	1.82	466	613	1876	2623	175
Mask	2428	638	1.55	530	547	1308	-	183
Spot	2545	479	1.87	1936	557	1362	6205	149

Table 5.1: For each model, we show the number of tiles, the area in the actuated configuration (cm^2), the ratio of the approximate area in the flat configuration to area in the actuated configuration, the coarse optimization times for only soft constraints, for hard constraints, the global verification time, the local refinement time (sec), and the required 3D printing time (min).

the friction bumps between tiles in contact, we set $d_{\min}^{\text{gap}} \approx 0.6$ mm and $p_{\min} \approx 0.6$ mm. We uniformly stretch the latex sheets to 900% of the original area and therefore set the stretch ratio $\tau = 3$ for all examples.

Performance of optimization

We evaluate the performance of our optimization starting from a uniform triangulation remeshing with an average edge length $L \approx 7$ mm, and we measure the timing to obtain a valid configuration. Our simplest model, Half-sphere, consists of 594 tiles, and our most complex model, Spot, contains 2545 tiles. All computations were performed on a standard desktop computer with 3.50 Ghz and 8 cores. We present statistics for all of our models in Tab. 5.1. The coarse optimization time is divided into two parts: only soft constraints (the user interaction might also take place) and with hard constraints. We then evaluate the exact forces, which requires meshing and simulating two latex sheets and a least squares solution for the contact forces. While for some input shapes (Half-sphere, Hyperboloid, Lilium, and Mask) the coarse optimization approach already provided a valid configuration, we noticed that the other models had several invalid contacts (below 10), mostly along the boundary. For each invalid contact, we perform local refinement by running three iterations of the optimization problem (Eq. 5.3). We observed significantly larger computation times in the presence of membrane buckling. This is a well-known problem, which could be addressed by using the relaxed energy density model from Tension Field Theory [Pip86, Ste90]. The convergence of this problem in our approach cannot be guaranteed in general, since the main optimization problem is highly nonlinear and non-convex, but in practice we observed that three optimization steps usually provide a satisfying solution.

Accuracy

In order to validate the reproduction accuracy of the design and fabrication method, we have computed a quantitative evaluation for the Lilium model with the size of ∼25 cm. We have 3D scanned the fabricated model using the *David* laser scanner and computed the Hausdorff distance between the target and designed models, which equals ∼8 mm. We also visualize point-wise errors in Fig. 5.12, color-coded from blue for no error to red

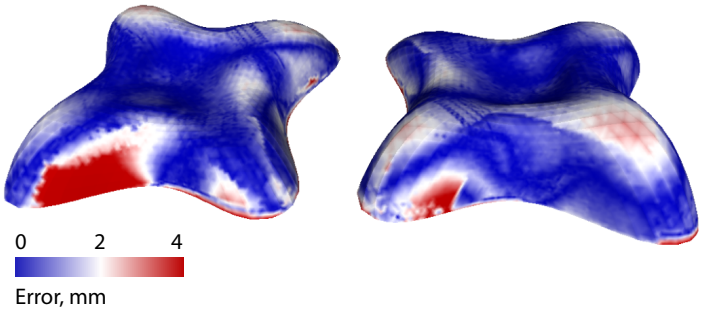


Figure 5.12: A 3D scan of the Lilium model viewed from two opposite sides.

clamped at 4 mm. The average point-wise distance is below 1 mm. We also scanned our most challenging model, Spot, which is approximately 20 cm large. The average error is below 4.3 mm. The maximal error is 12.8 mm, observed at the cow's muzzle, since the head is slightly tilted down due to a highly curved and narrow neck connecting it to the body.

5.6 Discussion

In this chapter, we have presented a new concept for designing structures we call CurveUps. They are stable, self-actuated, optionally doubly curved, smooth three-dimensional shells that form from an initially flat state. The core of our system is a computational approach for computing a physically valid layout, an optimal shape of tiles, and pins. Our method builds on an accurate physics-based model for evaluating and optimizing a design. To make this global optimization problem tractable, we have introduced an effective approximation with localized updates that allow optimizations at interactive rates. As demonstrated by our results, we successfully reproduced a variety a compelling shapes and also evaluated the accuracy of several models by 3D scanning their surfaces and comparing them to the predicted surface geometry.

While our structures are guaranteed to be in equilibrium in the computed final configuration, we cannot guarantee that the structures self-actuate from the flat to the final shape. Along the deformation path, they might have to overcome states with a minimum in deformation energy. In practice, we address this issue by slightly manipulating the object by hand. Taking the actuation path into account during the design stage would be an exciting avenue for future work.

Due to physical limitations with regard to the maximum stretchability of the elastic sheet, both the local compressibility and dihedral angle between neighboring elements are constrained. Therefore, not all shapes are realizable as a single piece. We provide a user interface that visualizes problematic regions, allows users to place cuts, and interactively updates the configuration. A potential extension would be a system that automatically suggests cuts or strategies as to how to split challenging models into multiple fabricable pieces. For future work, another interesting route would be to indicate limits on achievable shape approximations and provide this information during shape modeling.

A limitation of our model lies in the assumption of infinite friction between tiles, which turned out to be a reasonable approximation in the presence of friction bumps. For a more accurate treatment, one could measure material properties and employ a friction

model. We also neglect gravity, although this change is potentially easy to implement. In practice, these effects become more relevant as the size of the shell increases.

In the next chapter, we describe how in addition to geometric information, the deformation rates can be encoded into a self-morphing shell going from a flat to a doubly curved configuration.

CHAPTER 6

Encoding temporal information in self-morphing shells

In this chapter, we show that spatio-temporal information can be embedded in the geometry of architected shells that morph from flat to smooth three-dimensional shapes. This programmed temporal evolution enables reaching target geometries that would be impeded by collisions if shells actuated with uniform or unplanned deformation rates. We present an inverse design approach for a newly developed self-actuating mechanism based on a mechanical data-driven model. Furthermore, our shells use polymers that actuate when the temperature in their environment is set to a critical value. At room temperature, they remain flat, storing the energy necessary to drive the deformation, while after the actuation they regain their stiffness and can bear loads.

6.1 Overview

We propose an inverse design algorithm for shell architectures and the temporal evolution of their shapes (Fig. 6.1a). The algorithm collects user inputs at two stages: the first input is the desired 3D target surface, and the second is the specification of local deformation rates. We term this temporal map input an *actuation time landscape*. The algorithm outputs the mesostructure for initially flat shells that we fabricate and test. These shells have three layers, with a $\sim 4.6\,\text{mm}$ total thickness. The two outer layers are 3D-printed tessellations of non-uniform unit cells, made of Vero PureWhite (Stratasys). The middle layer is a $0.5\,\text{mm}$ thick pre-stretched elastic membrane, which stores the energy required to drive the morphing process. Actuation from the flat to the curved profile is triggered by immersing the shells into $56^\circ C$ water, which causes the outer layers to soften over the course of approximately 30 to 80 s.

The unit cells have a grid spacing of $\sim 10\,\text{mm}$, and are composed of cylindrical bases connected at the external shell surfaces by pairs of V-shaped brackets (Fig. 6.1b). The bases serve as attachment points to the elastic membrane and as mounting points for the brackets. The brackets serve as nonlinear springs: they hold the structure flat prior to being placed in water, and then they guide the temporal morphing process, softening at rates determined by their geometric parameters when heated (thicker brackets soften at slower rates). There are also bumpers attached to the bases in the space between the

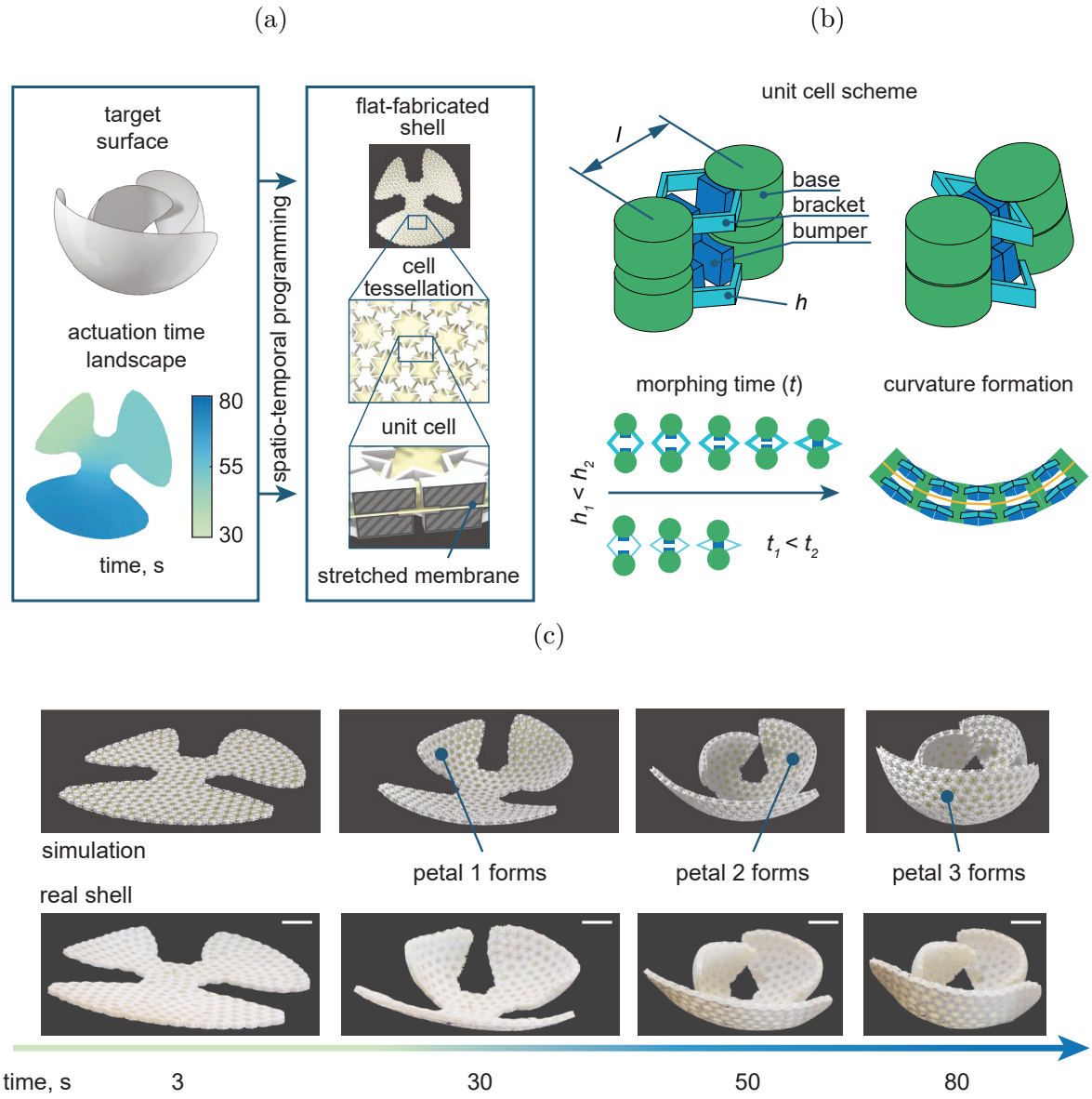


Figure 6.1: Encoding spatial and temporal shape evolution in a flat shell mesostructure. (a) A user-specified target surface and actuation time landscape (a field of deformation completion times) are inputs to an inverse design procedure that defines the mesostructure of flat-fabricated shells that morph into the target geometries. The shells are composed of inhomogeneous tessellations of unit cells with an interior pre-stretched membrane. (b) Each unit cell has an initial central length l. Brackets control actuation time through their softening rate, which is controlled by their thickness, h, and a set of bumpers prescribe final local curvatures upon collision. (c) Morphing of a petalled structure with an actuation time landscape ensuring that larger petals cover their smaller neighbors avoiding collisions on the way. Simulation and experiments are compared at 3, 30, 50, and 80 seconds in water. The structure replicates the encoded actuation time landscape shown in (a). Scale bars, 3 cm.

brackets, which collide when the local contraction reaches the target magnitude and control the contact angle between adjacent unit cells. The membrane provides energy for actuation, compressing the brackets as they soften. All of these components play an important role for reaching targeted geometries through spatio-temporal programming. For example, the

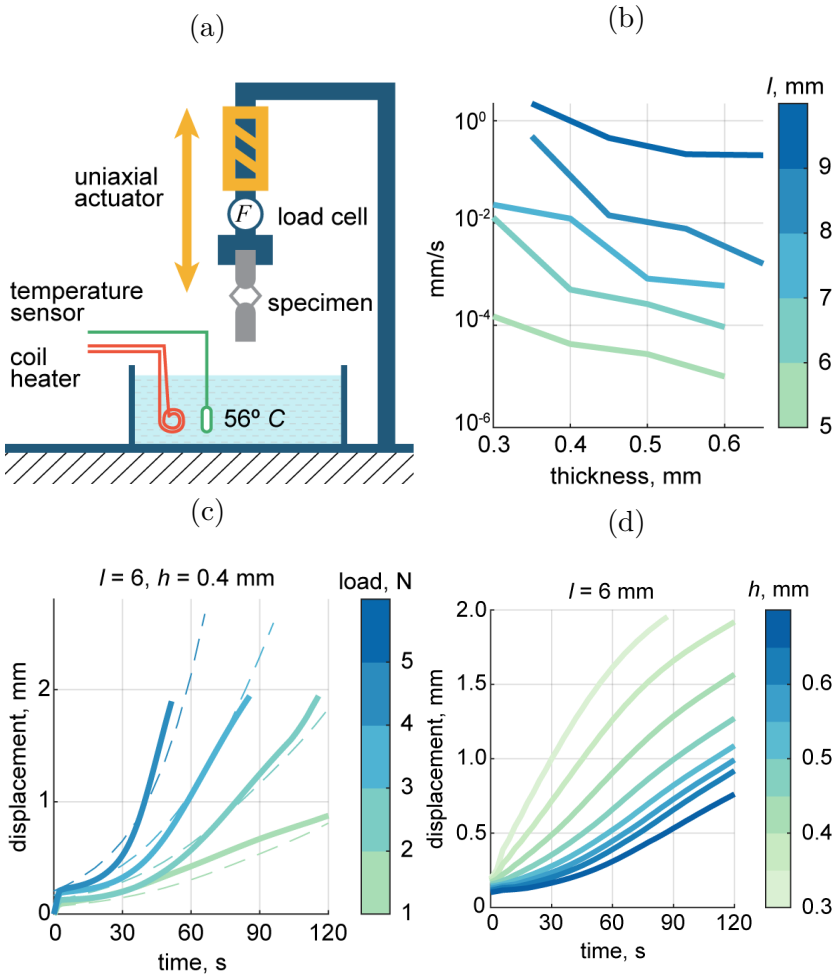


Figure 6.2: Measuring and modeling mechanical properties of brackets. (a) Load-controlled tensile tests were used to determine the deformation rates of unit cells in $56° C$ water. (b) Average deformation rates for specimens subject to constant loads of 4 N for $l < 7$ and 5 N for $l \geq 7$ N. These values are close to the inner membrane tractions on each unit cell in real shells. (c) Deformation rate measurements (solid lines) are fit (dashed lines) to produce a model of bracket softening. Here we show the fit for $l = 6$ mm, $h = 0.4$ mm. (d) The model is interpolated and queried to infer the mesostructure that yields target curvatures and deformation completion times in each section of the shell. Here, we show deformations of unit cells with central length $l = 6$ mm and a range of bracket thicknesses from 0.3 mm to 0.65 mm.

petalled structure shown in Fig. 6.1c has been programmed so that the petals reach their target shape sequentially, actuating from smallest to largest (Supplementary Movie 1). If all petals deformed at the same rate, they would collide and would not reach the target geometry. More details on shell design are presented in Sec. 6.2.

We incorporated a discretized mechanical model of our shells in an inverse design algorithm for obtaining desired temporal morphing. Given a target geometry and a smooth time landscape, the algorithm automatically generates the flat shell mesostructure that will produce the corresponding morphing process. It first composes a continuous target shell out of compressed unit cells. To do this, the target surface is isotropically triangulated, producing a stencil that serves as a placeholder for base locations. With the arrangement of bases in the target shape, the bumper geometries are defined to ensure that they are in

contact in this target shape (Fig. 6.1b top-right). Then, a minimal distortion conformal map [SSP08] flattens the stencil. The bases with bumpers are then relocated to the flat stencil and are interconnected by brackets. Note that this stencil has to be free of overlaps to enable fabrication. Therefore, base placement, bumper arrangements and bracket lengths are configured automatically given a target surface input. However, the selection of bracket thicknesses is governed by the designer's specification of the actuation time landscape. Thicker brackets soften at a slower rate than narrower ones, enabling distinct target deformation times to be realized in each region of the shell for collision avoidance, visual impression or other desired functionalities. Given that there can be a broad range of morphing sequences that yield certain target geometries, the morphing process can be designed according to the designer's goals by iterating through actuation time landscapes and observing their effect.

The time evolution of the shells is simulated quasistatically by coupling a finite element simulation of the rubber membrane with a data-driven spring model for the brackets and a rigid body model for bases. Bumper collisions are described as sharp increases in bracket stiffness in the model. A summary of the energy model is given below. Its constitutive parts are the energy associated with bracket compression (W_c) obtained from fitting and interpolating experimental data, an energy penalty to shearing (W_s) that replicates the effect of the shear-resisting bracket geometry, and the elastic membrane energy (W_m):

$$W(\mathbf{x}) = \sum_{c_{ij}} W_\mathrm{c} + \sum_{s_{ij}} W_\mathrm{s} + \sum_{\mathcal{T}_i} W_\mathrm{m}(\mathbf{G}_i). \tag{6.1}$$

Here, c_{ij} refers to the contractile springs that join the i-th and the j-th bases. Each unit cell is modeled with four of these springs to capture bending effects. Bumper collisions are modeled as a sharp stiffening of these elements. Shear-resisting elements s_{ij} have analogous indexing. $\mathcal{T}_i$ refers to the i-th element of the membrane discretization, and $\mathbf{G}_i$ is the deformation gradient of the membrane evaluated at this element. A complete description of the simulation procedure is presented in Sec. 6.3.

To construct the constitutive model for bracket softening, we conducted experiments (Fig. 6.2a) on brackets of varying length l (in a range 5–9 mm) and thickness h (0.3–

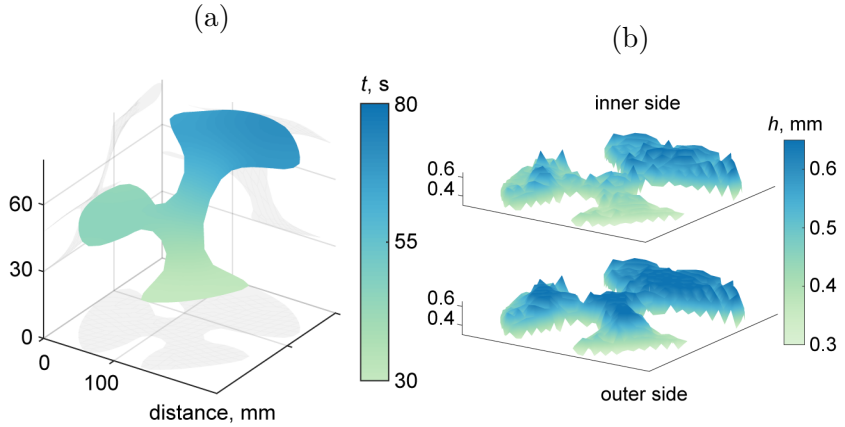


Figure 6.3: Inverse design of temporal morphing. (a) Smooth actuation time landscape that induces the sequential deformation process demonstrated in Fig. 6.1. (b) Bracket thickness fields for both sides of the petalled shape. Though the prescribed time landscape is smooth, the field of bracket thickness is highly irregular because bracket thicknesses also depend on initial unit cell lengths and their target deformations.

0.65 mm), applying constant forces (1-5 N) and tracking their compression over time spent in water using a Zwick tensile tester (Figs. 6.2b and 6.2c). Fits to the experimental data were then sampled from the space of bracket parameters and immersion times to build a time-dependent force-displacement model used in the simulation (Fig. 6.2d). Material measurement and modeling are discussed more extensively in Sec. 6.4. From this sampling, we select bracket thicknesses that yield target deformation timings under the loads generated by the membrane (see Sec. 6.5). We show that the relationship between the actuation time landscape and the resulting bracket thicknesses in nontrivial in Fig. 6.3.

6.2 Shell design

Our shells have three layers. Two 3D-printed non-uniform tessellations form the outer layers. These are glued to either side of a pre-stretched latex membrane. This section describes the design of the flat-printed geometries.

6.2.1 Single unit cell design

The shells' outer layers are tessellations of the structures shown in Fig. 6.4. Each of these unit cells is bounded by four cylinders of diameter 4 mm and height 2.3 mm, called bases. The bases are connected to their neighbors by two symmetric V-shaped spring elements (called brackets) that form an angle of 37° relative to the central axis and have a height of 1 mm. At room temperature, brackets are sufficiently stiff to prevent finite deformations due to compression by the pre-stretched elastic membrane that constitutes the shells' mid-planes. The brackets soften when placed in hot water, inducing in-plane contraction. This contraction occurs until the bumpers attached to the bases in the space between the brackets collide. This collision occurs in each unit cell once the target in-plane deformation for that unit is reached. Local curvatures are programmed in a unit cell by setting different bumper lengths for the opposite outer layers. To facilitate aligning the two outer layers with respect to each other during shell fabrication, cylindrical holes are subtracted from several bases (see Sec. 6.6 for more information).

The feature dimensions were chosen for the following reasons. Bases interface the membrane to the brackets, so they must have a sufficiently large gluing area to be reliably connected to the membrane, but should be small enough to allow large curvatures in decimeter-scaled specimens. The bracket shape is designed to reduce both in-plane and out-of-plane shearing. In-plane shearing is prevented due to the large (37°) angle between the bracket and the central axis. Out-of-plane shearing is prevented due to the rectangular shape of the bracket section. The 1 mm bracket height is always larger than its thickness ($h \leq 0.65\,\text{mm}$), which makes in-plane bending energetically favorable.

6.2.2 Tessellating unit cells

The design pipeline starts with a user-provided target surface (Fig. 6.5), which is isotropically remeshed into a triangular mesh $\mathcal{T}$ with a target number of vertices N [LWL$^+$09]. Each pair of adjacent triangles of this mesh represents one unit cell with two bases placed on opposite faces of each triangle, centered on its barycenter. Thus, the whole mesh $\mathcal{T}$ serves as a stencil for our structure's final state, when deformation has completed. It is scaled to ensure there is a minimal bumper length (0.3 mm) to limit bracket deformations in the final state. The actual size of the shell depends on the number of vertices in $\mathcal{T}$.

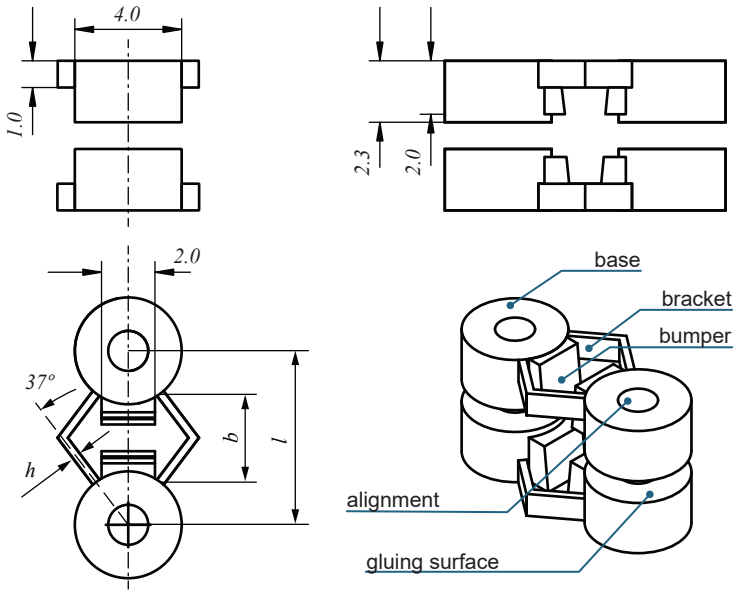


Figure 6.4: Unit cell scheme. The configurable parameters are the central length l (constant difference with bracket length b), bracket thicknesses h (which can be different for the two opposite layers), and the bumper cutting plane.

In order to generate a shell of dimensions close to the input surface, we aim to minimize the required scaling of the stencil. First, we find the best fitting number of vertices of the stencil. It can be coarsely estimated as $N \approx 0.38A/l_{\text{avg}}^2$ where A is total surface area and $l_{\text{avg}} = 7\,\text{mm}$ is the average unit cell length (using heuristic knowledge that the number of triangles is twice the number of vertices, assuming triangles are close to regular, and approximating unit cell lengths as twice the radius of circumscribed circles). Then, N can be varied to find a value leading to minimal stencil scaling.

Given the layout of bases on the stencil, the bumpers are first constructed as boxes that are aligned along the lines connecting the centers of the triangles and projected onto the corresponding stencil triangle. Matching pairs of boxes are trimmed by the bisector plane between the triangles, defining the interface between neighboring bumpers.

The flat arrangement of bases is then constructed using a minimal distortion conformal map [SSP08] from $\mathcal{T}$ to a resulting 2D mesh, $\mathcal{F}$. We exploit conformal flattening since it circumvents shearing which would result in undesired shear forces in the membrane as it contracts. The mesh $\mathcal{F}$ serves as a stencil for the structure's initial, printed state. We relocate the bases by translation from triangle centers in the final state's stencil to triangle centers in the initial state's stencil. The bases are rotated to align one of its bumper axes in the direction of the corresponding neighbor's center since it is generally not possible to perfectly align all of them. Then, $\mathcal{F}$ is scaled so that the relocated bases with bumpers do not intersect with their neighbors, moreover specifying the minimal gap between them necessary for fabrication (0.1 mm).

Once the flat layout of bases with bumpers is complete, V-shaped brackets that bridge the gaps between bases are generated. This produces a flat structure that fully encodes the target geometry. Conformal flattening may produce overlaps of unit cells, or some of the unit cells may be longer than our upper bound (9.5 mm), making the target shape impossible to replicate. Large unit cell lengths may occur if the target surface has regions with high curvatures or that require substantial stretching during conformal

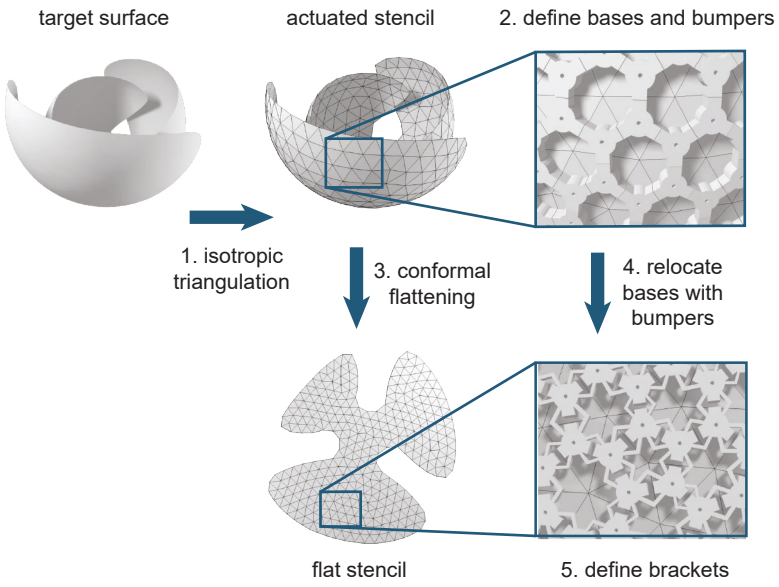


Figure 6.5: Shell design pipeline: 1. A target surface is isotropically triangulated. 2. This "actuated" stencil is populated with bases and bumpers touching their corresponding neighbors. 3. The "actuated" stencil is conformally flattened. 4. Bases with bumpers are relocated to the flat stencil. 5. Bracket lengths are set by the distance between bases in this configuration. Bracket thickness is defined later during the temporal programming phase.

flattening. It is in certain cases possible to resolve the latter issue by placing cuts across the surface [GMB17] or by editing the input geometry. Morphing times are programmed by configuring the thicknesses of the brackets (see Sec. 6.5).

OpenSCAD[1] scripts were used to generate STL meshes for the fabricated structures in this study.

6.3 Simulation

Our framework builds on physical simulation to predict the deformed configuration of the system at a given instant in time. We consider a discrete mechanical model that determines the mechanical behavior of a deformable object based on an elastoplastic model with elastic energy potential $W(\mathbf{x}, \mathbf{X}(t), \boldsymbol{\kappa}(t)) \in \mathcal{R}$ and plastic energy dissipation formulated through the rest configuration update. Here, $\mathbf{x}(t) \in \mathcal{R}^n$ is a vector containing n generalized coordinates that spatially discretize the kinematic state of the shell in different configurations, $\mathbf{X}(t)$ refers to the undeformed configuration, $\boldsymbol{\kappa}(t)$ is a vector grouping all material stiffness parameters, and t is the time instant.

Temporal effects are modeled through the explicit dependencies of the undeformed configuration and material stiffness on time. In the next section, we describe how these dependencies are estimated from empirical data. Because our structures morph at low strain rates, we neglect all dynamic effects and take discrete time increments of constant duration, $\delta = 0.5\,\mathrm{s}$. After each time increment, the undeformed configuration and stiffness parameters are updated quasistatically. We denote the magnitudes corresponding to the

[1] http://www.openscad.org

k-th time step of the morphing process as $\mathbf{x}^k$, $\mathbf{X}^k$, $\boldsymbol{\kappa}^k$. For notation simplicity, we will drop the superscript corresponding to the morphing time step unless specified. Simulating the static behavior of this mechanical system at the k-th morphing time step implies solving the nonlinear system of differential equations defined by net force equilibrium, i.e., $\mathbf{F}(\mathbf{x}^k, \mathbf{X}^k, \boldsymbol{\kappa}^k) = -\nabla_{\mathbf{x}} W = \mathbf{0}$, using standard numerical optimization methods.

Our computational model couples a FEM simulation of the membrane, a rigid body model for bases, and a data-driven spring model for the brackets. In the following sections, we describe the kinematics and mechanics of each of these subsystems separately and then specify how we model the coupling between them and solve the numerical problem.

6.3.1 Discrete kinematics

The bases are modeled using N_r prismatic rigid bodies. Each rigid body represents two bases attached to the membrane from the opposite sides. The kinematic state of the i-th rigid body can be determined by the position of the center-of-mass $\mathbf{v}_i \in \mathcal{R}^3$, together with its rotation $\mathbf{r}_i \in \mathcal{R}^3$, expressed in angle-axis format. The corresponding rotation matrix $\mathbf{R}_i$ can be easily computed using the Rodrigues' rotation formula

$$\mathbf{R}(\mathbf{r}) = \mathbf{I} + \sin(\theta)[\mathbf{u}]_\times + (1 - \cos(\theta))[\mathbf{u}]_\times^2, \tag{6.2}$$

where $\theta = ||\mathbf{r}||$, $\mathbf{u} = \mathbf{r}/||\mathbf{r}||$ and $[\mathbf{u}]_\times$ is the cross product matrix of $\mathbf{u}$, i.e., the matrix such that $[\mathbf{u}]_\times \mathbf{x} = \mathbf{u} \times \mathbf{x}$. This allows us to express the position of any point $\mathbf{p}_j$ in the local coordinates of the i-th rigid body through the non-linear relation $\mathbf{p}_j = \mathbf{R}_i(\mathbf{r}_i)\mathbf{p}_{ji}^0 + \mathbf{v}_i^k$, where $\mathbf{p}_{ji}^0 = \mathbf{p}_j^0 - \mathbf{v}_i^0$, are the coordinates of the point in the local frame of the rigid body.

Each pair of neighboring bases are joined by two brackets. The N_s brackets are modeled using two types of components: data-driven springs and shear-resisting elements. Both components are composed by line segments denoted $\mathbf{s}_{ij} = (\mathbf{s}_i, \mathbf{s}_j)$, where $\mathbf{s}_i$ and $\mathbf{s}_j$ are a pair of points on the surface of the i-th and j-th bases.

- Data-driven springs (Fig. 6.6, left), $\mathbf{c}_{ij}^q$, for $q = 1, \ldots, 4$, are responsible for modeling the time-evolving resistance to deformation as well as bumper collisions.
- Shear-resisting elements (Fig. 6.6, center), represented by crossing pairs of segments $\mathbf{s}_{ij}^q = \{\mathbf{s}_{ij}^{qa}, \mathbf{s}_{ij}^{qb}\}$, for $q = 1, \ldots, 4$, are responsible for penalizing undesired in-plane and out-of-plane shearing during the simulation. Resistance to shearing is inherent to the fabricated brackets due to their V-shaped design.

Finally, we represent the elastic membrane as a piecewise linear mesh of triangles (Fig. 6.6, right), with N_m vertices. The set of membrane vertices $\mathcal{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_{N_m}\}$ can be partitioned into two subsets: free vertices, $\mathcal{M}_\mathrm{f} = \{\mathbf{f}_1, \ldots, \mathbf{f}_{N_\mathrm{f}}\}$, and vertices coupled to the bases, $\mathcal{M}_\mathrm{g} = \{\mathbf{g}_1, \ldots, \mathbf{g}_{N_\mathrm{g}}\}$.

All points lying on the surface of a base are coupled to it implicitly. The positions of these points can be expressed in terms of the center-of-mass and rotation of the rigid bodies following the non-linear expression in Eq. 6.2. Therefore, the geometric configuration of the shell can be completely determined by the positions and orientations of the rigid bodies together with the membrane vertices that are not coupled to the rigid bodies. This leads to a total of $N_\mathrm{t} = 6N_\mathrm{r} + 3N_\mathrm{f}$ degrees of freedom which we group in the state vector $\mathbf{x} = \{\mathbf{v}_1, \mathbf{r}_1, \ldots, \mathbf{v}_{N_\mathrm{r}}, \mathbf{r}_{N_\mathrm{r}}, \mathbf{f}_1, \ldots, \mathbf{f}_{N_\mathrm{f}}\}$.

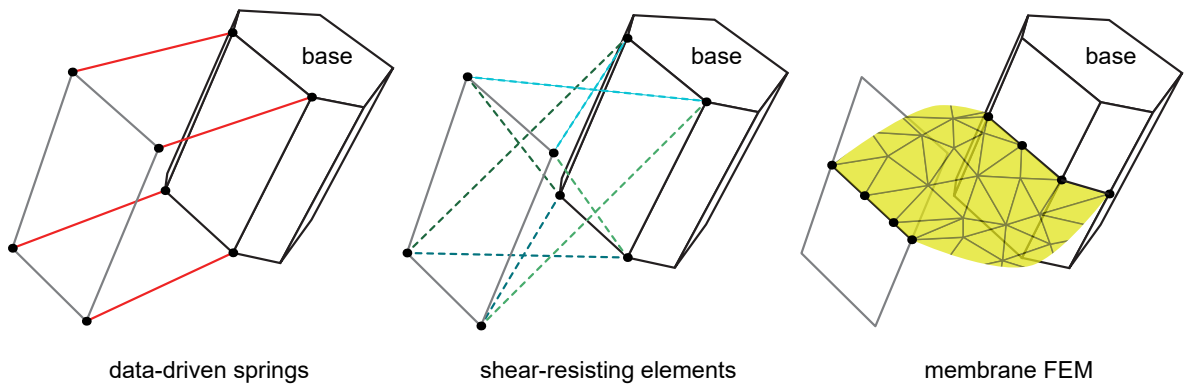


Figure 6.6: Discretization elements: data-driven springs, representing brackets' time-evolving stiffness and bumper collisions (left); shear-resisting elements, representing brackets' resistance to undesired shearing (center); and membrane FEM (right).

6.3.2 Discrete energies

Given this discretization, the mechanical behavior of the shell can be described by a conservative elastic potential W aggregating the contributions of the data-driven springs W_c, shear-resisting energy W_s, and the membrane W_m:

$$W(\mathbf{x}) = \sum_{U_{ij}} \left(\sum_{q=1}^{4} W_\mathrm{c}(\mathbf{c}_{ij}^q) + \sum_{q=1}^{4} W_\mathrm{s}(\mathbf{s}_{ij}^{qa}, \mathbf{s}_{ij}^{qb}) \right) + \sum_{\mathcal{T}_i} W_\mathrm{m}\Big(\mathbf{G}_i(\mathbf{m}_{\mathcal{T}_i})\Big). \tag{6.3}$$

here, U_{ij} refers to the unit cell joining the i-th and the j-th bases, $\mathcal{T}_i$ refers to the i-th element of the membrane discretization, and $\mathbf{G}_i$ is the deformation gradient of the membrane evaluated at this element. Let us separately explain each of the energy terms:

- The data-driven spring energy W_c, has the following expression:

$$W_\mathrm{c}(\mathbf{c}_{ij}^q) = \begin{cases} W_\mathrm{d}\Big(b, h, t, \bar{L} - L(\mathbf{c}_{ij}^q)\Big) & \text{if} \quad L(\mathbf{c}_{ij}^q) > L_\mathrm{c}, \\ \frac{\kappa_\mathrm{c}}{2}\Big(L(\mathbf{c}_{ij}^q) - L_\mathrm{c}\Big)^2 & \text{if} \quad L(\mathbf{c}_{ij}^q) \leq L_\mathrm{c}, \end{cases} \tag{6.4}$$

  where L_c is the collision distance for the spring determined by the bumpers geometry. Initially, the elastoplastic behavior of the spring follows a data-driven model $W_\mathrm{d}(b, h, t, x)$ for a given bracket length, b, bracket thickness, h, and time spent under water, t (see next section). It depends on spring deformation, $x = \bar{L} - L(\mathbf{s}_{ij}^q)$, where $\bar{L}$ is the rest length of the spring. To account for plastic effects, the rest length of the spring is updated after each time increment following the scheme $\bar{L}^{k+1} = \min\Big(\bar{L}^k, \bar{L}^0 - \eta(\bar{L}^0 - L^k)\Big)$ with constant plasticity fraction η. Once the current length of the spring is smaller than the collision distance, the actuation is stopped and the collision distance is enforced using a soft constraint defined through a high stiffness constant, κ_c.

- The shear-resisting energy W_s, has the following expression:

$$W_\mathrm{s}\Big(\mathbf{s}_{ij}^{qa}, \mathbf{s}_{ij}^{qb}\Big) = \frac{\kappa_s}{2}\Big(L(\mathbf{s}_{ij}^{qa}) - L(\mathbf{s}_{ij}^{qb}) - R^0\Big)^2, \tag{6.5}$$

where $L(\mathbf{s}_{ij}) = \|\mathbf{s}_i - \mathbf{s}_j\|$, is the distance between the spring segment end points, R^0 is the difference between distances in the initial morphing time step, and κ_s is a constant.

- For the membrane energy W_m, we use a classical FEM formulation with an incompressible Neo-Hookean material [Ogd97]. Continuum magnitudes are interpolated from nodal values using linear basis functions which allows us to discretely approximate the deformation gradient $\mathbf{G} = \nabla_{\bar{\mathbf{m}}}\mathbf{m}$. Here, the undeformed configuration can be computed from the membrane state at the initial configuration $\bar{\mathbf{m}} = \tau^{-1}\mathbf{m}^0$, where τ is the pre-stretch factor.

6.3.3 Coupling and solver

At each time step of the morphing process, we formulate and solve the nonlinear system of differential equations defined by the net force equilibrium $-\nabla_\mathbf{x} W = \mathbf{0}$, by minimizing the discrete elastic potential in Eq. 6.3. We solve this problem using Newton-Raphson method with Strong Wolfe convergence conditions for step length selection.

Solving this problem efficiently requires analytically computing both the first $\nabla_\mathbf{x} W$ and second $\nabla^2_{\mathbf{xx}} W$ derivatives of the elastic potential. As introduced above, points lying on the surface of the bases, $\mathbf{p}$, are implicitly coupled to the rigid bodies through Eq. 6.2. Hence, these derivatives can be easily computed using the chain-rule:

$$\frac{\partial W}{\partial \mathbf{x}} = \frac{\partial W}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \frac{\partial W}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \mathbf{x}}, \qquad \frac{\partial^2 W}{\partial \mathbf{x}^2} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^T \frac{\partial^2 W}{\partial \mathbf{f}^2}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} + \frac{\partial \mathbf{p}}{\partial \mathbf{x}}^T \frac{\partial^2 W}{\partial \mathbf{p}^2}\frac{\partial \mathbf{p}}{\partial \mathbf{x}}\frac{\partial W}{\partial \mathbf{p}^2}\frac{\partial^2 \mathbf{p}}{\partial \mathbf{x}^2}, \tag{6.6}$$

where $\nabla_\mathbf{x}\mathbf{f}$ is a selection matrix of the free membrane vertices and $\nabla_\mathbf{x}\mathbf{p}$ and $\nabla^2_{\mathbf{xx}}\mathbf{p}$ can be computed from Eq. 6.2. One common technique in rigid-body simulation to simplify this computation is to keep rotational degrees of freedom $\mathbf{r}_i$ close to zero. This is done by updating the local coordinates of the coupled points in the base frame after each successful iteration, i.e., for the j-th point attached to the i-th rigid body, $\mathbf{p}^0_{ji} \leftarrow \mathbf{R}(\mathbf{r}_i)\mathbf{p}^0_{ji}$, $\mathbf{r}_i \leftarrow \mathbf{0}$. We provide the following data as an example of the simulation scales. Our most complex model with self-interweaving shape (Fig. 6.15d) contains 549 rigid bodies and 7942 membrane elements. The full morphing process simulation (240 time increments) takes 43 minutes in total.

6.4 Material measurement and modeling

As discussed in Sec. 6.3, we represent brackets in simulations by data-driven springs and shear-resisting elements. Here we describe our approach to the mechanical modeling of the data-driven components. The brackets in our structure undergo large deformations and are made of a material with nonlinear elastic properties and time-dependent softening. This combination of material and geometric nonlinearities leads us to a data-driven effective spring model. We performed all measurements in settings that resemble conditions brackets are subjected to in an assembled structure. We first formulate the data-driven elastoplastic spring model $W_\mathrm{d}(b, h, t, x)$ (introduced in the previous section) and then describe our fitting strategy. We discuss several polynomial fittings in this section and display their output units in square brackets.

6.4.1 Physical model of brackets

Our elastoplastic bracket model is motivated by a set of material tests described below. The elastic component is expressed through an effective stiffness, and the plastic behavior is described as a dissipation of internal elastic energy due to deformation. We make the approximation of assuming a constant plasticity fraction, η, which we obtained experimentally. The plastic part of the displacement is then given by $x_{\text{pl}} = \eta x$ and the elastic part is $x_{\text{el}} = (1 - \eta)x$.

We aim to obtain elastic energy formulations $W_{\text{d}}(b, h, t, x_{\text{el}})$ that are functions of time t spent in hot water for each valid combination of bracket length b and thickness h. These formulations are modelled as trilinear interpolations between polynomials $p_{b,h,t}(x_{\text{el}})$ that are defined on a regular grid.

Under the assumption of a constant plasticity fraction and with material properties corresponding to time t, we relate elastic energy $W_{\text{d}}(b, h, t, x_{\text{el}})$ to the total external work $\bar{W}$ done through monotonic bracket displacements x in the following manner:

$$W_{\text{d}}(b, h, t, x_{\text{el}}) = (1 - \eta)\bar{W}\Big(b, h, t, \frac{x_{\text{el}}}{1 - \eta}\Big). \tag{6.7}$$

Here, $\bar{W}$ are trilinear interpolations of polynomials $p^{\text{work}}_{b,h,t}(x)$ [Nm] that are similarly related to $p_{b,h,t}(x_{\text{el}})$:

$$p_{b,h,t}(x_{\text{el}}) = (1 - \eta)p^{\text{work}}_{b,h,t}\Big(\frac{x_{\text{el}}}{1 - \eta}\Big). \tag{6.8}$$

These polynomials are obtained by integrating force over displacement:

$$p^{\text{work}}_{b,h,t}(x) = \int_0^x p^{\text{load}}_{b,h,t}(\tilde{x})d\tilde{x}\,, \tag{6.9}$$

where $p^{\text{load}}_{b,h,t}(x)$ [N] are polynomials representing the loads exerted on bracket over applied displacements. The polynomials $p^{\text{load}}_{b,h,t}(x)$ are fourth-order in x with no free term. The fitting methods for obtaining these polynomials are discussed in the following subsection.

It is challenging to measure time-dependent force-displacement relationships directly in an experimental setup since specimen submersion and loading takes a significant amount of time relative to our actuation time ranges. Specimen submersion and loading takes 8 seconds while the target deformations last approximately 30 to 80 seconds. This restrains us from an assumption that material properties are "fixed" at time t for the material measurements. Additionally, we collected a higher density of data in time rather than in displacement due to the capabilities of our experimental setup. This leads us to dividing the fitting problem into simpler components. We first fit (inverted) displacement-force relationships and then use the obtained model to reconstruct the desired force-displacement model $p^{\text{load}}_{b,h,t}(x)$. Our displacement-force model is the following:

$$x(b, h, t, F) = p^{\text{dry}}(b, h, F) + \int_0^t F \exp\Big(p^{\text{wet}}(b, h, \tilde{t}, F)\Big)d\tilde{t}, \tag{6.10}$$

where p^{dry} [m] is a third-order polynomial (limited to first order in b and h) for which all terms have F as a multiplier and p^{wet} [$\log\left(\mathsf{mN^{-1}s^{-1}}\right)$] is a fourth-order polynomial. Polynomial p^{dry} represents displacement-force relationships before putting into water while p^{wet} is the logarithmic evolution of deformation rates, divided by load, in water. The latter formulation restricts the deformation speed to be always positive which is a

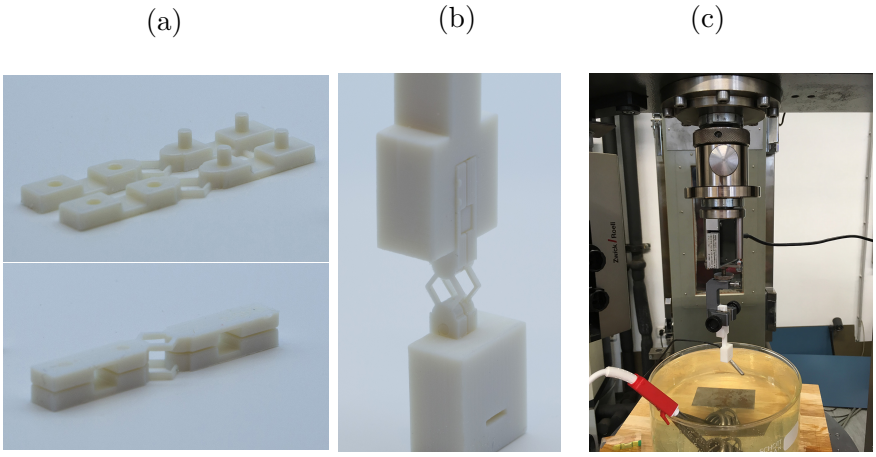


Figure 6.7: (a) Specimens used for material measurements are assembled from two printed parts to mimic a unit cell. Assembled specimens have holes to ensure consistent boundary conditions in a gripper that was fabricated in-house. (b) Custom-built gripper for quick specimen exchange and a "boot" for firm specimen compression against the floor. (c) Zwick tensile tester for measuring bracket deformations in hot water.

desirable property for our model and assures zero displacement under zero load. The integral does not generally have a simple explicit representation.

We apply the transformation mentioned above to the measured deformation speeds and fit the resulting data

$$p^{\text{wet}}(b, h, t, F) = \log\left(F^{-1}\frac{\partial x}{\partial t}\right). \tag{6.11}$$

The logarithmic function improves the quality of fitting by reducing extreme variation in deformation rates. Note that the formulation of displacement-force relationships is defined as a single continuous function across all parameters in our setup. This allows us to build a consistent model of data collected from all experimental measurements.

6.4.2 Data collection and fitting

A Zwick tensile tester (shown in Fig. 6.7c) was used for all bracket characterization experiments. The specimens tested (shown in Fig. 6.7a) were attached to a pair of prismatic grippers (Fig. 6.7b) to enforce uniaxial movement.

Three types of experiments were conducted: compression tests in both dry and wet states, as well as effective plasticity measurements. For the dry compression tests, displacement-force relations for brackets that hadn't been immersed in water were obtained quasistatically. For the wet compression tests, we immersed specimens into hot water and immediately applied constant loads. These specimens deformed gradually over time since exposure to hot water causes them to soften. For the effective plasticity tests, we compressed brackets by a prescribed displacement, unloaded them, and measured the restoration to derive the plastic component of the deformation. For all tests we used the same sampling of central lengths l (from Fig. 6.4, it is easy to see that $l = b + 4\cos 37°$) and bracket thicknesses h (in millimeters): $l \in \{5, 6, 7, 8, 9\}$, for $l \leq 7$ we choose $h \in \{0.3, 0.4, 0.5, 0.6\}$ and for $j > 7$ we choose $h \in \{0.35, 0.45, 0.55, 0.65\}$. This set of parameters amounts to 20 total combinations.

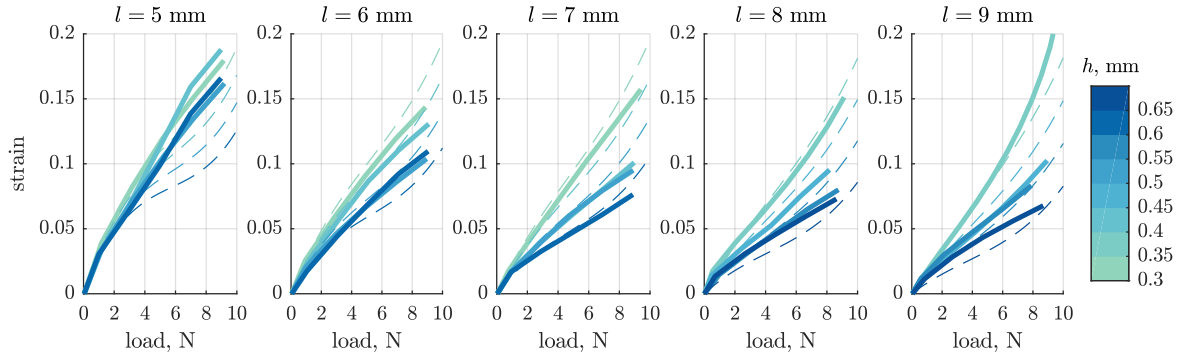


Figure 6.8: Compressive loading of dry specimens. Data (solid lines) and fitted curves (dashed lines).

Fitting polynomials with constrained derivatives

Empirical knowledge (such as the fact that thicker brackets deform at slower rates) was used to derive constraints on fitted functions. Similar to standard polynomial regression, we solve a quadratic programming problem $y = X\beta + \epsilon$, but instead of $\beta = (X^T X)^{-1} X^T y$ we solve:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\beta^T X^T X \beta - y^T X \beta \\ &\text{subject to} && A\beta \leq 0\,, \end{aligned} \tag{6.12}$$

where A expresses row-wise derivative constraints per point. By providing sufficiently many points in A we significantly improve fitted curve quality. We impose these constraints at all given data points. All the constraints used for each polynomial fitting are listed below.

Compressive loading of dry specimens

This experiment is conducted once for each specimen by increasing the applied load quasi-statically from 0 to 10 N. We fit to this data a polynomial p^{dry} expressing resulting displacements x given the loading F. Note that displacement is a monotonic function of bracket length b, unlike strain. We impose the following derivative constraints:

$$\frac{\partial p^{\text{dry}}}{\partial b} > 0, \quad \frac{\partial p^{\text{dry}}}{\partial h} < 0, \quad \frac{\partial p^{\text{dry}}}{\partial F} > 0. \tag{6.13}$$

The resulting curves are in Fig. 6.8.

Compressive loading of specimens in water

The next step is to model the evolution of the displacement-force curves over time spent in water. In order to do so, we test the behavior of each specimen under various loads (which are held constant throughout an individual test).

Since the initial force-displacement relationships p^{dry} are known with relatively high precision, we use those curves as our model for $t = 0$. We fit the displacement rate data to reconstruct the time-evolution of bracket behavior in water. We impose the following constraints on the derivatives:

$$\frac{\partial p^{\text{wet}}}{\partial b} > 0, \quad \frac{\partial p^{\text{wet}}}{\partial h} < 0, \quad \frac{\partial p^{\text{wet}}}{\partial t} > 0, \quad \text{and} \quad \frac{\partial p^{\text{wet}}}{\partial F} > 0. \tag{6.14}$$

Then we reconstruct time-evolving displacement-force relationships in Eq. 6.10. The resulting curves are shown in Fig. 6.9. We filter out displacements larger than $^2/_3$ of the initial length since brackets may be damaged and display inconsistent behavior at that point.

After obtaining a displacement-force relationship, we invert it with respect to displacement. We densely resample our target parameter space of b, h, t and refit force-displacement curves represented by polynomials $p^{\text{load}}_{b,h,t}(x)$ with constrained derivatives:

$$\frac{\partial p^{\text{load}}_{b,h,t}(x)}{\partial x} > 0 \quad \text{and} \quad \frac{\partial^2 p^{\text{load}}_{b,h,t}(x)}{\partial x^2} < 0. \tag{6.15}$$

Effective plasticity tests

In order to properly simulate actuation of the structure, we examine elastic energy dissipation during bracket compression. We approximate this phenomenon through an elasto-plastic model. We do not study the dependence of plasticity on temperature because all experiments occur at 56 ^oC. Thus, we analyze the dependence of plasticity on deformation rates. This experiment is done for specimens with fixed chosen parameters $l = 8\,\text{mm}$ and $h = 0.45\,\text{mm}$. We do multiple loading tests with different deformation rates and measure the restored strain after unloading. We observed that there is no significant dependence on deformation rates (see Fig. 6.10). We did not observe consistent strain restoration across different bracket thicknesses and lengths, with mean plastic fractions in a range 15%–25%. We use a constant mean plasticity fraction $\eta = 20\%$ in our simulations.

6.5 Temporal programming

In our design pipeline, the user specifies a time landscape. This is a smooth scalar field over the target surface that represents the desired deformation completion time at each point. In our implementation we define it as a piecewise linear function on top of the flat stencil triangulation. For each unit cell, we compute the average desired actuation time, t^*, using values specified at the ends of the associated stencil edge. Then we use t^* to configure the thicknesses of the brackets associated with the unit cell. This computation is based on the effective force-displacement curves modelled by trilinear interpolation of polynomials $p^{\text{load}}_{b,h,t}(x)$, which we denote as $F(b, h, t, x)$. Each unit cell's deformation is consistent with its neighbors' so long as the time landscape is sufficiently smooth.

Both pairs of brackets on the opposite sides of the unit cell have the same initial length b_0 by construction. We first compute their length in the fully actuated state b_1 (in general, different for different sides). Then, for the target actuation time t^*, we can formulate the bracket thickness h configuration problem as follows:

$$F(b_0, h, t^*, b_0 - b_1) = F^{\text{mem}}(b_0, d_1), \tag{6.16}$$

where F^{mem} is an approximation of the traction generated by the membrane dependent on the initial unit cell length, b_0, and on a parameter describing membrane deformation, d_1 with its value at the actuated state. Here we use the force balance between the brackets and the membrane at the target deformation after time t^* spent in water, ignoring plasticity (Fig. 6.11b).

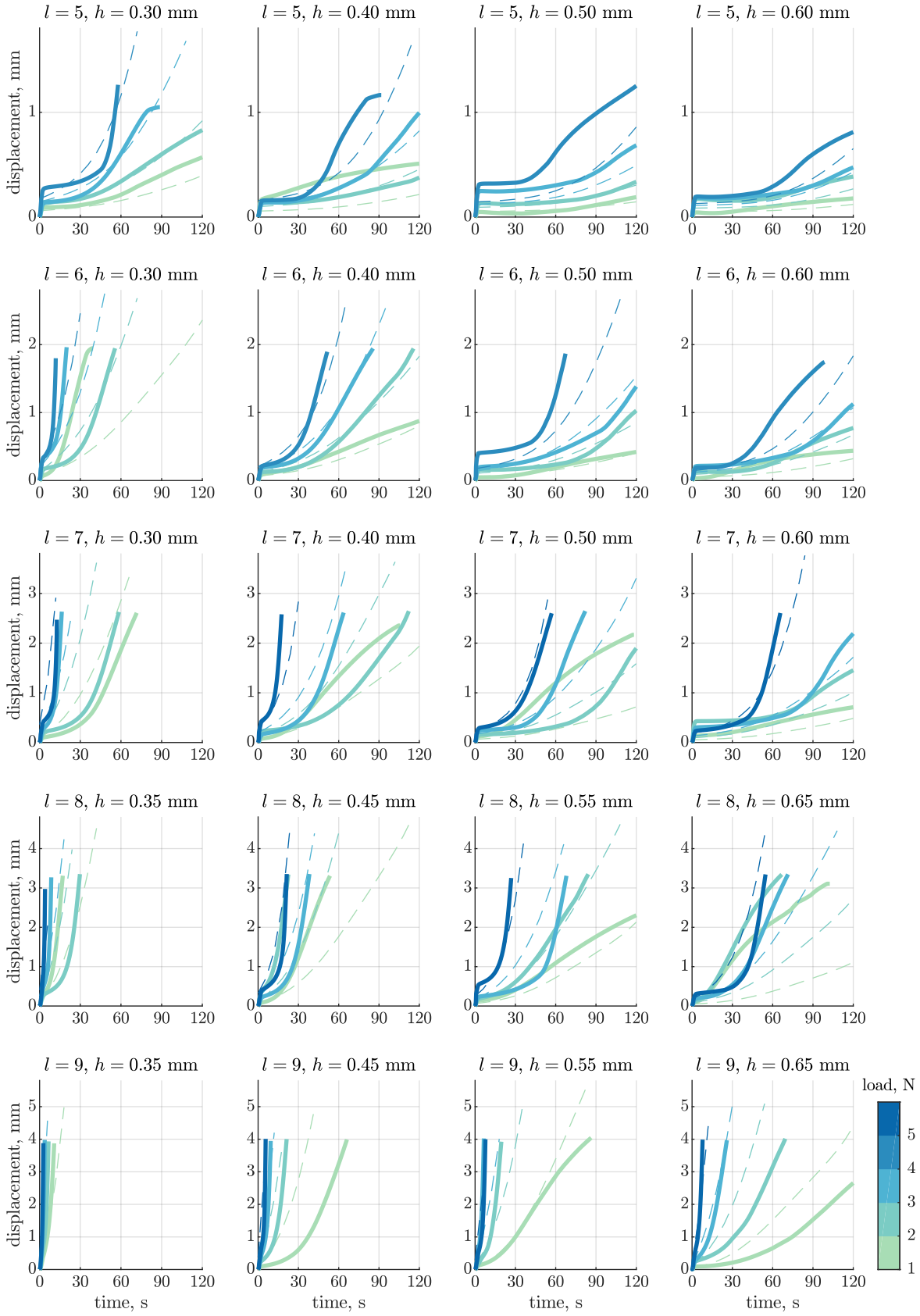


Figure 6.9: Compressive loading of specimens in water. Data (solid lines) and fitted curves (dashed lines).

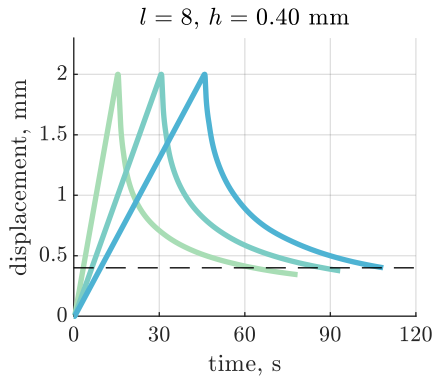


Figure 6.10: Plasticity does not depend on deformation rates. Three different deformation rates are shown for a unit cell specimen of length $l = 8\,\text{mm}$ and thickness $h = 0.4\,\text{mm}$. Dashed line represents 20% of maximal deformation which we use as a constant plasticity fraction in our simulations.

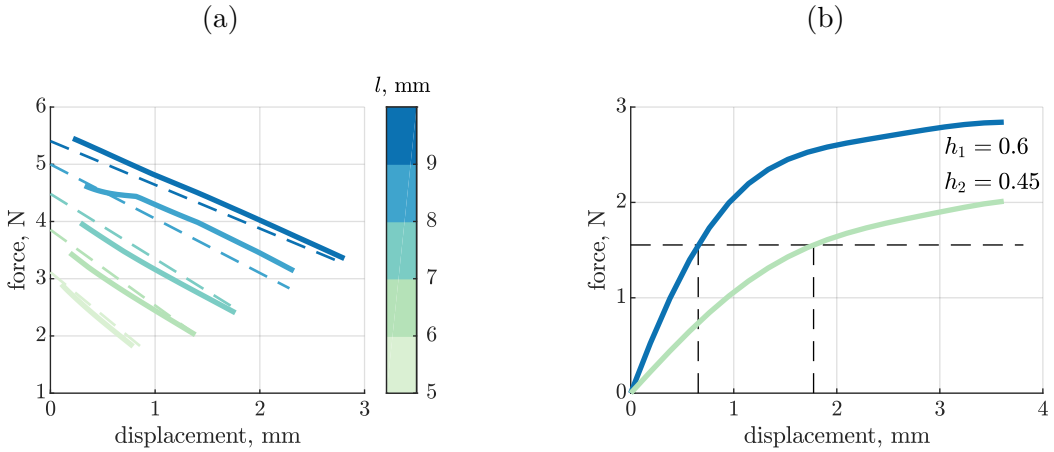


Figure 6.11: (a) Linearized model of the membrane (dashed lines) in comparison to FEM membrane (solid lines) for a set of unit cells of various initial lengths. The membrane tractions decrease with displacement as the pre-stretch is relaxed. (b) Configuring thicknesses for two pairs of brackets on opposite sides of a unit cell. Note that the one requiring larger target displacement is thinner to finish deformation at the same time as the one with smaller target deformation. The dashed horizontal line shows a sample approximation to the target membrane traction.

We build a linearized model of the membrane, representing it as a segment spring connecting the centroids of the end points of all brackets in a unit cell, and setting $F^{\text{mem}}(b, d)$ as a first-order polynomial in each of the variables. Here b mimics an initial membrane spring length and d is its displacement. We fit this model by sampling initial bracket lengths and deformations. It captures membrane forces well in our setting (Fig. 6.11a).

Our approach to finding the thickness h is a binary search through our interpolated model of $F(b_0, h, t^*, b_0 - b_1)$ to match a known value of $F^{\text{mem}}(b, d)$.

We need a specific treatment of boundary unit cells since the membrane ends there and has less stretch initially. To account for this, we reduce the force evaluation by a factor of 0.6 which was empirically found by comparing simulations of our examples with the membrane represented by linear springs to FEM membrane simulations.

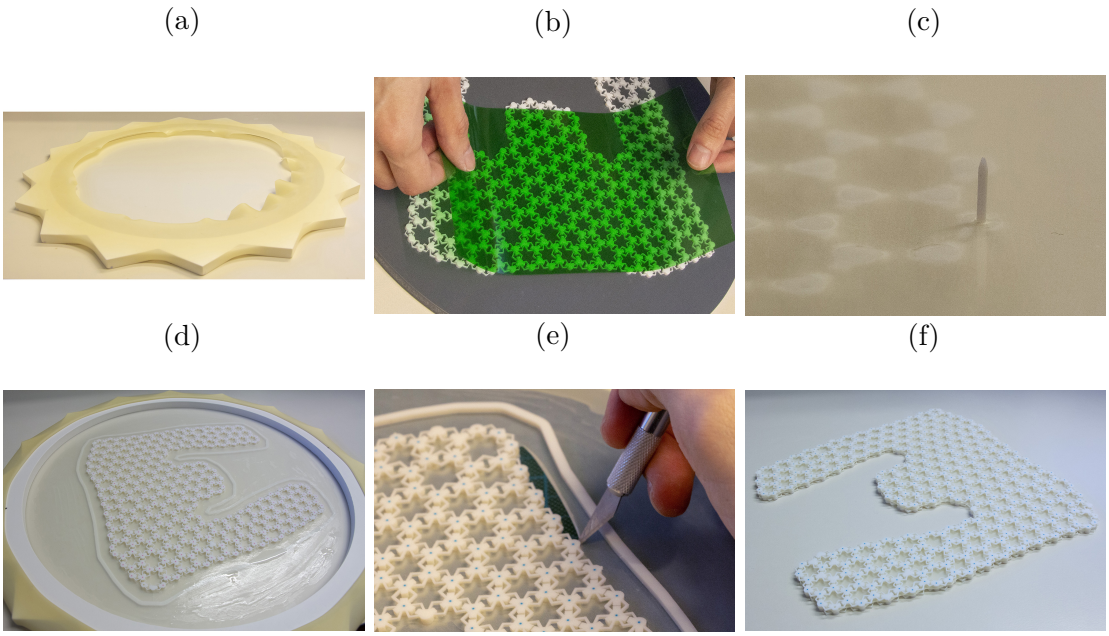


Figure 6.12: Fabrication process landmarks. (a) Star-shaped membrane stretching device back side up. Bottom part of the membrane is uniformly stretched due to markers. (b) Transferring glue from a plastic foil to the bases of the shell. (c) Passing a pin through one of the bases and the membrane to align with the second lattice. (d) Membrane surplus is covered by glue in order to "freeze" it and enable its easy removal. (e) Cutting out the shell from the membrane surplus by a scalpel. (f) Flat-fabricated shell ready for actuation in water.

Once the desired time landscape is specified and bracket thicknesses are computed accordingly, we start the simulation process described in Sec. 6.3. Unit cells' actuation completion might deviate from the specified time landscape due to imperfections in the linearized membrane force estimation and plastic effects. Apart from that, the resulting morphing process might not reach the final goal, for example there might be collisions on the way. In these cases time landscape has to be edited and the simulation of morphing has to be recomputed until the goal is achieved.

6.6 Fabrication procedure

We developed a custom fabrication procedure for the shells. It is described in detail in this section.

1. 3D print the outer layers of the shells using a Stratasys J750 printer. We also printed an outer frame to reduce undesired deformation of the structure during fabrication. Printing time for all of our shells is approximately 2 hours (it mostly depends on the number of layers, which is fixed). It is twice as long for the first petalled shape (Fig. 6.1c) since both sides do not fit on a single printing tray. We remove all overhangs, which does not affect shape replication but makes the cleaning process drastically easier: we only quickly remove almost all support material by several shaving movements with a sharp scraper and airflow the lattice to erase the rest.

2. Use a star-shaped device to manually stretch a latex sheet uniformly by gradually wrapping it around and fixing at the device's teeth (Fig. 6.12a). Friction between the membrane and the device is sufficient to keep the membrane stretched without any additional clamps. A stretch factor of 3 (900% the area) is enforced by matching markers on the sheet to the tips of the device's teeth. Note that large deformations of the membrane lead to slight stress relaxation in the long run. Given the variability of the rest of the system and our simplified membrane model, we neglect this effect. It is however possible to take it into account since after approximately 1.5 hours stress relaxation does not progress much further (Fig. 6.13).

3. Clean both surfaces of the sheet with 2-Propanol for better gluing. Apply super glue on a plastic film and distribute it uniformly with a brush. Transfer glue from the film to the 3D printed structures through contact with the plastic film (Fig. 6.12b).

4. Glue one lattice to the membrane and pass push pins through the alignment holes on the lattice and through the latex membrane (Fig. 6.12c). The membrane does not rip since we pinch it through an isolated area.

5. Glue the opposite side to the latex sheet matching the holes with corresponding alignment pins.

6. Distribute additional super glue on the latex membrane in the region surrounding the lattice perimeter (Fig. 6.12d).

7. Wait 5 minutes and then release the latex sheet to a stretch factor near 2. Then using a scalpel cut out the structure, keeping a tiny amount of degenerated latex at the border to prevent membrane ripping (Fig. 6.12e).

8. Submerge the structure in 56° C water and wait for morphing to complete.

9. Take out the deformed shape and let it dry. Under normal conditions the drying process may take roughly 15 minutes.

The whole fabrication process after 3D printing and before drying takes 30–50 minutes.

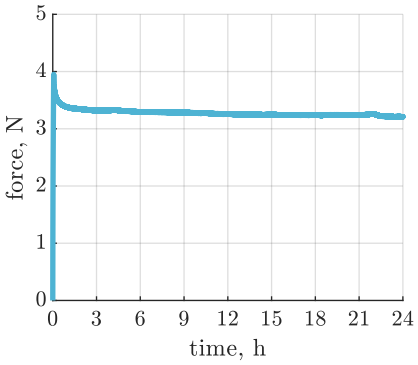


Figure 6.13: Membrane stress relaxation over the course of 24 hours. Evolution of the force generated by a dog-bone membrane specimen under a constant stretch factor of 3.

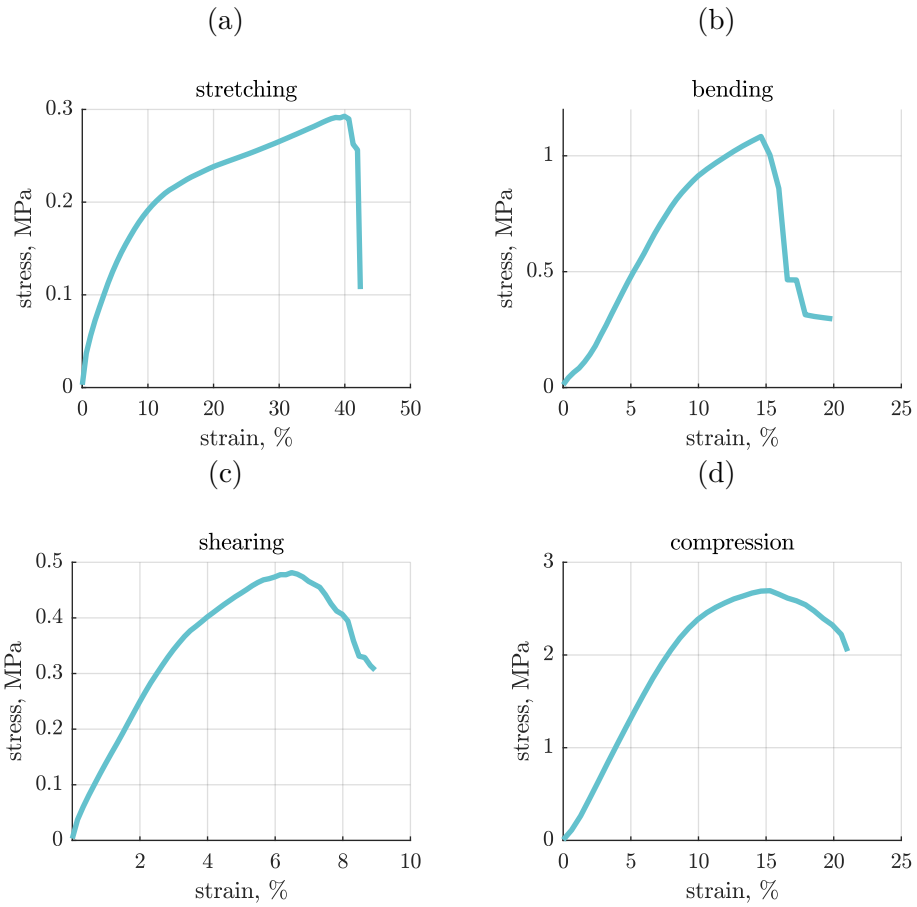


Figure 6.14: Mechanical tests of a flat regularly tessellated shell. Since our shells have cross-sections with a complex geometry, we provide the effective stress values (assuming shell homogeneity). (a) Stretching, (b) bending, (c) shearing, and (d) compression tests.

6.7 Mechanical measurements of shells

We performed a set of mechanical measurements of the shells for estimation of their load-bearing capabilities. We fabricated rectangular and flat regularly tessellated shells composed of identical unit cells of length 7 mm, bracket thickness 0.5 mm, and bumpers 0.3 mm with total dimensions 60×55 mm. Since the shell structure is not isotropic, we performed the tests aligning the deformation to one of the three axes parallel to the edges of the hexagonal pattern. The resulting plots are shown in Fig. 6.14. In all tests the deformation speed was approximately 1 mm/min. It is intuitive that the shells have higher resistance to pure in-plane compression due to the bumper contacts in contrast to the other cases when only the membrane and the brackets are loaded.

6.8 Examples of temporally programmed structures

We highlight the effect that different actuation time landscapes have on the final shapes of initially flat shells by comparing the example discussed in Fig. 6.1c to the shell shown in Fig. 6.15a. Both shells have similar flat geometries but different actuation time landscapes are encoded in their mesostructures. In the first example, smaller petals are covered by their larger neighbors. Meanwhile, each petal shown in the second example has an edge that covers a neighbor and one edge that is covered. For the latter case, all the petals have

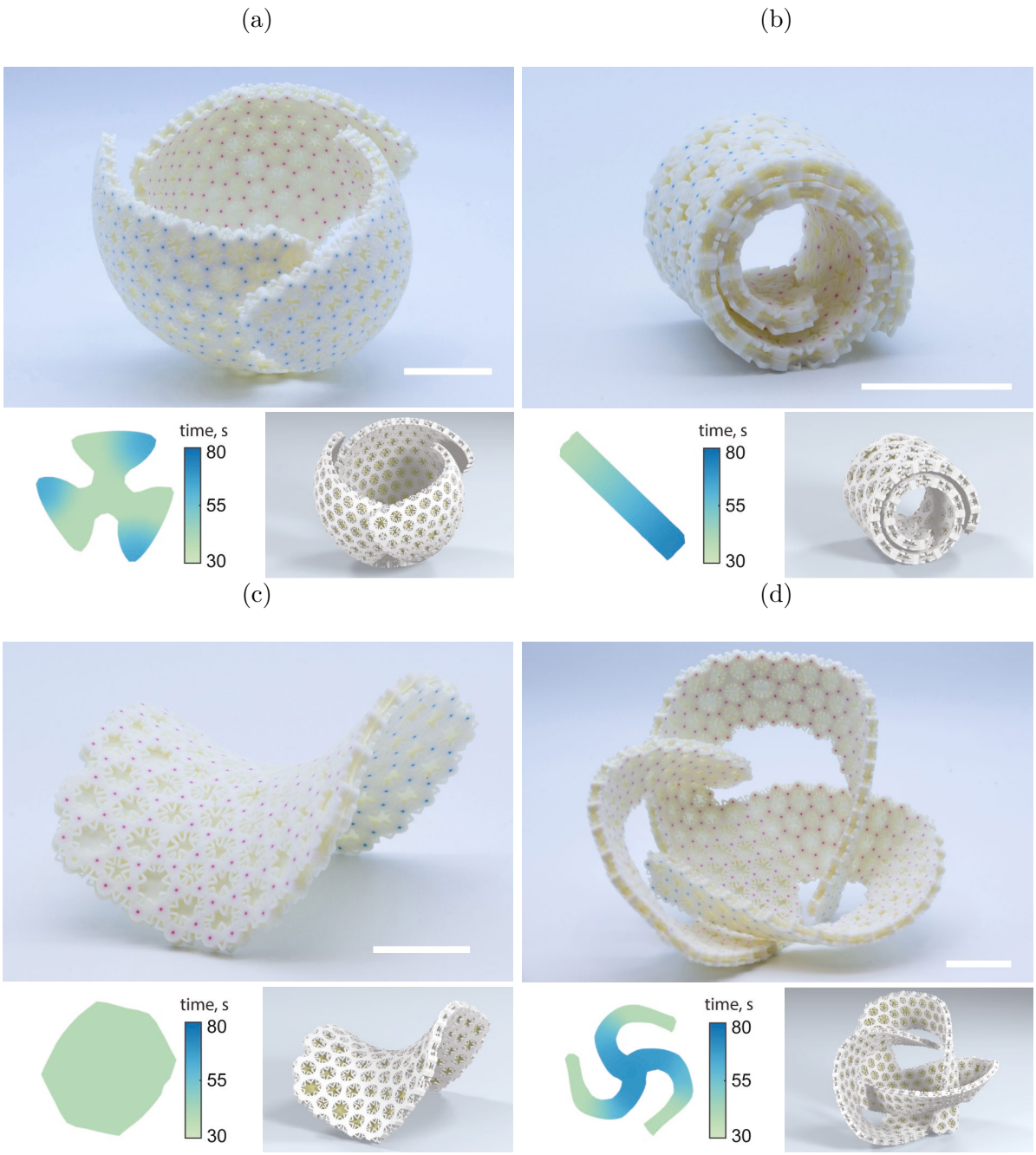


Figure 6.15: Spatio-temporally programmed shells. Each panel shows a real shell (top), its actuation time landscape (bottom-left), and its corresponding simulated shell (bottom-right). (a) Doubly curved shell where petals morph synchronously to cover each other in a cyclic manner. One corner of each petal is programmed to morph slower to increase the distance between petals during morphing. (b) A double spiral that approximates a developable surface. A gradient time landscape enables the inner spiral to curl first. (c) A saddle shape with negative curvature. (d) A shell with a complex self-interweaving shape prone to multiple collisions in the course of its morphing process. Scale bars, 3 cm.

been programmed with the same actuation time landscape so they deform simultaneously. We slightly increased target actuation times for some petal tips to increase the distance between neighboring petals on their morphing paths (Supplementary Movie 2). This way, the interior edge of each petal completes its deformation before being covered by its

neighbor. Shape-morphing precision was measured using a 3D scan of the final geometry. The blue markers were matched with their simulated locations. The resulting mean error for the pairwise distances in the experimental realization of the structure shown in Fig. 6.15a was 3.6% relative to the diameter of the target markers' point cloud.

Developable target surfaces can also be achieved using this structural framework. The actuation time landscape for the double-loop spiral shown in Fig. 6.15b is a constant gradient from one end to the other. This allows the interior to curl without colliding with the outer loop (Supplementary Movie 3). The 3D scan reveals a 2.4% mean error. Geometries with negative Gaussian curvature can also be realized, such as the saddle shown in Fig. 6.15c (Supplementary Movie 4). The relative mean error for the saddle's base positions was 0.8%.

We showcase the complexity of achievable shapes through the self-interweaving structure shown in Fig. 6.15d. This shape requires embedding precise morphing trajectories and time landscapes in the flat-printed structure in order to thread each arm through the loop created by a neighboring one without colliding (Supplementary Movie 5). The experimental replication of this challenging target geometry yields a 9.7% 3D scan mean error, and highlights the versatility of our design and simulation framework.

6.9 Discussion

The realization of complex 3D geometries from flat-fabricated structures, which are easier to manufacture and transport, is a compelling motivation for developing shape-morphing frameworks. However, to be used in a broad range of engineering applications, the morphed structures must remain structurally stable. In our examples, the shells' outer layers soften during deformation in hot water, but become stiff when returned to room temperature. We show their load-bearing ability realizing a mobile phone stand (Fig. 6.16a). Note that the mass of the phone (113 g) exceeds the mass of the shell (55 g). Data from simple mechanical tests on flat structures is provided in Sec. 6.7. Structural stability could be further increased by using snap-locking mechanisms instead of bumpers or coating the structure after actuation has completed.

Temporally programmed self-morphing opens the door to a broad range of industrial design applications. Sequentially interlocking joints, for example, can be employed in the self-assembly of initially flat furniture (Fig. 6.16b), while doubly curved surfaces can be used in drone shells (Fig. 6.16c). The Schwarz P triply periodic minimal surface structure (Fig. 6.16d) can be obtained from layered sheets that are programmed to have time-evolving porosity. Such structure could be used as an energy absorbing material or a microfluidic device with programmable channel cross-sections (Fig. 6.16e).

Our method for programming temporal morphing in shells is based on a combination of a pre-stressed mid-plane and outer layers with effective stiffness differentials that are configured to evolve over time according to user specifications. The encodement is performed using an inverse design algorithm that takes a target surface and an actuation time landscape as inputs and outputs a mesostructure with embedded self-morphing information. The significance of this method is that it enables collision-avoidance during deformations from flat shapes to curved geometries. We built a design system based on a data-driven mechanical model of mesostructures to predict shape evolution in time, enabling temporal morphing design. Applications of self-actuating shells to biomedical

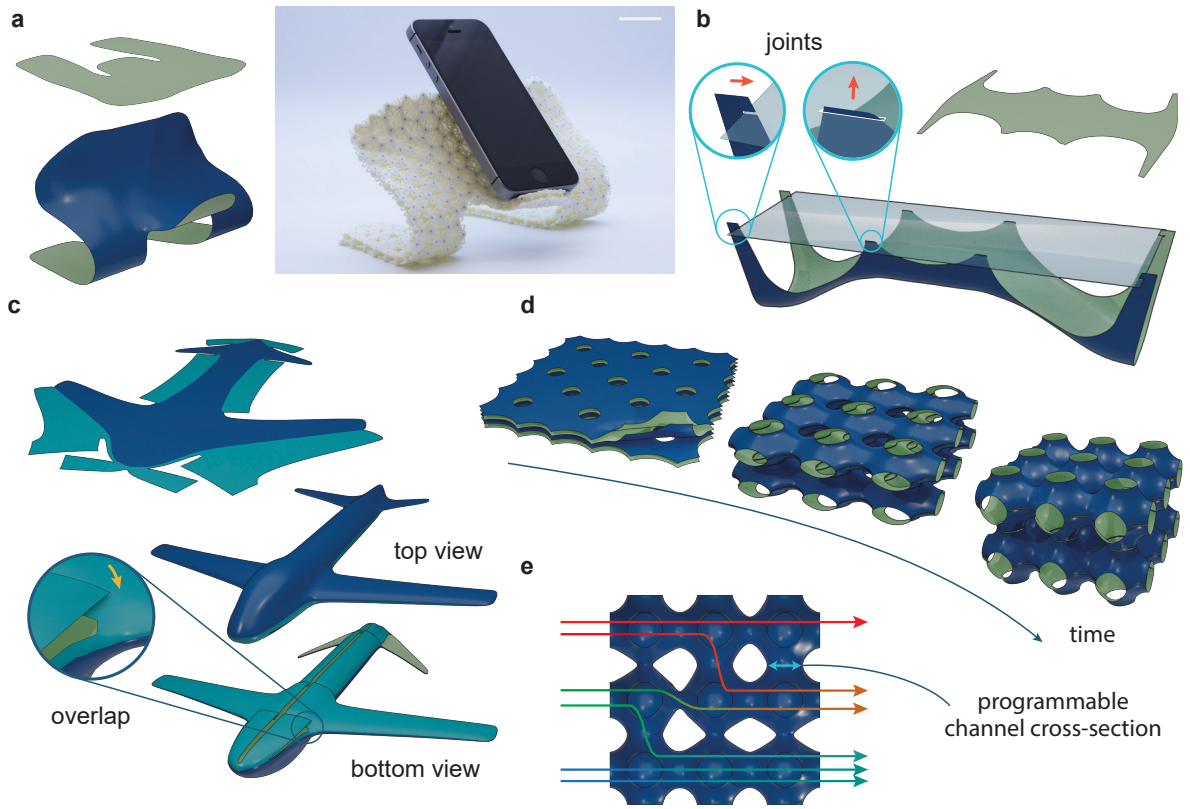


Figure 6.16: Future concepts for spatio-temporally programmed materials Potential applications for shells with programmable self-morphing, showing initial flat and final states. Opposite sides of each sheet are colored blue and green. (a) The load bearing capability of our morphing shells is shown by a mobile phone stand. Scale bar, 3 cm. (b) Self-morphing can be applied to industrial design, for example to build aesthetically curved furniture with sequentially interlocking joints. (c) The outer shell of a self-morphing drone can be fabricated from a single sheet. The smooth aerodynamic shape is achieved with temporal programming of sequentially overlapping parts. (d) Stacked multilayer sheets can be used to create stiff, yet lightweight periodic mesostructures with temporally programmed porosity. Four sheet layers (left) morph into a Schwarz P surface (right) which efficiently distributes external stresses through the structure. (e) Similar structures with programmable channel cross-section between the periodic blocks can be used to adjust flow rates through the device.

and construction industries are close to becoming reality with the fast advances in this field of study. Further generalizations of our approach to other materials such as liquid-crystal elastomers, bio-compatible polymers, and conventional engineering materials whose properties evolve in time due to other stimuli such as temperature, humidity, light, pH, etc., could enable rapid manufacturing of load-bearing structures that can only assume desired geometries through temporally planned deformations upon deployment as well as robotic materials temporally programmed for a broader range of functionalities.

CHAPTER 7

Conclusion

Overall, methods introduced in this thesis significantly contribute to computational design of shell-like structures by pushing the boundaries of state-of-the-art technologies, previously unexplored in-depth. At the same time, we contributed to mechanical engineering by introducing new types of self-morphing mechanisms. Our methods have great potential for applications in a wide range of industries ranging from building construction to miniature self-controlled devices.

Every method presented in this thesis includes computational approaches that we applied to specific materials and realizations of mechanisms. However, we did not tailor our approaches to the particular choices. Conversely, we intended our algorithms, modeling, and design techniques to be generalizable and extendable to alternative materials and mechanisms.

We have introduced an interactive, data-driven approach for material-aware form finding of cold bent glass façades, which can be seamlessly integrated into a typical architectural design pipeline. It allows non-expert users to interactively edit a parametric surface while providing real-time feedback on the deformed shape and feasibility of cold bent glass panels, and automatically optimize for fairness criteria while panels are kept within glass stress limits. We believe our workflow could serve as an inspiration for many other material-aware design problems. Future work in this direction may target extension to different materials, for instance metal, wood, or programmable matter that can respond to external stimuli, such as shape memory polymers or thermo-reactive materials.

Our method for programming doubly curved geometry into self-morphing shells brings to light a range of technical challenges. Flattening a doubly curved three-dimensional surface may lead to self-overlaps as well as excessive stretching or compression. Because of that, feasibility of target shapes remains an open fundamental question dependent on the constraints specific to the self-morphing mechanisms. Moreover, resolving the feasibility problem by the means of automatic or user-assisted manipulation of the target shape and its flattening is an important milestone in the design of self-morphing shells as well as flat sheets externally shaped into a curved state.

Our method for programming spatio-temporal morphing in shells enables spatio-temporal design of objects with high precision and makes possible self-assembly of complex shapes prone to self-collisions while morphing. Further development in this direction may

include automatic design of the morphing process with specific objectives, such as self-collision avoidance. Development of reversible mechanisms with temporally programmable deformations is an exciting research direction in the domain of robotic matter.

Applications of self-actuating shells to biomedical and construction industries are close to becoming reality with the fast advances in this field of study. Further generalizations of our approach to other materials such as liquid-crystal elastomers, bio-compatible polymers, and conventional engineering materials whose properties evolve in time due to other stimuli such as temperature, humidity, light, pH, etc., could enable rapid manufacturing of load-bearing structures that can only assume desired geometries through temporally planned deformations upon deployment as well as robotic materials temporally programmed for a broader range of functionalities.

Finally, it would be beneficial to combine the contributions of all the methods presented in this thesis in order to develop an interactive inverse design tool for self-morphing shells. Such tool may leverage its performance by the use of a deep learning model quickly predicting temporal transformations of the shell. Based on a differentiable neural network, inverse design of the shell's structural elements can be performed. Since the aesthetics of shapes often plays an important role in fabrication, the user would be able to interactively manipulate the target surface or even the whole transformation process under the guidance of the design tool.

Bibliography

[ABB+17] Cedric P. Ambulo, Julia J. Burroughs, Jennifer M. Boothby, Hyun Kim, M. Ravi Shankar, and Taylor H. Ware. Four-dimensional printing of liquid crystal elastomers. *ACS Applied Materials & Interfaces*, 9(42):37332–37339, 2017. PMID: 28967260.

[ABVW14] Sigrid Adriaenssens, Philippe Block, Diederik Veenendaal, and Chris Williams. *Shell Structures for Architecture: Form Finding and Optimization*. Routledge, 2014.

[AHB18] Thomas Auzinger, Wolfgang Heidrich, and Bernd Bickel. Computational design of nanostructural color for additive manufacturing. *ACM Transactions on Graphics (SIGGRAPH 2018)*, 37(4), 2018.

[AMG+19] Thomas Alderighi, Luigi Malomo, Daniela Giorgi, Bernd Bickel, Paolo Cignoni, and Nico Pietroni. Volume-aware design of composite molds. *ACM Trans. Graph.*, 38(4), July 2019.

[AMT+14] B. An, S. Miyashita, M. T. Tolley, D. M. Aukes, L. Meeker, E. D. Demaine, M. L. Demaine, R. J. Wood, and D. Rus. An end-to-end approach to making self-folded 3d surface shapes by uniform heating. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1466–1473, New York, NY, USA, May 2014. IEEE.

[BCMP18] Bernd Bickel, Paolo Cignoni, Luigi Malomo, and Nico Pietroni. State of the art on stylized fabrication. In *Computer Graphics Forum*, volume 37, pages 325–342. Wiley Online Library, 2018.

[Bee15] B Beer. Structural silicone sealed cold-bent glass–high-rise projects experience leading to a new design concept. *GPD Glass Performance Days*, pages 235–240, 2015.

[BFR17] Amit H Bermano, Thomas Funkhouser, and Szymon Rusinkiewicz. State of the art in methods and representations for fabrication-aware design. In *Computer Graphics Forum*, volume 36, pages 509–535. Wiley Online Library, 2017.

[BG17] Aysu Berk and Harry Giles. Quadrilateral panelization of freeform surface structures. *Automation in Construction*, 76:36 – 44, 2017.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[BIVIC07] Jan Belis, Bart Inghelbrecht, Rudy Van Impe, and Dieter Callewaert. Cold bending of laminated glass panels. *Heron*, 52(1-2):123–146, 2007.

[BJ05] Jernej Barbič and Doug L James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph.*, 24(3):982–990, 2005.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[BKR+17] A. A. Bauhofer, S. Krödel, J. Rys, O. R. Bilal, A. Constantinescu, and C. Daraio. Harnessing photochemical shrinkage in direct laser writing for shape morphing of polymer sheets. *Adv. Mater.*, 29(42):1703024, 2017.

[BvRL+19] J William Boley, Wim M van Rees, Charles Lissandrello, Mark N Horenstein, Ryan L Truby, Arda Kotikian, Jennifer A Lewis, and L Mahadevan. Shape-shifting structured lattices via multimaterial 4d printing. *Proceedings of the National Academy of Sciences*, page 201908806, 2019.

[CBW+18] Jiong Chen, Hujun Bao, Tianyu Wang, Mathieu Desbrun, and Jin Huang. Numerical coarsening using discontinuous shape functions. *ACM Trans. Graph.*, 37(4):120, 2018.

[CK08] Peter W. Christensen and Anders Klarbring. *An Introduction to Structural Optimization*. Solid Mechanics and Its Applications. Springer Netherlands, 2008.

[Cla08] Mark Clampin. Status of the james webb space telescope (jwst). In *Space Telescopes and Instrumentation*, volume 7010, 2008.

[CLSM15] Desai Chen, David IW Levin, Shinjiro Sueda, and Wojciech Matusik. Data-driven finite elements for geometry and material design. *ACM Trans. Graph.*, 34(4):74, 2015.

[CMR+18] Paolo Celli, Connor McMahan, Brian Ramirez, Anton Bauhofer, Christina Naify, Douglas Hofmann, Basile Audoly, and Chiara Daraio. Shape-morphing architected sheets with non-periodic cut patterns. *Soft Matter*, 14(48):9744–9749, 2018.

[CPMS14] Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. Field-aligned mesh joinery. *ACM Trans. Graph.*, 33(1):11, 2014.

[CUH15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*, 2015.

[Dat17] Kyriaki Corinna Datsiou. *Design and performance of cold bent glass*. PhD thesis, University of Cambridge, 2017.

[DO07] Erik D Demaine and Joseph O'Rourke. *Geometric folding algorithms*. Cambridge university press, Cambridge, UK, 2007.

[DPW+14] Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. Assembling self-supporting structures. *ACM Trans. Graph.*, 33(6):1, 2014.

[DQH19] A. N. Damdam, N. Qaisar, and M. M. Hussain. Honeycomb-serpentine silicon platform for reconfigurable electronics. *Applied Physics Letters*, 115(11):112105, 2019.

[DVTM16] Levi H. Dudte, Etienne Vouga, Tomohiro Tachi, and L. Mahadevan. Programming curvature using origami tessellations. *Nature Materials*, 15(5):583–588, 2016.

[EIL16] Philipp Eversmann, André Ihde, and Christian Louter. Low cost double curvature–exploratory computational modelling, fe-analysis and prototyping of cold-bent glass. In *Challenging Glass Conference Proceedings*, volume 5, pages 81–92, 2016.

[EKS+10] Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy J Mitra, Helmut Pottmann, and Mark Pauly. Paneling architectural freeform surfaces. *ACM Trans. Graph.*, 29(4):45, 2010.

[ESIL16] Philipp Eversmann, Eike Schling, André Ihde, and Christian Louter. Low-cost double curvature: Geometrical and structural potentials of rectangular, cold-bent glass construction. In *Proceedings of IASS Annual Symposia*, volume 2016, pages 1–10. International Association for Shell and Spatial Structures (IASS), 2016.

[FK11] Thiemo Fildhuth and Jan Knippers. Geometrie und tragverhalten von doppelt gekrümmten ganzglasschalen aus kalt verformten glaslaminaten. *Stahlbau*, 80(S1):31–44, 2011.

[FMD+19] Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, and Alec Jacobson. Latent-space dynamics for reduced deformable simulation. In *Computer Graphics Forum*, volume 38, pages 379–391. Wiley Online Library, 2019.

[FRY+17] Patrick F. Flowers, Christopher Reyes, Shengrong Ye, Myung Jun Kim, and Benjamin J. Wiley. 3d printing electronic components and circuits with conductive thermoplastic filament. *Additive Manufacturing*, 18:156 – 163, 2017.

[FTD+14] S. Felton, M. Tolley, E. Demaine, D. Rus, and R. Wood. A method for building self-folding machines. *Science*, 345(6197):644–646, 2014.

[FTS+13] Samuel M Felton, Michael T Tolley, ByungHyun Shin, Cagdas D Onal, Erik D Demaine, Daniela Rus, and Robert J Wood. Self-folding with shape memory composites. *Soft Matter*, 9(32):7688–7694, July 2013.

[Gau28] C.F. Gauss. *Disquisitiones generales circa superficies curvas*. Typis Dietericianis, 1828.

[GDR+16] C. Gosselin, R. Duballet, Ph. Roux, N. Gaudillière, J. Dirrenberger, and Ph. Morel. Large-scale 3d printing of ultra-high performance concrete – a new processing route for architects and builders. *Materials & Design*, 100:102 – 109, 2016.

[GGP+20] K. Gavriil, R. Guseinov, J. Pérez, D. Pellis, P. Henderson, F. Rist, H. Pottmann, and B. Bickel. Computational design of cold bent glass façades. *ACM Trans. Graph.*, 39(6):208, 2020.

[GGRZ06] Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. Computing discrete shape operators on general meshes. *Comput. Graph. Forum*, 25:547–556, 09 2006.

[GMB17] R. Guseinov, E. Miguel, and B. Bickel. Curveups: Shaping objects from flat plates with tension-actuated curvature. *ACM Trans. Graph.*, 36(4):64, 2017.

[GMN+16] A. S. Gladman, E. A. Matsumoto, R. G. Nuzzo, L. Mahadevan, and J. A. Lewis. Biomimetic 4d printing. *Nat. Mater.*, 15:413–418, 2016.

[GMP+20] R. Guseinov, C. McMahan, J. Pérez, C. Daraio, and B. Bickel. Programming temporal morphing of self-actuated shells. *Nat. Commun.*, 11(1):237, Jan 2020.

[GQM+17] Shuang-Zhuang Guo, Kaiyan Qiu, Fanben Meng, Sung Hyun Park, and Michael C. McAlpine. 3d printed stretchable tactile sensors. *Advanced Materials*, 29(27):1701218, 2017.

[GSC+04] James Glymph, Dennis Shelden, Cristiano Ceccato, Judith Mussel, and Hans Schober. A parametric strategy for free-form glass structures using quadrilateral planar facets. *Automation in Construction*, 13(2):187–202, 2004.

[GSFD+14] Akash Garg, Andrew O Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. Wire mesh design. *ACM Trans. Graph.*, 33(4):66–1, 2014.

[GSH+04] Yotam Gingold, Adrian Secord, Jefferson Han, Eitan Grinspun, and Denis Zorin. A discrete model for inelastic deformation of thin shells. *Technical Report*, 01 2004.

[GUPZ19] Francisca Gil-Ureta, Nico Pietroni, and Denis Zorin. Structurally optimized shells, 2019.

[GWH+19] Philippe Grönquist, Dylan Wood, Mohammad M. Hassani, Falk K. Wittel, Achim Menges, and Markus Rüggeberg. Analysis of hygroscopic self-shaping wood at large scale for curved mass timber structures. *Science Advances*, 5(9), 2019.

[HAB+10a] E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, and R. J. Wood. Programmable matter by folding. *Proc. Natl. Acad. Sci. U.S.A.*, 107(28):12441–12445, 2010.

[HAB+10b] Elliot Hawkes, B An, NM Benbernou, H Tanaka, S Kim, ED Demaine, D Rus, and RJ Wood. Programmable matter by folding. *Proc. National Academy of Sciences*, 107(28):12441–12445, 2010.

[HBA12] Kristian Hildebrand, Bernd Bickel, and Marc Alexa. crdbrd: Shape fabrication by sliding planar slices. *Computer Graphics Forum*, 31(2pt3):583–592, 2012.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[IIM12] Yuki Igarashi, Takeo Igarashi, and Jun Mitani. Beady: interactive beadwork design and construction. *ACM Trans. Graph.*, 31(4):49, 2012.

[KB15] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Int. Conf. on Learning Representations*, 2015.

[KCD+16] Mina Konaković, Keenan Crane, Bailin Deng, Sofien Bouaziz, Daniel Piker, and Mark Pauly. Beyond developable: computational design and fabrication with auxetic materials. *ACM Trans. Graph.*, 35(4):89, 2016.

[KFC+08] Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J Mitra, Alla Sheffer, and Helmut Pottmann. Curved folding. *ACM Trans. Graph.*, 27(3):75, 2008.

[KHB+12] Jungwook Kim, James A Hanna, Myunghwan Byun, Christian D Santangelo, and Ryan C Hayward. Designing responsive buckled surfaces by halftone gel lithography. *Science*, 335(6073):1201–1205, 2012.

[KLPCP18] Mina Konaković-Luković, Julian Panetta, Keenan Crane, and Mark Pauly. Rapid deployment of curved surfaces via programmable auxetics. *ACM Trans. Graph.*, 37(4):106, 2018.

[KMD+19] Arda Kotikian, Connor McMahan, Emily C Davidson, Jalilah M Muhammad, Robert D Weeks, Chiara Daraio, and Jennifer A Lewis. Untethered soft robotic matter with passive control of shape morphing and propulsion. *Science Robotics*, 4(33):eaax7044, 2019.

[Koi66] Warner T. Koiter. On the nonlinear theory of thin elastic shells. 1966.

[KTB+18] Arda Kotikian, Ryan L. Truby, John William Boley, Timothy J. White, and Jennifer A. Lewis. 3d printing of liquid crystal elastomeric actuators with spatially programed nematic order. *Adv. Mater.*, 30(10):1706164, 2018.

[KWD+15] Tsz-Ho Kwok, Charlie C. L. Wang, Dongping Deng, Yunbo Zhang, and Yong Chen. Four-dimensional printing for freeform surfaces: Design optimization of origami and kirigami structures. *Journal of Mechanical Design*, 137(11), 10 2015.

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, July 2002.

[LPW+06] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3):681–689, 2006.

[LSDG17] Ying Liu, Brandi Shaw, Michael D. Dickey, and Jan Genzer. Sequential self-folding of polymer sheets. *Science Advances*, 3(3), 2017.

[LSW+18] Ran Luo, Tianjia Shao, Huamin Wang, Weiwei Xu, Xiang Chen, Kun Zhou, and Yin Yang. Nnwarp: Neural network-based nonlinear deformation. *IEEE TVCG*, 2018.

[LT03] Wei Liu and Joseph J Talghader. Current-controlled curvature of coated micromirrors. *Optics letters*, 28(11):932–934, 2003.

[LWL+09] Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On Centroidal Voronoi Tessellation–Energy Smoothness and Fast Computation. *ACM Trans. Graph.*, 28(4):Article 101, August 2009.

[LXW+11] Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen, and Guoping Wang. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph.*, 30:#140, 1–10, 2011.

[MAB+15] Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, and Leif Kobbelt. Reduced-order shape optimization using offset surfaces. *ACM Trans. Graph.*, 34(4):102:1–102:9, 2015.

[MDBL17] Romain Mesnil, Cyril Douthe, Olivier Baverel, and Bruno Léger. Marionette meshes: modelling free-form architecture with planar facets. *International Journal of Space Structures*, 32(3-4):184–198, 2017.

[MGE07] F. Massarwi, C. Gotsman, and G. Elber. Papercraft models using generalized cylinders. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 148–157, New York, NY, USA, Oct 2007. IEEE.

[MI11] Jun Mitani and Takeo Igarashi. Interactive design of planar curved folding by reflection. *Pacific Graphics (short paper)*, 19, 2011.

[MKG+19] Niloy J. Mitra, Iasonas Kokkinos, Paul Guerrero, Nils Thuerey, Vladimir Kim, and Leonidas Guibas. Creativeai: Deep learning for graphics. In *SIGGRAPH 2019 Courses*, Siggraph 2019, 2019.

[MPI+18] Luigi Malomo, Jesús Pérez, Emmanuel Iarussi, Nico Pietroni, Eder Miguel, Paolo Cignoni, and Bernd Bickel. Flexmaps: Computational design of flat flexible shells for shaping 3d objects. In *SIGGRAPH Asia 2018 Technical Papers*, page 241. ACM, 2018.

[MSA+18] I. Maskery, L. Sturm, A.O. Aremu, A. Panesar, C.B. Williams, C.J. Tuck, R.D. Wildman, I.A. Ashcroft, and R.J.M. Hague. Insights into the mechanical properties of several triply periodic minimal surface lattice structures made by polymer additive manufacturing. *Polymer*, 152:62 – 71, 2018. SI: Advanced Polymers for 3DPrinting/Additive Manufacturing.

[MTN+15] Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. Interactive design of 3d-printable robotic creatures. *ACM Trans. Graph.*, 34(6):216, 2015.

[MYI+15] Yiqi Mao, Kai Yu, Michael S. Isakov, Jiangtao Wu, Martin L. Dunn, and H. Jerry Qi. Sequential self-folding structures by 3d printed digital shape memory polymers. *Sci. Rep.*, 5:13616, September 2015. Article.

[NEB+14] J. H. Na, A. A. Evans, J. Bae, M. C. Chiappelli, C. D. Santangelo, R. J. Lang, T. C. Hull, and R. C. Hayward. Programming reversibly self-folding origami with micropatterned photo-crosslinkable polymer trilayers. *Adv. Mater.*, 27(1):79–85, 2014.

[Ogd97] Ray D. Ogden. *Non-linear Elastic Deformations*. Dover Publications, 1997.

[OSV+16] Jifei Ou, Mélina Skouras, Nikolaos Vlavianos, Felix Heibeck, Chin-Yi Cheng, Jannik Peters, and Hiroshi Ishii. aeromorph-heat-sealing inflatable shape-change materials for interaction design. In *Proc. 29th Annual Symposium on User Interface Software and Technology*, pages 121–132, New York, NY, USA, 2016. ACM.

[PEVW15] Helmut Pottmann, Michael Eigensatz, Amir Vaxman, and Johannes Wallner. Architectural geometry. *Computers and Graphics*, 47:145–164, 2015.

[Pip86] Allen C. Pipkin. The relaxed energy density for isotropic elastic membranes. *IMA Journal of Applied Mathematics*, 36(1):85–99, 01 1986.

[PKD+19] Davide Pellis, Martin Kilian, Felix Dellinger, Johannes Wallner, and Helmut Pottmann. Visual smoothness of polyhedral surfaces. *ACM Trans. Graph.*, 38(4):260:1–260:11, 2019.

[PKLI+19] Julian Panetta, MINA Konaković-Luković, Florin Isvoranu, Etienne Bouleau, and Mark Pauly. X-shells: A new class of deployable beam structures. *ACM Trans. Graph.*, 38(4):83, 2019.

[PKWB18] P. Plucinsky, B. A. Kowalski, T. J. White, and K. Bhattacharya. Patterning nonisometric origami in nematic elastomer sheets. *Soft Matter*, 14:3127–3134, 2018.

[PNdJO14] Tobias Pfaff, Rahul Narain, Juan Miguel de Joya, and James F. OBrien. Adaptive tearing and cracking of thin sheets. *ACM Trans. Graph.*, 33(4), July 2014.

[PSB+08] Helmut Pottmann, Alexander Schiftner, Pengbo Bo, Heinz Schmiedhofer, Wenping Wang, Niccolo Baldassini, and Johannes Wallner. Freeform surfaces from single curved panels. *ACM Trans. Graph.*, 27(3):76, 2008.

[PTH+15] E. Peral, S. Tanelli, Z. Haddad, O. Sy, G. Stephens, and E. Im. Raincube: A proposed constellation of precipitation profiling radars in cubesat. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1261–1264, July 2015.

[PW89] Alexander Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. 1989.

[RDC+12] Jennie Ryu, Matteo D'Amato, Xiaodong Cui, Kevin N Long, H Jerry Qi, and Martin L Dunn. Photo-origami—bending and folding polymers with light. *Applied Physics Letters*, 100(16):161908, 2012.

[RZM+14] Dan Raviv, Wei Zhao, Carrie McKnelly, Athina Papadopoulou, Achuta Kadambi, Boxin Shi, Shai Hirsch, Daniel Dikovsky, Michael Zyracki, Carlos Olguin, Ramesh Raskar, and Skylar Tibbits. Active printed materials for complex self-evolving deformations. *Scientific Reports*, 4(1):7422, 2014.

[SAK+19] Mohand O. Saed, Cedric P. Ambulo, Hyun Kim, Rohit De, Vyom Raval, Kyle Searles, Danyal A. Siddiqui, John Michael O. Cue, Mihaela C. Stefan, M. Ravi Shankar, and Taylor H. Ware. Molecularly-engineered, 4d-printed liquid crystal elastomer actuators. *Adv. Funct. Mater.*, 29(3):1806412, 2019.

[SCGT15] Mélina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. Interactive surface design with interlocking elements. *ACM Trans. Graph.*, 34(6):224, 2015.

[SIC+11] Robert F Shepherd, Filip Ilievski, Wonjae Choi, Stephen A Morin, Adam A Stokes, Aaron D Mazzeo, Xin Chen, Michael Wang, and George M Whitesides. Multigait soft robot. *Proc. National Academy of Sciences*, 108(51):20400–20403, 2011.

[SP13] Yuliy Schwartzburg and Mark Pauly. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum*, 32(2pt3):317–326, 2013.

[SRBR19] Emmanuel Siéfert, Etienne Reyssat, José Bico, and Benoît Roman. Bio-inspired pneumatic shape-morphing elastomers. *Nature Materials*, 18(1):24–28, Jan 2019.

[SSP08] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. *ACM Trans. Graph.*, 27(3):77:1–77:11, August 2008.

[STBG12] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. Computational Design of Rubber Balloons. *Computer Graphics Forum*, 31(2pt4):835–844, May 2012.

[Ste90] D. J. Steigmann. Tension-field theory. *Proc. Royal Society of London. Series A: Mathematical and Physical Sciences*, 429, 1990.

[STG16] Christian Schumacher, Bernhard Thomaszewski, and Markus Gross. Stenciling: Designing structurally-sound surfaces with decorative patterns. In *Computer Graphics Forum*, volume 35, pages 101–110. Wiley Online Library, 2016.

[STK+14] Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. Designing inflatable structures. *ACM Trans. Graph.*, 33(4):1–10, July 2014.

[STTP14] Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. High-contrast computational caustic design. *ACM Trans. Graph.*, 33(4):74:1–74:11, July 2014. Proc. SIGGRAPH 2014.

[SVB+12] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. Stress relief: improving structural strength of 3d printable objects. *ACM Trans. Graph.*, 31(4):48, 2012.

[SXZ+17] Adriana Schulz, Jie Xu, Bo Zhu, Changxi Zheng, Eitan Grinspun, and Wojciech Matusik. Interactive design space exploration and optimization for cad models. *ACM Trans. Graph.*, 36(4):157:1–157:14, July 2017.

[SZB18] Christian Schumacher, Jonas Zehnder, and Moritz Bächer. Set-in-stone: worst-case optimization of structures weak in tension. In *SIGGRAPH Asia 2018 Technical Papers*, page 252. ACM, 2018.

[TBWP16] Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. Interactive design of developable surfaces. *ACM Trans. Graph.*, 35(2):12, 2016.

[TFM+14] Michael T Tolley, Samuel M Felton, Shuhei Miyashita, Daniel Aukes, Daniela Rus, and Robert J Wood. Self-folding origami: shape memory composites activated by uniform heating. *Smart Materials and Structures*, 23(9):094006, 2014.

[TGA+18] Ye Tao, Jianzhe Gu, Byoungkwon An, Tingyu Cheng, Xiang 'Anthony' Chen, Xiaoxiao Zhang, Wei Zhao, Youngwook Do, Teng Zhang, and Lining Yao. Demonstrating thermorph: Democratizing 4d printing of self-folding materials and interfaces. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, pages D405:1–D405:4. ACM, 2018.

[Tol03] Sivan Toledo. Taucs, a library of sparse linear solvers, 2003.

[UB18] Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Trans. Graph.*, 37(4):89, 2018.

[UIM12] Nobuyuki Umetani, Takeo Igarashi, and Niloy Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4), July 2012.

[UMK17] Erva Ulu, James Mccann, and Levent Burak Kara. Lightweight structure design under force location uncertainty. *ACM Trans. Graph.*, 36(4):158:1–158:13, July 2017.

[VCO20] Mickeal Verschoor, Dan Casas, and Miguel A. Otaduy. Tactile Rendering Based on Skin Stress Optimization. *ACM Trans. Graph.*, 39(4), 2020.

[VHWP12] Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. Design of self-supporting surfaces. *ACM Trans. Graph.*, 31(4):87, 2012.

[Wei12] Clarisse Weischedel. A discrete geometric view on shear-deformable shell models. 2012.

[WGX+17] F. Wang, X. Guo, J. Xu, Y. Zhang, and C. Q. Chen. Patterning curved three-dimensional structures with programmable kirigami designs. *J. Appl. Mech.*, 84(6):061007, 2017.

[WMW+15] T. H. Ware, M. E. McConney, J. J. Wie, V. P. Tondiglia, and T. J. White. Voxelated liquid crystal elastomers. *Science*, 347(6225):982–984, 2015.

[WS19] Katja Wolff and Olga Sorkine-Hornung. Wallpaper pattern alignment along garment seams. *ACM Trans. Graph.*, 38(4), 2019.

[WSFM19] Tuanfeng Y. Wang, Tianjia Shao, Kai Fu, and Niloy J. Mitra. Learning an intrinsic garment space for interactive authoring of garment animation. *ACM Trans. Graph.*, 38(6), November 2019.

[YC04] Jun-Hai Yong and Fuhua (Frank) Cheng. Geometric hermite curves with minimum strain energy. *Comp. Aided Geom. Design*, 21(3):281 – 301, 2004.

[ZXZ+17] Haiming Zhao, Weiwei Xu, Kun Zhou, Yin Yang, Xiaogang Jin, and Hongzhi Wu. Stress-constrained thickness optimization for shell object fabrication. In *Computer Graphics Forum*, volume 36, pages 368–380. Wiley Online Library, 2017.

[ZYN+15] Y. Zhang, Z. Yan, K. Nan, D. Xiao, Y. Liu, H. Luan, H. Fu, X. Wang, Q. Yang, J. Wang, W. Ren, H. Si, F. Liu, L. Yang, H. Li, J. Wang, X. Guo, H. Luo, L. Wang, Y. Huang, and J. A. Rogers. A mechanically driven form of kirigami as a route to 3d mesostructures in micro/nanomembranes. *Proc. Natl. Acad. Sci. U.S.A.*, 112(38):11757–11764, 2015.

APPENDIX A

Computational design of cold bent glass façades

A.1 Sampling panel boundaries

We briefly describe how the panel boundaries forming the training set for our data-driven model are sampled (Sec. 4.4.2). We parameterize panel boundaries invariantly to rigid transformations, by corner pairwise squared-distances $\mathbf{d}$, edge-plane inclinations $\boldsymbol{\gamma}$, and halfedge tangent directions $\boldsymbol{\theta}$ (Sec. 4.2). To sample $\mathbf{d}$ such that it represents a valid quad, we start with two adjacent edge lengths l_1, l_2, an angle α between them, and a displacement $\mathbf{a}$ of the remaining vertex from the point that would form a parallelogram. We sample each of these parameters as follows:

- l_1, $l_2 \sim \text{Uniform}[0.15, 0.60]$; this corresponds to 15–60 cm for a 1 mm thick panel,
- $\alpha \sim \text{Uniform}[60°, 120°]$,
- $\mathbf{a}$ is given by sampling a point on the unit sphere, then scaling it by a factor drawn from $\text{Uniform}[0, \min(l_1, l_2)/4]$,
- $\gamma_i \sim \text{Uniform}[-90°, 90°]$,
- θ_i is given by arccos of a value sampled from $\text{Uniform}[\cos 5°, 1]$, negated with probability 1/2, so $\theta_i \in [-5°, 5°]$.

Note that our model for the deformed shape and stress is invariant under scaling of all geometric magnitudes. Our sampling ranges are chosen to allow scaling the results to thickness/curvature ratios commonly used in cold bent glass façades.

APPENDIX B

Encoding geometric information in material

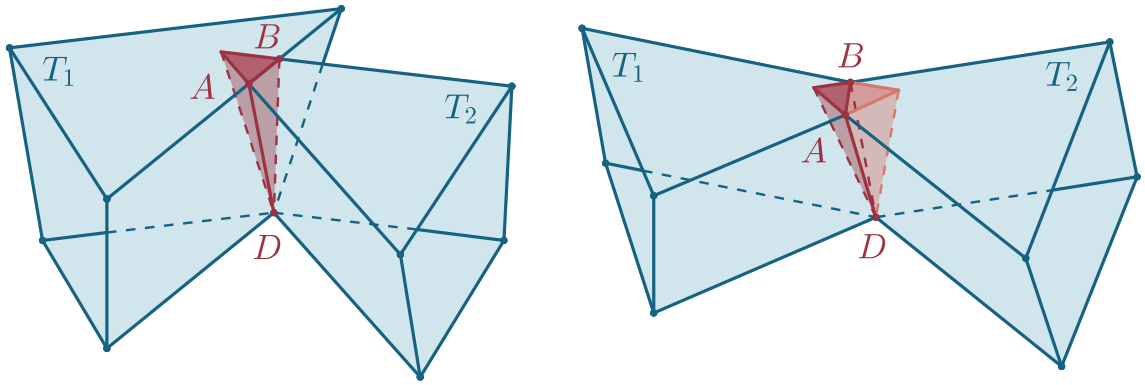
B.1 Computation of contact points

We only take into account interaction between tiles sharing a front face edge. These pairs of tiles always have a flat interface with four contact points shared by both tiles. These points define a new "truncated" shape of each tile such that they do not intersect anywhere in the actuated structure.

In order to resolve self-intersections of tiles and compute correct contact points, we cut each tile by three planes, one per front face vertex. In Fig. B.1, we denote D as the front face vertex shared by two tiles, and A and B are the intersection points for tile edges and contact planes. We cut each pair of tiles with planes ABD.

The points A and B can be located on one edge of the tile or on two different edges. We call an intersection Type 1 if one tile has both intersection points on one edge (see Fig. B.1a), and Type 2 if for both tiles the intersection points lie on different edges (see Fig. B.1b). In the first case, only one tile is effectively cut, whereas in the second case both tiles are cut. It is easy to see that there is no other type of intersection.

Originally, each contact face has contact points set to the four vertices of the contact face, but each time the tile is cut, we update the contact points to the new intersection points. Note that a pair of tiles in contact never has an intersection to resolve, since they share the entire front face edge, not just one vertex.



(a) Type 1: only T_2 is affected.

(b) Type 2: both tiles are affected.

Figure B.1: Two types of intersections of tiles, cut by plane ABD, where D is a front face vertex and A and B are the intersection points for tile edges and contact planes.

APPENDIX C

Encoding temporal information in material

C.1 Experiments

All specimens used for bracket characterization and the outer layers of the shape-morphing structures were 3D-printed using a Stratasys J750 using Vero Pure White material. Colored markers were included in the bases to facilitate visualization. Water was kept at 56° C using a temperature controller, and two Canon 700D cameras were used for imaging.

The mechanical properties of the brackets were measured using a Zwick tensile tester with a custom-built water tank attachment. Experiments measuring strain restitution after unloading were conducted to estimate the plastic fraction of the deformation.

All shells were fabricated by first uniformly stretching a latex sheet of thickness 0.5 mm to 900% its initial area. After cleaning the membrane surfaces with 2-propanol, the printed lattices are glued to the membrane. In each structure, several bases have holes to align the opposite shell layers using push-pins. Latex surplus surrounding the assembled flat shell is removed, then the shells are submerged into a $350 \times 350 \times 350$ mm water tank to induce shape-morphing.

C.2 Data fitting and simulation

Experimental data was used to generate a force-displacement model of brackets by combining simpler fitting components. First, displacement-force curves were fitted so that the initial state corresponded to the behavior at room temperature. Second, displacement rates dependent on time in water were fitted. Based on the combination of these two fittings, the parameter domain was resampled to yield time-dependent force-displacement relationships. Data from the plasticity experiments were used to build the final elastoplastic model used in the simulation software for inverse design.

In addition to implementing this data-driven bracket compression model in simulations, shear-resisting elements are included to capture brackets' geometry-based shear resistance. Bases are simulated as rigid bodies and a FEM approach is used for the membrane simulation. Bumper collisions are modelled as abrupt bracket stiffness increases. All

simulation elements are coupled via shared vertices. We implemented the simulations using a C++ code developed in-house. A simple user interface was designed to import target surfaces and specify time landscapes. User inputs are automatically processed to configure the simulated structure and display resulting morphing processes. Once the desired morphing is achieved, the system automatically generates structures for 3D printing.

C.3 3D scanning

An HP David SLS-3 structured light scanner with two cameras was used to generate textured 3D meshes in OBJ format. Then, the textures were filtered to obtain binary images with markers. Markers were lifted to their actual scanned positions using the 3D mesh and resolved to single points. The obtained point clouds were registered to the clouds generated by the simulation software and points were matched using Munkres' algorithm.