String Actuated Curved Folded Surfaces

MARTIN KILIAN
University College London, Vienna University of Technology
and
ARON MONSZPART and NILOY J. MITRA
University College London

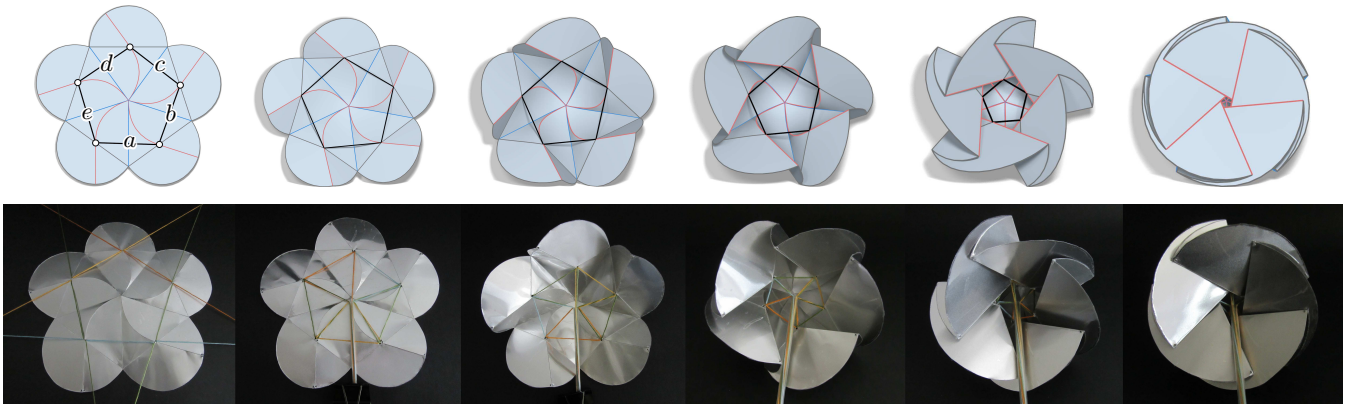

Fig. 1: The top row shows a crease pattern together with the strings $\mathcal{S} = \{a, b, c, d, e\}$ computed by our algorithm and simulation of the induced, string driven folding motion. The bottom row shows a physical realization of the same crease pattern as a thin aluminum sheet. Each string is realized as a strand of colored thread. Strings are collected at a single anchor point and folding is driven by pulling the threads.

Curved folded surfaces, given their ability to produce elegant freeform shapes by folding flat sheets etched with curved creases, hold a special place in computational Origami. Artists and designers have proposed a wide variety of different fold patterns to create a range of interesting surfaces. The creative process, design as well as fabrication, is usually only concerned with the static surface that emerges once folding has completed. Folding such patterns, however, is difficult as multiple creases have to be folded simultaneously to obtain a properly folded target shape. We introduce string actuated curved folded surfaces that can be shaped by pulling a network of strings thus vastly simplifying the process of creating such surfaces and making the folding motion an integral part of the design. Technically, we solve the problem of which surface points to string together and how to actuate them by locally expressing a desired folding path in the space of isometric shape deformations in terms of novel string actuation modes. We demonstrate the validity of our approach by computing string actuation networks for a range of well known crease patterns and testing their effectiveness on physical prototypes. All the examples in this paper can be downloaded for personal use from http://geometry.cs.ucl.ac.uk/projects/2017/string-actuated/.

CCS Concepts: •**Computing methodologies** → **Shape modeling;**

Additional Key Words and Phrases: curved folding, computational design, string actuation, folding motion, computational Origami, fabrication

ACM Reference Format:

Kilian, M., Monszpart, A., Mitra, N. J., 2017. String Actuated Curved Folded Surfaces. *ACM Trans. Graph.* 36, 3, Article 25 (March 2017), 13 pages.
DOI: http://dx.doi.org/10.1145/3015460

Martin Kilian was supported by the Erwin Schrödinger fellowship J-3678-N25 awarded by the Austrian Science Fund (FWF), ERC StG-2013-335373, and the DFG-Collaborative Research Center, TRR 109, 'Discretization in Geometry and Dynamics' through grant I-706-N26 of FWF. Aron Monszpart was supported by a Google PhD Fellowship, the ERC Starting Grant SmartGeometry (StG-2013-335373), and Adobe Research.

DOI: http://dx.doi.org/10.1145/3015460

1. INTRODUCTION

Originally popularized by David Huffman [1976], curved folded shapes continue to capture the interest and imagination of architects, artists, and hobbyists. The possibility of folding a set of simple curves, commonly referred to as creases, on a single flat sheet into a complex freeform surface, without any joining or gluing, is both intriguing and fascinating.

Artists and designers create such curved folded surfaces by a combination of prior experience, trial-and-error, and prototyping with paper or similar materials like felt for example. Instructions for actually folding such surfaces are difficult to produce. The key complexity comes from the requirement that multiple creases have

to be folded simultaneously to arrive at the target shape. Otherwise, the folding process can easily reach a 'locked' state where further folding is not possible without producing undesirable creases or damaging the sheet. This is different from classical Origami employing only straight folds that are folded sequentially. Hence, curved folding requires significant experience, expertise, and many folding attempts (see [McArthur 2013]).

In case of architectural and industrial applications, it is highly desirable to have an automated folding process. Unfortunately, very little is known in this context. A rare exception is the particular demonstrations from RoboFold [Epps 2014]. Such robotic folding, however, strongly limits the type of folded surfaces due to space required for robotic arm manipulation and also for placing suction cups on the sheet. This greatly restricts wide spread usage of curved folding and limits large-scale installations.

In this paper, we introduce string actuated curved folded surfaces as a versatile yet simple contraption for folding complex curved folded surfaces starting from their unfolding. We ask the question of how to fold a given crease pattern (CP) by pulling a network of strings. Given a CP, this requires answering: (i) what is the folded shape and what is a corresponding folding motion; and, more importantly, (ii) how to actuate the CP with easy-to-rig networks of strings.

Our method works in two phases: first, it finds a folding motion that takes a flat CP to a folded shape via an approximately isometric deformation; and then discovers a corresponding string network (i.e., which surface points to connect) that can be actuated to produce the same folding sequence.

Technically, we introduce the notion of string actuation modes as deformation vector fields that characterize surface deformations when pulling a set of strings connecting pairs of surface points. We then demonstrate how solving for a network of strings amounts to expressing the desired folding sequence in terms of a minimal set of time-consistent string actuation modes. The resultant solution gives the final network of strings, which when pulled folds the flat sheet to a curved folded surface, see Figure 1.

We demonstrate our approach by computing string actuation networks for a set of classical crease patterns and validate them by physically constructing them. In summary, we introduce a string actuated curved folded surface as a planar sheet augmented with a network of strings which when pulled can bring the flat sheet to a folded shape by simultaneously actuating multiple curved folds to appropriately fold the surface.

2. RELATED WORK

We briefly survey the research efforts that are closely related to our problem.

Computational fabrication. In recent years, with growing interest in customized fabrication, various approaches have been proposed for computational design and fabrication. Coros et al. [2013] and Ceylan et al. [2013] support gear-driven computational rigging for humanoid characters, Song et al. [2013] support design of self-supporting reciprocal structures, Deuss et al. [2014] propose construction sequences along with intermediate support structures for constructing self-supporting structures, Garg et al. [2014] create freeform surfaces supported by woven wires, Mellado et al. [2014] explore construction sequences for easy installation of reciprocal structures, Pérez et al. [2015] support design and fabrication of flexible meshes, while Koo et al. [2016] propose design refinements to reduce wastage of materials due to offcuts in plank-based furniture.

More relevant to our work are the following two large scale fabrication efforts: Robofold© has been investigating ways to link industrial design with curved folding to get robots to fold surfaces starting from flat metal sheets with creases scored using a laser cutter, see `www.robofold.com`. They have been combining physical simulation and experience from manually folding with paper to design actuation forces and folding paths for the suction-based robotic arms. While the results have been fascinating, such a folding procedure limits closely folded surfaces and pleats as seen in many curved folded shapes. The other notable effort is the impressive case study 99 Failures Pavilion [Obuchi et al. 2013] that realize a tensegrity structure by balancing torsion and compression forces in sculptural objects. The project symbolizes a combination of computation design, large scale physical prototyping, and realizing a beautiful mixed material installation. Motivated by these examples we propose an algorithm for computing string actuation networks for given crease patterns.

Developable surfaces. In the context of differential geometry, developable surfaces are well understood and there exists a considerable body of work to integrate them with standard CAGD techniques [Chu and Squin 2002; Aumann 2003] or using special representations, such as rectifying developables [Bo and Wang 2007]. Mesh-based approaches to developable surfaces include fitting developable surface patches to point clouds [Peternell 2004], minimizing Gaussian curvature for designing developable Bézier patches [Guoliang and Yanan 2006], constructing a developable surface from a set of space curves [Rose et al. 2007], Laguerre minimal surfaces [Pottmann et al. 2008], or creating developable surfaces from silhouette curves [Jung et al. 2015]. Kilian et al. [2008] reconstruct scanned paper surfaces as Origami patterns using curved folds, utilizing the reference surface to estimate ruling directions. Later, Solomon et al. [2012] presented a subdivision based modeling approach involving curved folds assuming the crease pattern and corresponding crease angles as input. Tang et al. [2015] encode developability conditions as nonlinear constraints and solve for curved folded surfaces using a projection-based efficient constraint solver for interactive modeling. These efforts typically assume access to target shapes, which are analyzed to initialize direction of rulings, estimate the crease angles, or tessellate the planar domain. Above approaches mainly focus on optimizing the final folded shapes, rather than designing mechanisms to facilitate the folding process, as is our goal.

Computational Origami. Origami continues to draw the attention (and time) of amateurs and skilled designers. In recent years there has been significant interest in mapping the underlying mathematical constraints to computational methods in an effort to create novel tools for designing new Origami shapes. For example, in the context of traditional Origami involving straight creases, the widely used TreeMaker package [Lang 2011] provides computational support to create classical Origami bases. This has resulted in design and creation of very intricate folded shapes (see [McArthur 2013] for many examples).

In the context of design of pleated shapes, i.e., paper surfaces exhibiting curved folds that are based on reflections [Mitani and Igarashi 2011], column-shaped structures [Mitani 2012], and surfaces of revolution [Mitani 2009] have been investigated. Tachi [2010] develop an algorithm to 'Origamize' an input freeform shape into a polyhedral surface that can then be tuck-folded from a sheet of paper. Li et al. [2010] propose a computational framework to help create paper popups by allowing cutting and folding. For a detailed exposition on geometric folding, we re-

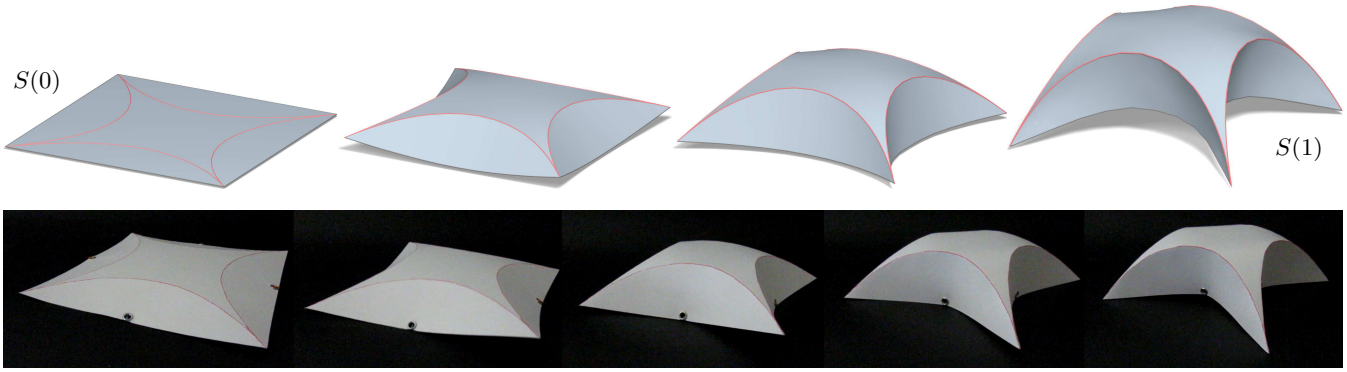

Fig. 2: A sequence of shapes along the target deformation S, starting with the planar CP, are shown in the top row. Red crease curves indicate mountain folds. Boundary curves and creases which have not been given a fold orientation are shown in gray. The bottom row shows a corresponding string actuated folding sequence. Threads used for actuation are shown in Figure 5, showing the model from below.

fer the reader to the now classic reference [Demaine and O' Rourke 2007]. Tachi [2013] introduce kinematic Origami by investigating design exploration governed by the equilibrium of forces from the elastic bending of each panel. The resultant shapes are intricate and fascinating, but restricted to particular shape families whose mathematical characteristics are directly exploited in these modeling setups. Zhu et al. [2013] develop an interactive system to design and explore thin-plate forms through fold fields, a generalization of discrete fold graphs in Origami. They first approximate a folded surface by assembling locally folded patches and later refine using a nonlinear physical simulation. In contrast, beyond efficiently computing a folding sequence from a CP, we focus on designing an actuation sequence to achieve such a folding path.

Geometric deformation. In the wider context of geometry processing, surface modeling and deformation techniques remain active research topics. One of the most widely used methods preserves Laplacian coordinates to retain surface details during surface modeling and editing tasks [Sorkine et al. 2004]. The work has later been generalized to the local-global as-rigid-as-possible surface deformation method developed by Sorkine and Alexa [2007]. Relevant to our goal are methods for constrained modeling and deformation (e.g., [Yang et al. 2011]) that aim to achieve or preserve certain surface characteristics (see the survey by [Botsch and Sorkine 2008]). Skouras et al. [2013] investigate physical deformation of elastic shapes by pulling along designed threads using finite elements to analyze the internal elastic behavior of the (fabricated) models. In other efforts, isometric deformations have been investigated as geodesics in appropriate shape spaces [Kilian et al. 2007; Wirth et al. 2011], or in the context of bending energies in the PriMo setup [Botsch et al. 2006]. In our implementation we extend on the PriMo approach to handle creases and improve treatment of near isometric deformations. Further, we model the effect of contracting a string network by proposing a variant that captures the resultant deformation behavior without having to explicitly specify target handle positions (which are unknown).

Shell models. Algorithms have also been proposed to explicitly model shell structures [Grinspun et al. 2003; Burgoon et al. 2006; Heeren et al. 2014]. Recently, modeling deformation behavior of paper in the context of thin shells has been explored by Narain et al. [2013]. Schreck et al. [2015] propose a hybrid surface model to support crumpling behavior for interactive paper simulation while explicitly accounting for isometric deformations. However, such methods do not help in predicting a folded shape starting only from an input crease pattern, as is our goal. Specifically, since the required folding sequence is also unknown, and hence nearly impossible to guess what sequence of forces to apply, often simultaneously involving multiple creases, in order to reach the final shape. Thus, having access to an accurate deformation model by itself does not solve our problem. Further, even if the exact force sequences can be established, they do not necessarily simplify the manual folding of a physical crease pattern.

3. OVERVIEW

We consider shapes defined by two-dimensional crease patterns (CP) that exhibit curved and straight creases, such as the ones shown in the top left of Figures 1 and 2. Given such a pattern, we compute a sparse string graph on top of the CP such that a physical realization of the CP (as a sheet of paper, aluminum, or similar thin sheet material) and the graph (with edges realized as threads attached to the sheet) lifts the flat sheet to the corresponding three-dimensional shape when the threads are pulled (Figure 2, bottom).

Our key insight is that the effect of any single string on the deformation of the developable surface can be studied independently, and these deformations can be linearly combined to yield a local estimate of how the surface will evolve in time through shape space under the influence of a network of strings. Since we can a priori extract the desired trajectory of the surface in space, the nonlinear deformation due to string contraction can be re-linearized at each point along that trajectory and solved for a consistent network of strings across time to realize a desired curved folding.

To this end, we first compute a folding motion S that serves as the target deformation (Figure 2, top row). Even though this deformation is driven by fold angles we only need a valid mountain/valley assignment of folds to compute it, while actual fold angles are determined by the algorithm. Given such a target deformation sequence, we start from a dense string graph whose vertices are sampled from the CP. Based on local analysis of the deformation induced by individual strings – so called actuation modes – we pick the strings whose combined deformation locally best approximates the given target motion. To arrive at a consistent selection of strings that can drive the deformation globally, we formulate the selection process as an ℓ_0-sparsity optimization problem.

We realize the computed results as paper or aluminum models. To make the model fold without buckling, creases are scored or

etched and precreased, a commonly applied technique when folding Origami tessellations to achieve high quality results. Finally, threads are fit to the model by punching holes at the computed attachment points.

4. STRING ACTUATED FOLDING

For the time being we assume that we are given a target deformation, i.e., a folding motion,

$$S : [0,1] \times U \to \mathbb{R}^3$$
$$(t, \mathbf{u}) \mapsto S(t, \mathbf{u})$$

that assigns each point $\mathbf{u} \in U \subset \mathbb{R}^2$ of the flat sheet its time dependent location $S(t, \mathbf{u})$ in space, see Figure 2 for an example. Details on how to compute such a deformation sequence are given in Section 6. For fixed value of t each $S(t, \mathbf{u}) : U \to \mathbb{R}^3$ is a surface parametrization in the usual sense. We abuse notation by writing $S(t)$ when referring to the shape at time t. For now, assuming access to the folding motion S, we argue in the space of isometric deformations. First, we show how to solve the local problem of selecting a set of strings that facilitate the transition from $S(t)$ to a close-by shape $S(t + \Delta t)$ for small time steps Δt along the target folding motion.

Local analysis. At every instant $t \in [0, 1]$ of time the infinitesimal deformation of the surface $S(t)$ is described by the vector field

$$X : [0,1] \times U \to \mathbb{R}^3$$
$$(t, \mathbf{u}) \mapsto X(t, \mathbf{u}) := \frac{\partial S}{\partial t}(t, \mathbf{u}),$$

which assigns every point $\mathbf{u} \in U$ of the planar sheet its time dependent velocity vector $X(t, \mathbf{u})$. Again we just write $X(t)$ to refer to the vector field at time t as a whole. Considering S as a curve in the space of developable surfaces, $X(t)$ is the tangent vector to this curve at time t, see Figure 3. This vector field describes the movement of every surface point during folding.

We decompose $X(t)$ into a finite set of deformation fields $X_i(t)$ that correspond to infinitesimal deformations induced by a set of strings attached to the surface $S(t)$. Note that the term string always refers to a connection between exactly two surface points. Let $\mathbf{u}_i, \mathbf{v}_i \in U$ be two points to be joined by a string. Attaching this string to $S(t)$ yields the line segment $\mathbf{s}_i(t) = [S(t, \mathbf{u}_i), S(t, \mathbf{v}_i)]$. Tightening this string induces a deformation of $S(t)$. We refer to the corresponding deformation field $X_i(t)$ as the actuation mode induced by the string $\mathbf{s}_i(t)$. Such a string induced deformation is consistent with the target deformation if the corresponding mode $X_i(t)$ is similar to $X(t)$. Referring to Figure 3, the tangent vectors $X(t)$ and $X_i(t)$ have to point in the same direction.

In general a single string is not sufficient to approximate the target deformation reasonably well. Fidelity can be improved by superposing modes of more than one string. Locally, i.e., at time t, tightening such strings should then induce a deformation similar to the target motion. Hence, at any given time t, we are looking for a set of strings $\mathbf{s}_i(t)$ such that the linear superposition of the corresponding actuation modes $X_i(t)$ approximates the given deformation field $X(t)$ while minimizing the total number of used strings. To express this goal mathematically, let $\boldsymbol{\lambda}(t) = \big(\lambda_i(t)\big)$ be a vector of weights, one for each mode. To not overload formulas, we drop the parameter t from notation for now since its value is fixed during local considerations.

Given a set of candidate strings $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^m$, we formulate the problem of finding a minimal subset $\mathcal{S}_*$ of $\mathcal{S}$ consistent with the target deformation at time t as

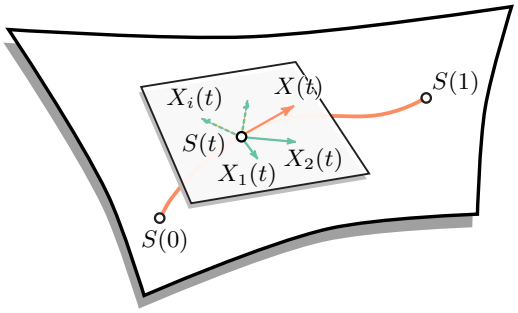

Fig. 3: A folding sequence describes a curve S in the space of developable surfaces. We locally decompose the deformation field $X(t)$ into a set of modes $X_i(t)$, each of which can be realized physically as a contracting thread attached to the surface $S(t)$.

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} \left[\omega \|\boldsymbol{\lambda}\|_0 + \|X - \sum_{i=1}^m \lambda_i X_i\|^2 \right] \tag{1}$$

with

$$\|X(t)\|^2 = \iint_U \|X(t, \mathbf{u})\|_2^2 \, d\mathbf{u} \tag{2}$$

and set $\mathcal{S}_* = \{\mathbf{s}_i \in \mathcal{S} : \lambda_i^* \neq 0\}$. Ways of initializing $\mathcal{S}$ are described later. The ℓ_0-'norm' is used to count the number of active strings. The weight ω controls the trade-off between sparsity of the solution and fidelity. Note that we have the natural constraint $\lambda_i \geq 0$, since strings can only pull but not push. In order to apply numerical optimization techniques to (1), we model the ℓ_0-'norm' by introducing auxiliary variables $\xi_i \in [0, 1]$ as proposed in [Feng et al. 2013]. Enforcing the complementary conditions $\lambda_i \xi_i = 0$, we can replace $\|\boldsymbol{\lambda}\|_0$ in (1) with $\sum_i (1 - \xi_i)$. In other words, the value of $\neg\xi_i$ indicates whether to use string $\mathbf{s}_i$ or not. With those observations problem (1) can now be formulated as

$$\min_{\boldsymbol{\xi}, \boldsymbol{\lambda}} \left[\omega \sum_{i=1}^m (1 - \xi_i) + \|X - \sum_{i=1}^m \lambda_i X_i\|^2 \right] \tag{3}$$

subject to

$$0 \leq \xi_i \leq 1, \quad \lambda_i \xi_i = 0, \quad 0 \leq \lambda_i, \quad i = 1, \ldots, m.$$

The binary selection vector $\neg\boldsymbol{\xi} = (\neg\xi_i)$ tells us which strings of $\mathcal{S}$ to pull to get from $S(t)$ to the close-by shape $S(t + \Delta t)$.

Global solution. Our goal is to identify a set of strings that can drive the whole folding motion. Independently computing solutions at different times t_j, $j = 1, \ldots, n$, as outlined in the previous paragraph, will not yield a globally minimal number of strings, since the optimization is not coupled across time in any way. Nevertheless, the idea behind formulation (3) can be extended to achieve such a coupling. Let $X_{ij} := X_i(t_j)$ be the deformation field induced by $\mathbf{s}_i(t_j)$, $\lambda_{ij} = \lambda_i(t_j)$ its weight and

$$X_j := \frac{\partial S}{\partial t}(t_j), \qquad j = 1, \ldots, n$$

the target deformation field at time t_j. By construction, the coefficient vector $\boldsymbol{\lambda}$ as well as the selection vector $\boldsymbol{\xi}$ depend on time. This implies that a string can appear/disappear during folding, which is physically not possible. Hence, the proper set of time independent

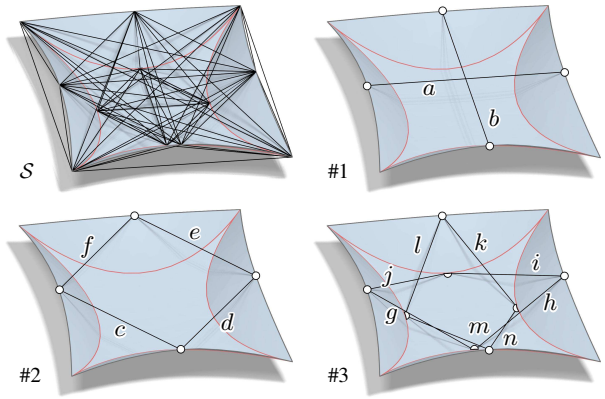

Fig. 4: Input strings $\mathcal{S}$ to global optimization (top left) and three solutions obtained by iteratively removing a discovered solution $\mathcal{S}_*$ from $\mathcal{S}$ and then rerunning the optimization.

complementary conditions is given as

$$\xi_i \lambda_{ij}, \quad i = 1, \ldots, m, \;\; j = 1, \ldots, n, \tag{4}$$

which means that there is only one selection variable ξ_i for all strings $s_i(t_j)$, $j = 1, \ldots, n$, making the vector $\boldsymbol{\xi}$ time independent. Letting $\boldsymbol{\xi}$ and $\boldsymbol{\lambda}$ be the vectors that result from concatenating all the ξ_i and λ_{ij}, leaves us with a total of $m + nm$ variables. The global problem can now be formulated as

$$\min_{\boldsymbol{\xi},\boldsymbol{\lambda}} \left[\omega \sum_{i=1}^{m} (1 - \xi_i) + \sum_{j=1}^{n} \| X_j - \sum_{i=1}^{m} \lambda_{ij} X_{ij} \|^2 \right] \tag{5}$$

subject to

$$0 \leq \xi_i \leq 1, \;\; \lambda_{ij} \xi_i = 0, \;\; 0 \leq \lambda_{ij}, \;\; i = 1, \ldots, m, \, j = 1, \ldots, n.$$

Figure 4 shows the result of global optimization for the sequence S shown in Figure 2 using $n = 3$ poses $S(0.25)$, $S(0.5)$, $S(0.75)$ and the initial set $\mathcal{S}$ of $m = 66$ strings shown in the upper left corner. The solution $\mathcal{S}_* = \{a, b\}$ of (5) corresponding to $\omega = 0.1$ are the two strings shown in #1. Note that solutions will only grow when reducing ω since the best strings are picked first by the least squares fitting term. In order to further explore the space of solutions we reran the optimization with $\mathcal{S}' = \mathcal{S} - \mathcal{S}_*$ as initial set of strings, i.e., we removed the current solution from the search space. Optimization yields the solution shown in #2. A final run with $\mathcal{S}'' = \mathcal{S}' - \mathcal{S}'_*$ results in the strings shown in #3. The multitude of extracted solutions illustrates the non-uniqueness of a valid stringing. The bottom row of Figure 2 shows the string actuated folding motion corresponding to solution #1, Figure 5 shows the corresponding physical rigging.

A note on symmetry. Note that we obtain symmetric solutions without explicit symmetry terms in (5). This behavior, favoring symmetric solutions, was also observed in later experiments (Section 5). Since this is only empiric evidence, formulation (5) can be explicitly biased towards symmetric solutions by adjusting the selection vector $\boldsymbol{\xi}$ appropriately: Once symmetries in the pattern are detected, using e.g. [Mitra et al. 2006], $\boldsymbol{\xi}$ is contracted such that all symmetric modes use the same entry of $\boldsymbol{\xi}$ as their selection variable. Intuitively this means that using a mode that is symmetric to an already active mode comes at no cost.

Initialization. In order to solve (5) we need to specify the set $\mathcal{S}$ of candidate strings. Since the optimization is responsible for the actual selection of strings, we only need to generate a reasonable sampling of all possible strings. We refer to the endpoints of a string as actuation points. While it is possible to use a uniform sampling $\mathcal{A}$ of the sheet as actuation points and then consider the set of all such pairs as candidate strings $\mathcal{S}$, it is more efficient to start with a more conservative choice that is motivated by practical considerations.

We distinguish between three types of potential actuation points: (i) crease curve intersection points, (ii) arbitrary points on crease curves, and finally (iii) all surface points. The main difference among these points is the amount of force that can be applied before unwanted wrinkles form and the paper starts to crumple. Arbitrary surface points are the most undesirable choice as the surface is prone to buckle when applying force at such a point. Hence we restrict the search for actuation points to types (i) and (ii) only. In practice, we obtain actuation points by sampling crease curves according to a prescribed sampling strategy. In the examples shown in the paper, we worked with equidistant crease sampling and iterative midpoint sampling. Further, we always included crease endpoints and intersection points in the sampling.

Given the set $\mathcal{A}$ of actuation points we set $\mathcal{S} = \mathcal{A} \times \mathcal{A}$. Figure 4 shows the input strings $\mathcal{S}$ when sampling crease curves at their midpoints. Since we know the target deformation, the set $\mathcal{S}$ can be further pruned. Specifically, during folding some strings might intersect the surface and hence become invalid. We remove all such invalid strings before computing a solution to (5). Note that strings may also collide with each other during folding. Although this can be detected before running the optimization we do not remove such strings from $\mathcal{S}$ since this would restrict the search space unnecessarily. Instead we check the extracted solutions for such problem and discard solutions if necessary. Most of the times one can get around string-string collision by correctly layering the strings during rigging, although we do not explicitly model this in our optimization. The set $\mathcal{S}$ is conveniently stored as a graph $\mathcal{G}$ with vertex set $\mathcal{A}$. The value of ξ_i can then be interpreted as a binary edge weight.

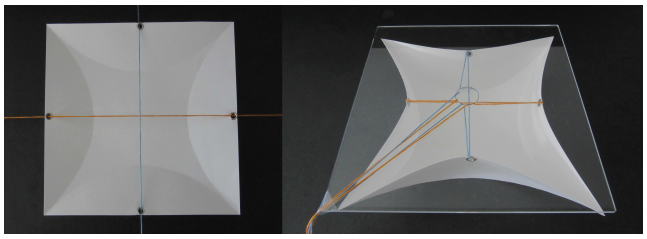

Fig. 5: Rigging a CP. Holes are punched at actuation point locations. All threads are collected at an anchor point (hole in the plexiglass pane). Actuation point locations may be reinforced with metal eyelets as shown.

Rigging. While the vector $\boldsymbol{\xi}$ gives a 'yes/no' answer on whether to use a string at all, the vector $\boldsymbol{\lambda}_j = (\lambda_{ij})_{i=1}^m$ tells us about the relative importance of a string $\mathbf{s}_i$ at time t_j. This information can be used to define a global schedule of string activation, i.e., when and how much to pull a given string. One option is to seek multiple actuations to pull the strings individually at different rates, but would require an elaborate mechanical setup. Instead we aim for the minimal number of loose thread ends a user has to pull without thinking of any scheduling. Although we aim for a minimal number of strings, using a separate physical thread for each actuation mode can be cumbersome in practice. In order to reduce the number of such threads we connect adjacent strings to obtain a set of loops $\mathcal{L} = \{L_k\}$. Each such loop is defined as an ordered set of strings and realized by a single thread traversing the strings in

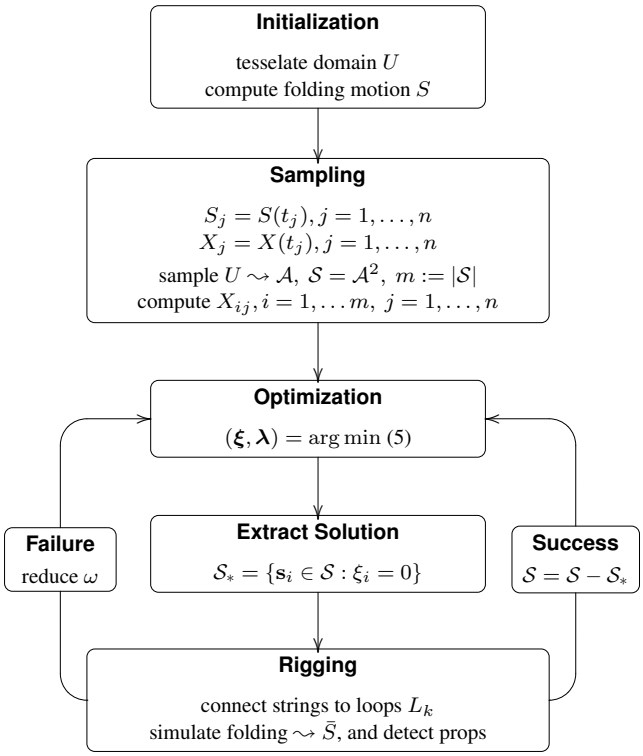

Fig. 6: Main steps to compute a stringing that drives the folding of a planar sheet according to a given crease pattern. The optimization loop terminates once a given number of solutions is extracted or after repeated failure.

the given order. Referring to Figure 4 solution #1 consists of two loops $L_1 = \{a\}$ and $L_2 = \{b\}$, each comprising of one string. Solution #2 forms a single loop $L_1 = \{c, d, e, f\}$. Solution #3 consists of two loops $L_1 = \{g, h, i, j\}$ and $L_2 = \{k, l, m, n\}$, each of which comprises of four strings. Note that a loop does not need to be closed in the sense that it starts and ends at the same actuation point, i.e., the polyline of strings can be open. All threads are collected at anchor points, any open loop is closed at such an anchor, see Figure 5. We extract loops by tracing strings, starting at an actuation point until all actuation points are visited. At actuation points of valence greater than two the successor string is chosen randomly among all adjacent strings that have not been visited. Whenever the folded shape sits on a plane we use a single anchor point located beneath the model, realized as a hole in that plane. For better visualization we used a plastic tube to collect the strings in Figures 1 and 10. The tube is not moving when pulling and its tip acts as anchor point.

Defining the actuation mode of a loop is straight forward and can be handled seamlessly in our optimization. Nonetheless we do not consider loops during global optimization to avoid combinatorial explosion (we would have to consider all possible loops contained in the initial set $\mathcal{S}$) and rather introduce them in a post processing step. As mentioned earlier, when combining strings into loops we can no longer exploit the know actuation weights λ_{ij}. Instead we check in a simulation step whether the surface's resistance to bending can be used for implicit scheduling as described next.

Simulation. Since the rigging, including the loops, was derived from local analysis, we next simulate the induced string driven folding motion to validate the solution. If the simulation fails, the sparsity weight ω in (5) is reduced to allow for more strings. We determine success of folding visually. The decision process could be automated by computing the Hausdorff distance [Hausdorff 1957] between shapes of the target motion and the simulated folding motion at corresponding times and rejecting a solution if a given threshold on the shape distance is exceeded. During simulation, instead of considering lengths of each individual string, we consider the total length l_k of each loop $L_k \in \mathcal{L}$ since we do not have control over those lengths when realizing a loop as a single thread. Simulation is driven by prescribing the length of each loop. Specifically, starting from the actual length l_k^0 of a loop in the unfolded position, we model its contraction with a shortening parameter Δl_k. The prescribed length is iteratively defined as $l_k^i = l_k^{i-1} - \Delta l_k$. In the shown examples we set $\Delta l_k = 0.01 \cdot l_k^0$, i.e., the length of a loop is reduced by 1% of its initial length in each iteration. The value of Δl_k roughly corresponds to how fast the corresponding thread is pulled. Figure 1 shows simulation results as well as actual string based folding where all the strings are realized as individual threads.

Props. Simulation of string based folding results in a folding sequence $\bar{S}(t)$. We can compare the length progression of a string $\mathbf{s}_i \in \mathcal{S}_*$ in both sequences by looking at the trajectories of the line segments $[S(t, \mathbf{u}_i), S(t, \mathbf{v}_i)]$ and $[\bar{S}(t, \mathbf{u}_i), \bar{S}(t, \mathbf{v}_i)]$. To faithfully reproduce the target deformation S the progression of corresponding string lengths should be roughly equal in both deformations. If, relative to the deformation $\bar{S}(t)$, the length of a string $\mathbf{s}_i$ becomes shorter than it should be according to $S(t)$, we need to realize a length constraint for the thread segment that corresponds to $\mathbf{s}_i$. For manual actuation we only compare string lengths at time $t = 1$. In this simplified situation a length constraint can be realized by enclosing the corresponding thread segment in a thin plastic tube during the physical rigging process that prevents the segment from becoming too short, see Figure 8 for an example. Note that this is different from slack that might accumulate in a string segment. As long as segments do not become too short in the final pose, slack is automatically reeled in during actuation.

Finally, Figure 6 gives a high level overview of the processing pipeline described in this section. Details on mode computation, optimization and simulation can be found in Section 6.

5. RESULTS

We tested our algorithm on a number of freely available crease patterns. After computing the strings we used paper and aluminum sheets to realize the models. In the case of paper the crease curves were either scored manually or by using a laser cutter. In either case, they are manually precreased after scoring. Precreasing means folding each crease once according to fold orientation and then unfolding again. Since precreasing is done one fold at a time, no coordinated folding of several creases is required. While practitioners do this to achieve high folding accuracy, weakening the paper along creases in this way is necessary to facilitate foldability in our setting. Preparing the CP in this way was not possible for the aluminum sheets due to material fatigue. Aluminum can only be brought into a folded shape once; further interaction, such as unfolding, splits the sheet along crease curves. For this reason we laminated the aluminum sheet after scoring and before precreasing.

Actuation points are realized by punching holes in the CP. Eyelets can be used to reinforce actuation points, reduce the wear on the sheet and increase the durability of the model, see Figure 5. Threads were realized using colored floss and nylon twine. While

we did not experience problems with friction, the use of eyelets makes actuation very smooth.

Please note for all the figures that the meaning of mountain and valley folds is reversed when viewing the sheet from below. It is only important that all red curves fold in the same direction and all blue curves fold the other way – absolute fold orientation is not essential. If not mentioned otherwise we used $n = 10$ poses along the target folding motion.

Asimov Chair. The Asimov chair concept (Figure 8) is based on a CP designed by Benjamin Spöth and owned by Joris Laarman Lab. The top row of Figure 8 shows the set $\mathcal{S}$ of 210 initial strings as well as two solutions that were extracted using the iterative procedure described in Section 4. In #1 we have two loops $L_1 = \{a\}$ and $L_2 = \{b\}$, each consisting of a single string; while in #2 all strings are connected to form a single loop $L = \{c, d, e, f, g\}$. The second row shows the target shape $S(1)$ as well as the final poses of simulated string actuated folding. Comparing the target shape with the simulated results reveals rather large deviations. The simulation results are confirmed by rigging the CP accordingly: For solution #1 the blue and orange thread were collected in front of the chair in a plastic tube that acts as an anchor point (this is not a prop), in case of #2 threads were collected at the point marked A at the base of the chair. Comparison of string lengths indicates that the ratio of lengths of strings a and b of solution #1 is different from the target value. Strings c, g of solution #2 became too short in comparison with the lengths of strings e, d, f. Since #1 consists of two separate loops we can either insert props or just stop pulling once the proper lengths are reached. The result when inserting props of the proper length is shown in Figure 8.1. For #2 we added props of the appropriate length around thread segments c and g. Corresponding folding results are shown in Figure 8.2 and 8.3. Finally, Figure 7 shows the full folding simulation according to solution #2. Computationally props are realized as length constraints on the strings c and g. The bottom row shows snapshots along the string actuated folding motion using the props shown in 8.3.

Apricot. The Apricot CP is based on a design by Jun Mitani. Figure 9 shows the computed target deformation (top row). The initial set $\mathcal{S}$ of 120 strings and three solutions are shown in Figure 11. Note that the final shape of stringing #3 is given by $S(0.75)$. At this time the flaps touch and the distance of actuation points cannot be shortened any more by pulling the strings without damaging the sheet. The reason for considering stringing #3 is explained below. Each solution can be realized as a single loop $L = \{a, b, c, d, e\}$. Figure 10 shows a simulation of string actuated folding when using a single thread to realize solution #1. The simulated behavior is reproduced by the physical model. This rather large deviation from the target deformation illustrates the effect of using a single thread without control over individual string lengths versus using a separate thread for each string. Using one thread for each string results in the expected deformation behavior, see Figure 1. To make the CP fold such that the base of the apricot ends up sitting on the tabletop as shown in Figure 9 we use the stringing as shown in the top right images of Figure 11, i.e., solution #1 (orange) and solution #3 (blue) together. This is necessary because of the interaction of the CP with the tabletop. The actuation points of solution #3 need to slide on the table in order for the flaps to move underneath the model. This cannot be achieved with solution #1 or #2 since the actuating thread will slide on the table when pulling downwards and the points A, B, C, D, E move well below this level during folding (Figure 11, third row), i.e., the tabletop blocks the CP from folding. We use solution #3 to make the points A, B, C, D, E slide on the table and #1 to finish the folding process.

Nautilus. The Nautilus shape, see Figure 12, is based on a sphere CP, i.e., a pattern that folds to a spherical shape with parts bordered by curved folds mapping to globe gores. The rectangular sphere CP is warped to a trapezoid as shown in Figure 13. The warped pattern folds to a spiral shape instead of a sphere. This CP contains a lot of repetition, hence, for efficient computation, we analyzed the deformation behavior on a section of 6 consecutive segments, where a segment is defined as the region between two consecutive mountain folds, see Figure 13. The complete CP (Figure 12) consists of 15 such segments. The target deformation of such a section of the CP is shown in Figure 13. The set $\mathcal{S}$ of 325 candidate strings is shown in Figure 14 together with the extracted solution. Note, that we show the strings in a folded pose since a lot of them would be hard to see in the unfolded position as they are parallel to edges of the CP. The strings connecting the end points of valley folds (blue) as proposed in solution #1 are not fit for actual fabrication, considering that the complete CP is a paper strip about 2 meters long. We removed those long edges from $\mathcal{S}$ to arrive at solution #2 that can easily be adapted to any number of segments. Figure 13 shows the string driven folding result employing solution #2. All strings a_i and $\bar{a}_i$ have been realized as individual threads whereas the strings b_i and $\bar{b}_i$ are connected to

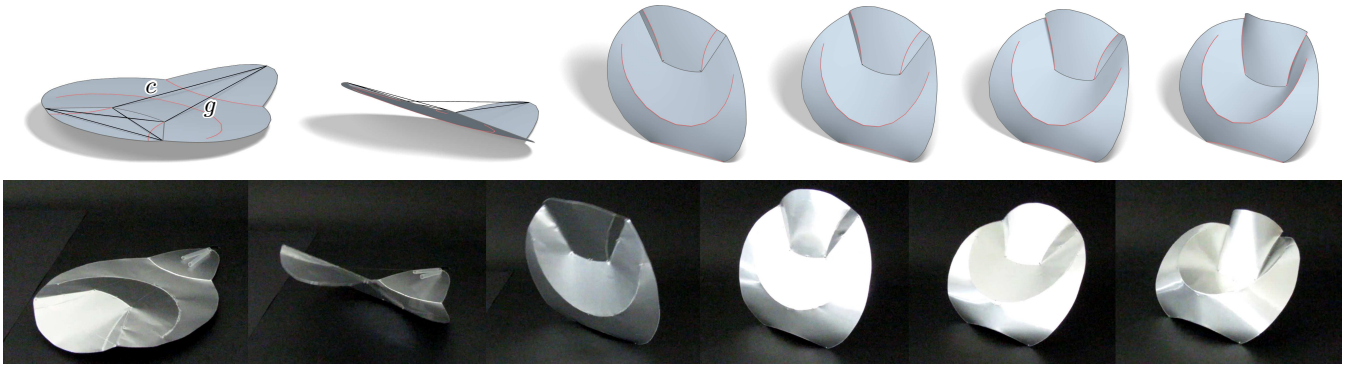

Fig. 7: The top row shows the result of simulated string actuated folding employing a length constraint on strings c and g. We realized those length constraints by enclosing the corresponding thread segments in plastic tubes of the appropriate length. The folding result without props is shown in Figure 8, third row, right.

form two loops $L = \{b_1, b_2, \dots\}$ and $\bar{L} = \{\bar{b}_1, \bar{b}_2, \dots\}$. The complete model (Figure 12) was rigged using a total of 4 threads corresponding to the loops $\{a_1, a_2, \dots\}$, $\{\bar{a}_1, \bar{a}_2, \dots\}$, $\{b_1, b_2, \dots\}$, and $\{\bar{b}_1, \bar{b}_2, \dots\}$. Using only 4 threads we cannot ensure uniform contraction of strings. This partly explains the difference in target deformation and string folded deformation. Additionally, such a model has a considerable self weight, even if only made of paper.

6. IMPLEMENTATION DETAILS

In the following sections, we present our approach to discretize the concepts introduced in Section 4.

6.1 Global Optimization

Presenting problems (3) and (5) directly to a constrained solver requires a good initialization of the λ_{ij} (one can set $\xi_i = 0$). Since such an initialization is not available, considering a sequence of relaxed problems provides good convergence properties (see [Feng et al. 2013]). To this end, we replace the complementary conditions $\xi_i \lambda_{ij} = 0$ with

$$-\varepsilon \leq \xi_i \lambda_{ij} \leq \varepsilon \tag{6}$$

with $\varepsilon \geq 0$ and gradually reduce the value of ε starting from 0.1. Since the actual values of λ_{ij} are not used, it is much easier to interpret the parameter ω in (5) when all the deformation fields are normalized. Intuitively, normalized modes correspond to uniformly contracting strings. In that case the maximum approximation error for each time t_j is $\|X_j\| = 1$ and the maximum global approximation error equals n. Hence, depending on the desired sparsity of the solution we used values of $\omega \in (0, 1]$.

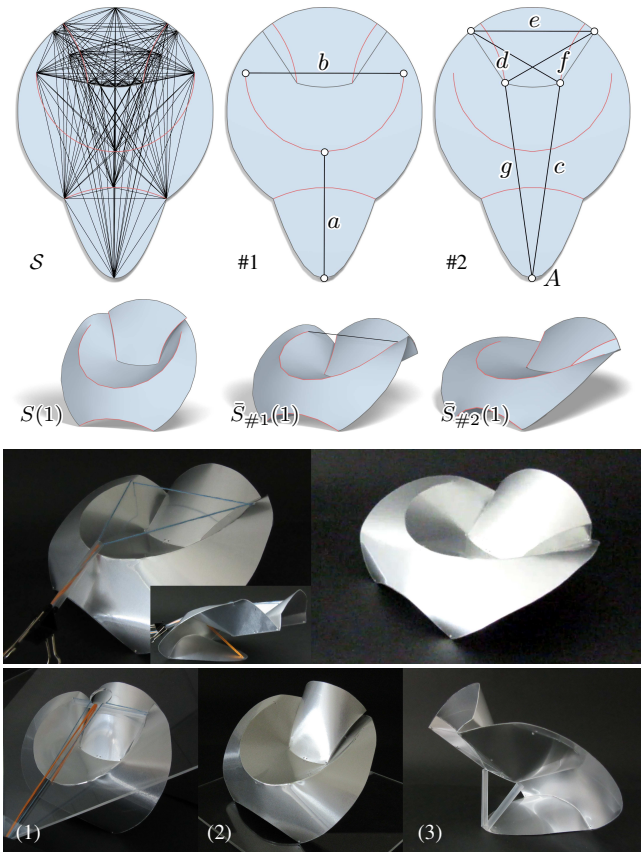

Fig. 8: Initial string graph $\mathcal{S}$ for the Asimov chair concept and two solutions. Starting from the left, the second row shows the target shape and the simulated string driven folding results using #1 and #2, respectively. Actual folding results confirm the simulation (third row). The fourth row shows final folding results and how props are used for the stringing of #1 and #2.

6.2 Deformation Model

Isometric deformations of triangle meshes are characterized by rigidly transforming triangles. Such deformations are referred to as as-rigid-as-possible transformations [Sorkine and Alexa 2007]. We start from a planar mesh with faces Δ_k and define isomorphic 3-dimensional shapes in terms of rigid face transformations T_k. Transformed vertex coordinates are recovered by averaging as

$$\mathbf{p}_i = \frac{1}{n_i} \sum_{k:i \in \Delta_k} T_k\big(\mathbf{p}_i^k\big), \quad n_i = |\{k : i \in \Delta_k\}|. \tag{7}$$

In the following sections, any triangle related properties, such as vertices, normals, edges, etc. that carry a superscript k refer to properties of the triangle Δ_k.

To penalize the distortion of transformed triangles we want the $T_k\big(\mathbf{p}_i^k\big)$ in (7) to be as close as possible before averaging. We achieve this by penalizing the gap between neighboring triangles Δ_i and Δ_j. We parametrize edges as $\mathbf{e}^i(u) = (1-u)\mathbf{p}^i + u\mathbf{q}^i$ and let

$$A_{ij} = \int_0^1 \Big(T_i(\mathbf{e}^i(u)) - T_j(\mathbf{e}^j(u))\Big)^2 du, \tag{8}$$

which can be considered as the energy stored in an elastic strip that joins both triangles. Deformations that minimize the weighted sum $\sum a_{ij} A_{ij}$ subject to a sparse set of boundary conditions on some transformations T_k result in as-rigid-as-possible deformations without any smoothness properties. This idea can be generalized to model the bending energy of a sheet by enclosing the faces of the initially flat mesh within a thin layer of prisms, see [Botsch et al. 2006]. Each face of the mesh gives rise to such a prism by extruding it along its normal (in both directions), see Figure 15. The corresponding face transformation acts on the face's prism, introducing a gap between prism faces that were perfectly aligned in the planar state. To measure this gap, we parameterize a prism face along the edge $(\mathbf{p}^i, \mathbf{q}^i) \in \Delta_i$ as

$$\begin{aligned} F^i(u,v) = (1-v)\Big((1-u)\mathbf{p}^i_- + u\mathbf{q}^i_-\Big) \\ + v\Big((1-u)\mathbf{p}^i_+ + u\mathbf{q}^j_+\Big) \end{aligned} \tag{9}$$

with $\mathbf{p}^i_\pm = \mathbf{p}^i \pm h\mathbf{n}^i$, $\mathbf{n}^i$ the normal of Δ_i and h the height of the prism. Analogous to (8) we define

$$E_{ij} = \iint_{[0,1]^2} \Big(T_i(F^i(u,v)) - T_j(F^j(u,v))\Big)^2 du dv. \tag{10}$$

The value of E_{ij} can be interpreted as the energy stored inside an elastic material that joins adjacent prism faces. The energy $E = \sum w_{ij} E_{ij}$ penalizes both stretching and bending. For our application, penalizing stretching is more important while we use bending mainly as a regularizer. For this reason we use

$$f_\lambda\big((T_k)\big) = \sum a_{ij} A_{ij} + \lambda \sum w_{ij} E_{ij}, \quad \lambda \in [0, 1) \tag{11}$$

as our global surface energy functional. We use the values proposed in [Botsch et al. 2006] for a_{ij} and w_{ij} along with $\lambda = 0.1$.

We consider two main drivers during folding: (i) fold angles along crease curves, when computing the target deformation S; and (ii) by prescribing the length of a thread that is attached to certain surface points during simulation $\bar{S}$. Both formulations use (11) as surface bending energy. For angle-based folding, we modify the prism face defining normal vectors along crease edges, while for string actuation we add a thread length term to (11). The effect of precreasing, i.e., weakening the sheet along creases, is modeled by reducing the value of h at crease edges of the mesh.

Quadratic optimization. Given a sparse set of constrained faces or vertices, we compute the per-face transformations T_k by converting the non-linear problem of minimizing

$$f_\lambda\big((T_k)\big) = \sum a_{ij} A_{ij} + \lambda \sum w_{ij} E_{ij} \tag{12}$$

to a sequence of quadratic problems using instantaneous kinematics [Pottmann et al. 2002]. Specifically, in each iteration, we approximate the action of T_k in the next iteration as,

$$T_k^{i+1}(\mathbf{p}) \approx T_k^i(\mathbf{p}) + \mathbf{c}_k^i \times T_k^i(\mathbf{p}) + \bar{\mathbf{c}}_k^i. \tag{13}$$

Substituting (13) into (8) and (10) converts (12) into a quadratic function of the form $f_\lambda^i\big((\mathbf{c}_k^i, \bar{\mathbf{c}}_k^i)\big)$ in the unknowns $(\mathbf{c}_k^i, \bar{\mathbf{c}}_k^i)$, which are computed by solving the linear system $\nabla^2 f_\lambda^i(0) \cdot \mathbf{c} = -\nabla f_\lambda^i(0)$. Having solved for $(\mathbf{c}_k^i, \bar{\mathbf{c}}_k^i)$, we compute a corresponding helical motion R_k^i [Pottmann and Wallner 2001] and update the face transformation as $T_k^{i+1} = R_k^i \circ T_k^i$.

Computing a folding sequence. To model the process of folding along crease edges we use the prism-based deformation model described above to prescribe fold angles in a least squares fashion. Let $\mathbf{e}^i = (\mathbf{p}^i, \mathbf{q}^i) \in \Delta_i$ be oriented such that Δ_i is to the left of the edge, and $\mathbf{x}^i$ be the unit vector parallel to $\mathbf{q}^i - \mathbf{p}^i$ of the same orientation. We define the crease angle across $\mathbf{e}^i$ with respect to the coordinate system $(\mathbf{n}^i \times \mathbf{x}^i, \mathbf{n}^i)$ by replacing $\mathbf{n}^i$ with

$$\mathbf{n}^i(\alpha) = \cos(\alpha)\mathbf{n}^i - \sin(\alpha)(\mathbf{n}^i \times \mathbf{x}^i) \tag{14}$$

when evaluating (9). The modified normal defines a mountain (valley) fold for positive (negative) values of α, see Figure 15. In practice, we use a more symmetric approach and incline the normal vectors on both sides of the edge by $\alpha/2$. In all the presented examples, we started from $\alpha = 0$ and increased α in 1° steps.

While other thin shell deformation models like [Grinspun et al. 2003] can be used, we chose to base our implementation on [Botsch et al. 2006] because fold angles and varying paper thickness – needed to model precreasing – can easily be integrated.

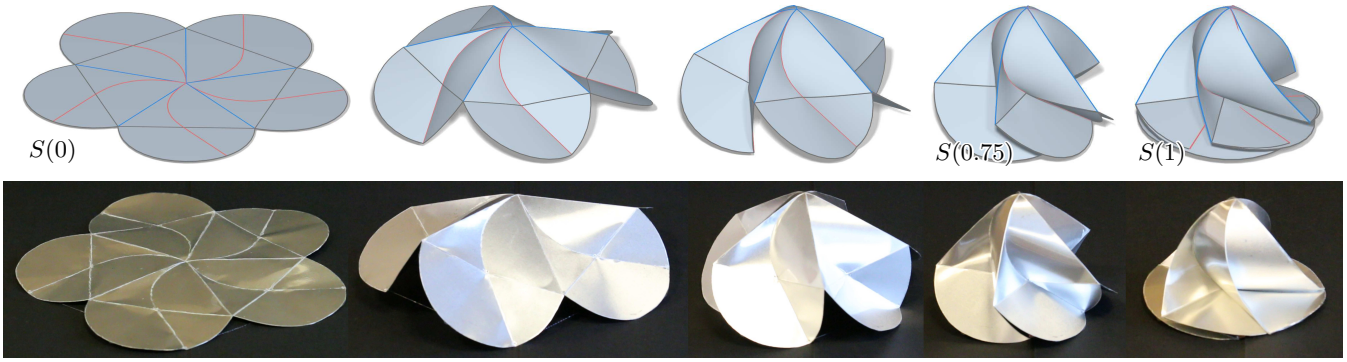

Fig. 9: Folding the Apricot. The target deformation is shown in the top row. String acuated folding (bottom row) employs solutions #1 and #3 as shown in the top right images of Figure 11. The two nylon threads are pulled downwards through a hole in the tabletop.

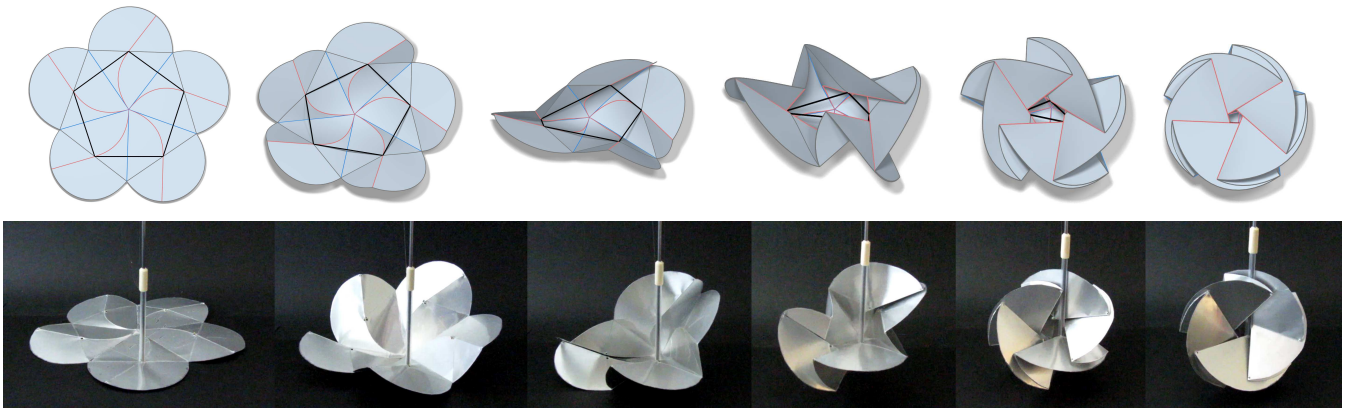

Fig. 10: Simulation of string actuated folding and corresponding physical deformation. We use a single thread, i.e., one loop, to realize solution #1. Folding is not only guided by the thread but also by the interaction of the CP with itself until it locks into place. Compare with Figure 1 where we used 5 loops, i.e., one thread per string of solution #1.

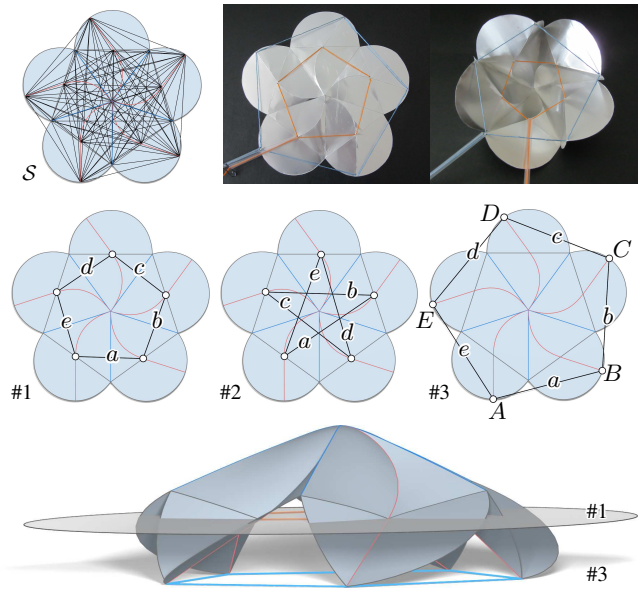

Fig. 11: Initial string graph (top left) and three solutions. In all cases the stringing is given as $\{a, b, c, d, e\}$. The top right shows the rigged CP and the partially folded CP viewed from below. The last row shows the different planes in which the strings of solution #1 and #3 move during folding.

String actuated deformation. We encode a stringing as a graph $\mathcal{G}$. Edges of $\mathcal{G}$ that define the loop L_ν are labeled with the integer ν. Extracting those edges gives rise to a subgraph $\mathcal{G}_\nu$. The length of the loop L_ν is then given as the sum of the edge lengths of $\mathcal{G}_\nu$. We achieve shortening of the corresponding thread by prescribing a length value l_ν. However, instead of setting thread lengths as equality constraints, we use the terms

$$L_{\mathcal{G}_\nu}\big((T_k)\big) = \Big(\big(\sum_{(i,j)\in\mathcal{G}_\nu} \|\mathbf{p}_i - \mathbf{p}_j\|\big) - l_\nu\Big)^2. \qquad (15)$$

Note that in contrast to (11) the expression $L_{\mathcal{G}_\nu}$ is not quadratic when substituting (13). Therefore we replace it with its second order Taylor expansion. This results in a system of the form,

$$\big(\nabla^2 f^i_\lambda(0) + \nabla^2 L^i_{\mathcal{G}_\nu}(0)\big)\mathbf{c} = -\big(\nabla f^i_\lambda(0) + \nabla L^i_{\mathcal{G}_\nu}(0)\big) \qquad (16)$$

that is solved in each iteration. We use IPOPT [Wächter and Biegler 2006] with PARDISO [Kuzmin et al. 2013] as solvers.

Mode computation. Given a partially folded CP, i.e., face transformations (T_k) and actuation points $\mathbf{p}_i$, $\mathbf{q}_i$ we compute the corresponding mode by considering the string length term

$$L_i\big((T_k)\big) = \Big(\|\mathbf{p}_i - \mathbf{q}_i\|\big) - l_i\Big)^2,$$

where $l_i = \|\mathbf{p}_i - \mathbf{q}_i\| - \Delta l_i$ is the new target length and Δl_i the amount of length reduction. The deformation mode X_i is obtained as the solution of the corresponding linear system (16).

Handling self-collision. We detect self collisions during folding as follows. If a collision occurs, i.e., a vertex $\mathbf{p}_i$ penetrates a face Δ_k we add a barrier hyperplane defined by the plane of the triangle Δ_k. In terms of triangle transformations this plane corresponds to the xy-plane under the transformation of Δ_k. With $\mathbf{b}^k$ the barycenter of Δ_k, we add one of the following linear inequality constraints

$$\mathbf{n}^k \cdot (\mathbf{p}_i^j - \mathbf{b}^k) \begin{cases} > 0 \\ < 0 \end{cases} \qquad (17)$$

for each $\mathbf{p}_i^j$ in the right hand side of (7) depending on which side of Δ_k the vertex $\mathbf{p}_i$ was before collision. Note that $\mathbf{n}^k$ and $\mathbf{b}^k$ get transformed by T_k (respectively $(\mathbf{c}_k, \bar{\mathbf{c}}_k)$) like vertices of Δ_k, cf. (13). Since $\mathbf{n}^k$ is a vector only the rotation part of T_k (represented by $\mathbf{c}_k$) is relevant.

6.3 Meshing

We use a triangle mesh as the underlying representation of the deforming sheet. The crease curves defined by the CP are represented as edge polylines of this mesh. We initialize the planar mesh by computing a constrained Delaunay triangulation of the set $\mathcal{C} = \{C_i\}$ of crease curves [Shewchuk 1996]. The straight line graph that is used as input is generated by equidistantly sampling the curves C_i according to some sampling distance ε. We use ε as a hint for the desired edge length throughout the mesh. After computing the constrained Delaunay triangulation, we run a remeshing step that terminates once all the edge lengths conform to the length hint. Specifically, we encode the edge length information as a sizing field [Bossen and Heckbert 1996] in the form of symmetric positive definite matrices $Q_\mathbf{v}$ assigned to vertices $\mathbf{v}$ of the mesh. The matrices $Q_\mathbf{v}$ are update during deformation using curvature information (see e.g. [Narain et al. 2012]) and remeshing is applied as described in the next paragraph.

Remeshing. Given the $Q_\mathbf{v}$ we employ the remeshing algorithm described in [Bossen and Heckbert 1996] that consists of three basic steps: (i) vertex relocation, (ii) vertex deletion, and (iii) edge splits. After each basic operation the local neighborhood of the affected vertex is retriangulated using the Delaunay edge flip condition relative to the sizing field. Sizing field values for newly inserted vertices are computed as an average of neighboring values. Additionally, we take special care at the crease edges and crease vertices: A crease edge may never be flipped. Further, if a crease edge is split, the new vertex is projected onto the crease. Crease vertices are allowed to move only along the corresponding crease, while crease intersection points are not allowed to move at all.

Figure 16 shows the change over time of the sizing field and the corresponding mesh for the example of Figure 2. Unit balls with respect to the metric defined by $Q_\mathbf{v}$ are shown as ellipses. The mesh stays isotropic in the center, which remains planar during deformation, while all other triangles change their shape and orientation according to curvature. In this way we maintain mesh quality while avoiding unnecessarily dense meshes.

7. CONCLUSION

We presented string actuated curved folded surfaces as crease patterns rigged with networks of strings, which when pulled lift the planar sheets to freeform shapes. The results are computational mechanisms to facilitate folding of crease patterns, a process that has so far been largely restricted to manual folding. This opens up possibilities to realize movable lightweight large-scale freeform surfaces without requiring complex manual or robotic actuation. As key technical enabler, we introduced the notion of string actuation modes, and used them to select actuation points and decide how to string these points together.

Limitations. Not every curved folded shape is foldable using strings. Depending on the folding motion there might be no set of

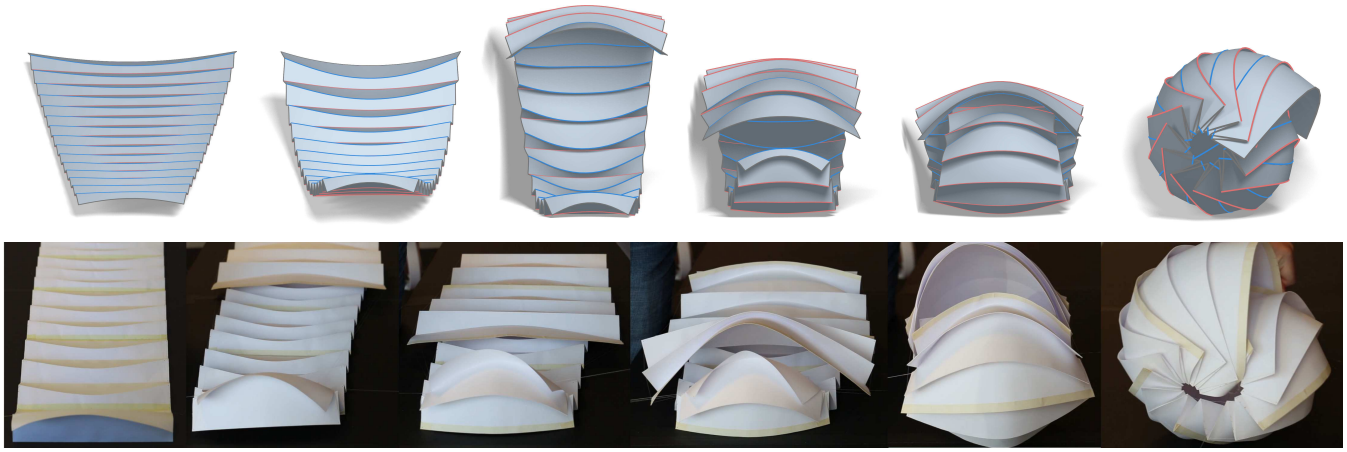

Fig. 12: Target deformation of the Nautilus CP (top row) and string driven folding motion induced by extending solution #2 (Figure 14) of a 6 segment section to the complete pattern consisting of 15 segments. The paper model is made from 5 sheets of A3 drawing paper. Each sheet holds 3 segments, sheets are joined along the boundary mountain folds using masking tape (yellow).

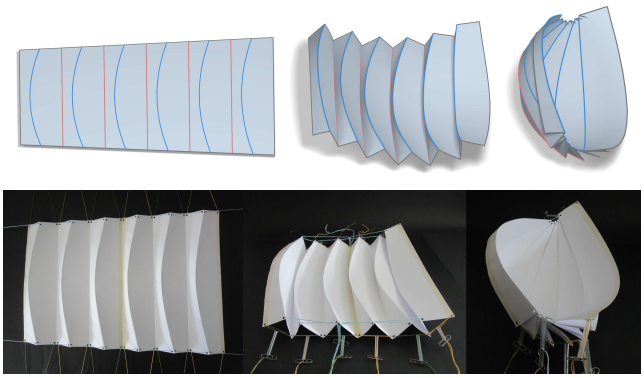

Fig. 13: Target folding motion for a 6 segment section of the Nautilus crease pattern. The bottom row shows a rigged paper model of the pattern.

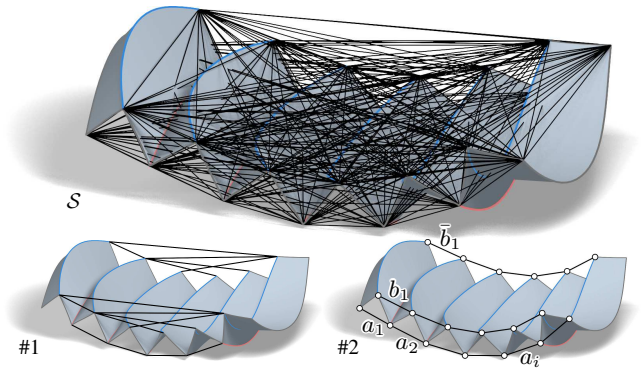

Fig. 14: Initial set $\mathcal{S}$ of strings for the Nautilus CP and the computed solutions. Please refer to the text for details.

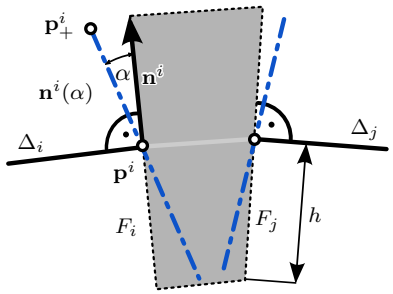

Fig. 15: Notation for surface energy and crease angles. In the above α defines a valley fold once prism faces are aligned.

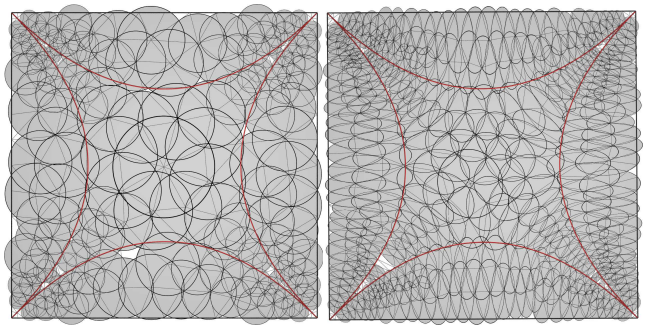

Fig. 16: Visualizing the sizing field used during remeshing as ellipses, which correspond to the unit ball in the metric defined by the matrices $Q_{\mathbf{v}}$. The left hand image shows the initial isotropic meshing of the CP at time $t = 0$, while the right hand side shows the mesh at time $t = 1$.

actuating strings that does not intersect the partially folded shape at some point. In other words, such a shape cannot be folded using only tensile forces.

Our simulation does not consider gravity, i.e, the self-weight of the shape. Hence there are no strings to counter balance self weight. This limitation is most notable for the Nautilus (Figure 12) which is the biggest and also heaviest prototype that we built. The inclusion

of weight balancing threads would require a generalization of the concept of anchor points and to enclose the model inside a cage to attach those balancing threads.

Future Work. An interesting future direction will be to consider staged folding sequences, where the flat surface can be physically 'animated' by cycling through a set of predefined 'keyframe' surfaces. A possible approach will be to design a set of string networks that are to be activated in sequential order to progressively create the curved surfaces. Based on recent tiny robotic Origami [Miyashita et al. 2015], it will be fascinating to explore actuation of curved folds using other mechanism like sheets of PVC and laser-cut layers of polystyrene and paper with a magnet inside it. At a large scale, in spirit with efforts from Robofold [Epps 2014], it will be interesting to realize string actuated curved folds using stiffer materials and wires (instead of threads). However, friction then needs to be accounted for explicitly in the computation. Finally, it will be worth exploring multiple actuations to control the strings individually or in bunches, instead of actuating them all together.

REFERENCES

Günter Aumann. 2003. A Simple Algorithm for Designing Developable Bézier Surfaces. *Comput. Aided Geom. Des.* 20, 8-9 (2003), 601–619.

Pengbo Bo and Wenping Wang. 2007. Geodesic-Controlled Developable Surfaces for Modeling Paper Bending. *Computer Graphics Forum* (2007).

Frank Bossen and Paul S. Heckbert. 1996. A Pliant Method for Anisotropic Mesh Generation. In *5th International Meshing Roundtable*. 63–74.

Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. 2006. PriMo: Coupled Prisms for Intuitive Surface Modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing (SGP '06)*. 11–20.

Mario Botsch and Olga Sorkine. 2008. On Linear Variational Surface Deformation Methods. *IEEE Transactions on Visualization and Computer Graphics* 14, 1 (Jan. 2008), 213–230.

Robert Burgoon, Eitan Grinspun, and Zoë Wood. 2006. Discrete Shells Origami. In *Proceedings of Computers And Their Applications*. 180–187.

Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and Fabricating Mechanical Automata from Mocap Sequences. *ACM SIGGRAPH Asia* 32, 6 (2013), 11.

C.-H. Chu and C.H. Squin. 2002. Developable Bézier patches: properties and design. *Computer-Aided Design* 34 (2002), 511–527.

Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM SIGGRAPH* 32, 4, Article 83 (2013), 12 pages.

Erik Demaine and Joseph O' Rourke. 2007. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press.

Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. 2014. Assembling Self-supporting Structures. *ACM SIGGRAPH Asia* 33, 6, Article 214 (2014), 10 pages.

Gregory Epps. 2014. *Made by Robots: Challenging Architecture at a Larger Scale*. Wiley.

Mingbin Feng, John E. Mitchell, Jong-Shi Pang, Xin Shen, and Andreas Wächter. 2013. *Complementary formulations of l_0-norm optimization problems*. Technical Report. Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY.

Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire Mesh Design. *ACM SIGGRAPH* 33, 4, Article 66 (2014), 12 pages.

Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete Shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, 62–67.

Mo Guoliang and Zhao Yanan. 2006. Designing Bézier Surfaces Minimizing the Gaussian Curvature. In *Proc. International Conference on Robotics, Control and Manufacturing Technology*. 271–276.

Felix Hausdorff. 1957. *Set Theory*. American Mathematical Soc.

Behrend Heeren, Martin Rumpf, Peter Schröder, Max Wardetzky, and Benedikt Wirth. 2014. Exploring the Geometry of the Space of Shells. *Computer Graphics Forum* 33, 5 (2014), 247–256.

D.A. Huffman. 1976. Curvature and Creases: A Primer on Paper. *IEEE Trans. Comput.* 25, 10 (1976), 1010–1019.

Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Begault, Laurence Boissieux, and Marie-Paule Cani. 2015. Sketching Folds: Developable Surfaces from Non-Planar Silhouettes. In *ACM Transactions on Graphics*. Article 75.

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J. Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved Folding. In *ACM SIGGRAPH 2008 Papers (SIGGRAPH '08)*. ACM, New York, NY, USA, Article 75, 9 pages.

Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. 2007. Geometric Modeling in Shape Space. *ACM SIGGRAPH* 26, 3 (2007), #64, 1–8.

B. Koo, J. Hergel, S. Lefebvre, and N. Mitra. 2016. Towards Zero-Waste Furniture Design. *IEEE Transactions on Visualization and Computer Graphics* 99 (2016).

A. Kuzmin, M. Luisier, and O. Schenk. 2013. Fast Methods for Computing Selected Elements of the Greens Function in Massively Parallel Nanoelectronic Device Simulations. In *Euro-Par 2013 Parallel Processing*, F. Wolf, B. Mohr, and D. Mey (Eds.). Lecture Notes in Computer Science, Vol. 8097. Springer Berlin Heidelberg, 533–544.

Robert J. Lang. 2011. *Origami Design Secrets: Mathematical Methods for an Ancient Art*. A K Peters.

Xian-Ying Li, Chao-Hui Shen, Shi-Sheng Huang, Tao Ju, and Shi-Min Hu. 2010. Popup: automatic paper architectures from 3D models. *ACM Transactions on Graphics* 29, 4 (2010), 111:1–9.

Meher McArthur. 2013. *Folding Paper: The Infinite Possibilities of Origami*. Tuttle Publishing.

Nicolas Mellado, Peng Song, Xiaoqi Yan, Chi-Wing Fu, and Niloy J. Mitra. 2014. Computational Design and Construction of Notch-free Reciprocal Frame Structures. (2014).

Jun Mitani. 2009. A Design Method for 3D Origami Based on Rotational Sweep. *Computer-Aided Design and Applications* 6, 1 (2009), 69–79.

Jun Mitani. 2012. Column-shaped Origami Design Based on Mirror Reflections. *Journal for Geometry and Graphics* 16, 2 (2012), 185–194.

Jun Mitani and Takeo Igarashi. 2011. Interactive Design of Planar Curved Folding by Reflection. In *Pacific Graphics Short Papers*, Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin (Eds.). The Eurographics Association.

N. J. Mitra, L. Guibas, and M. Pauly. 2006. Partial and Approximate Symmetry Detection for 3D Geometry. *ACM Transactions on Graphics (SIGGRAPH)* 25, 3 (2006), 560–568.

Shuhei Miyashita, Steven Guitron, Marvin Ludersdorfer, Cynthia R. Sung, and Daniela Rus. 2015. An Untethered Miniature Origami Robot that Self-folds, Walks, Swims, and Degrades. In *Proc. Int. Conf. on Robotics and Automation*.

Rahul Narain, Tobias Pfaff, and James F. O'Brien. 2013. Folding and Crumpling Adaptive Sheets. *ACM Trans. Graph.* 32, 4, Article 51 (July 2013), 8 pages.

Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Transactions on Graphics* 31, 6 (Nov. 2012), 147:1–10. Proceedings of ACM SIGGRAPH Asia 2012, Singapore.

Yusuke Obuchi and others. 2013. *99 Failures Pavilion*. Technical Report. Digital Fabrication Lab, The University of Tokyo.

Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. 2015. Design and Fabrication of Flexible Rod Meshes. *ACM SIGGRAPH* 34, 4, Article 138 (2015), 12 pages.

Martin Peternell. 2004. Developable Surface Fitting to Point Clouds. *Comput. Aided Geom. Des.* 21, 8 (2004), 785–803.

Helmut Pottmann, Philipp Grohs, and Niloy J. Mitra. 2008. Laguerre Minimal Surfaces, Isotropic Geometry and Linear Elasticity. *J. Comput. Appl. Math.* 31, 4 (2008), 391–419.

Helmut Pottmann, Stefan Leopoldseder, and Michael Hofer. 2002. Simultaneous registration of multiple views of a 3D object. In *Intl. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXIV, Part 3A, Commission III*. 265–270.

Helmut Pottmann and Johannes Wallner. 2001. *Computational Line Geometry*. Springer Verlag.

Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. 2007. Developable Surfaces from Arbitrary Sketched Boundaries. In *Eurographics Symposium on Geometry Processing*. Eurographics.

Camille Schreck, Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani, Shuo Jin, Charlie C. L. Wang, and Jean-Francis Bloch. 2015. Nonsmooth Developable Geometry for Interactively Animating Paper Crumpling. *ACM Trans. Graph.* 35, 1, Article 10 (Dec. 2015), 18 pages.

Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, Ming C. Lin and Dinesh Manocha (Eds.). Lecture Notes in Computer Science, Vol. 1148. Springer-Verlag, 203–222. From the First ACM Workshop on Applied Computational Geometry.

Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. 2013. Computational Design of Actuated Deformable Characters. *ACM SIGGRAPH* 32, 4 (2013), 82:1–82:10.

Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2012. Flexible Developable Surfaces. *Comp. Graph. Forum* 31, 5 (Aug. 2012), 1567–1576.

Peng Song, Chi-Wing Fu, Prashant Goswami, Jianmin Zheng, Niloy J. Mitra, and Daniel Cohen-Or. 2013. Reciprocal Frame Structures Made Easy. *ACM SIGGRAPH* 32, 4 (2013), 10.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Symposium on Geometry Processing*. 109–116.

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. 2004. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP '04)*. ACM, New York, NY, USA, 175–184.

Tomohiro Tachi. 2010. Origamizing Polyhedral Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 16, 2 (March 2010), 298–311.

Tomohiro Tachi. 2013. Interactive Form-Finding of Elastic Origami. In *Proc. of IASS*.

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2015. Interactive design of developable surfaces. *ACM Trans. Graphics* (2015). to appear.

A. Wächter and L.T. Biegler. 2006. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming* 1, 106 (2006), 25–57.

Benedikt Wirth, Leah Bar, Martin Rumpf, and Guillermo Sapiro. 2011. A Continuum Mechanical Approach to Geodesics in Shape Space. *International Journal of Computer Vision* 93, 3 (2011), 293–318.

Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J. Mitra. 2011. Shape Space Exploration of Constrained Meshes. *ACM Transactions on Graphics* 30, 6, Article 124 (2011), 12 pages.

Lifeng Zhu, Takeo Igarashi, and Jun Mitani. 2013. Soft Folding. *Comp. Graph. Forum* 32, 7 (Oct. 2013).