Volume 37 (2018), Number 2

Packable Springs

Katja Wolff, Roi Poranne, Oliver Glauser and Olga Sorkine-Hornung

Department of Computer Science, ETH Zurich, Switzerland

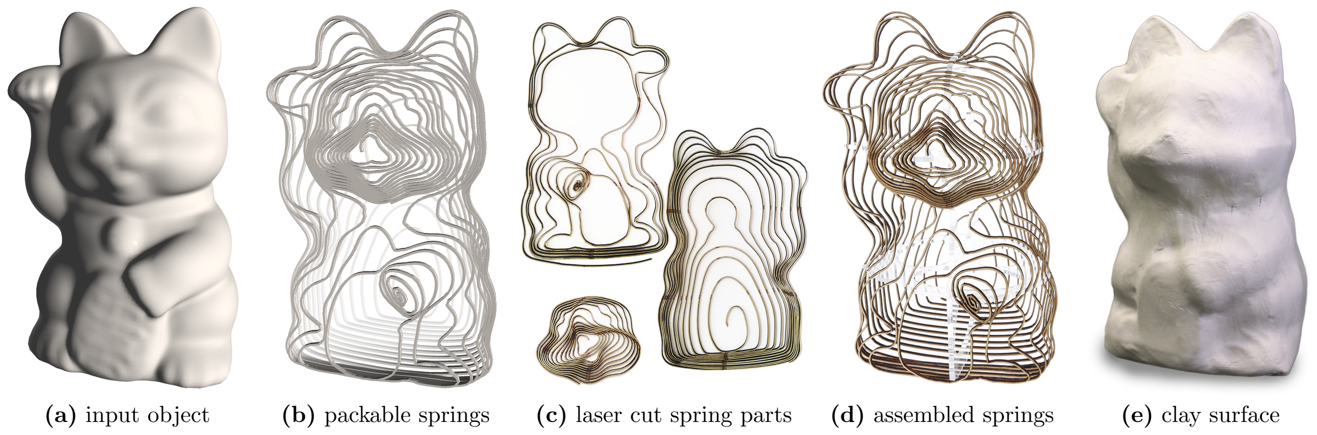

Figure 1: *Given an input object* (a)*, we construct a spring representation* (b)*, such that the springs consist of a small number of packable parts, which can be flattened to a plane. This allows us to laser cut the springs from a small number of flat material sheets* (c) *and easily assemble them to form their intended shape in 3D* (d)*. This structure can serve as formwork for a multitude of different materials and applications, e.g. for sculpting with clay* (e)*.*

Abstract

Laser cutting is an appealing fabrication process due to the low cost of materials and extremely fast fabrication. However, the design space afforded by laser cutting is limited, since only flat panels can be cut. Previous methods for manufacturing from flat sheets usually roughly approximate 3D objects by polyhedrons or cross sections. Computational design methods for connecting, interlocking, or folding several laser cut panels have been introduced; to obtain a good approximation, these methods require numerous parts and long assembly times. In this paper, we propose a radically different approach: Our approximation is based on cutting thin, planar spirals out of flat panels. When such spirals are pulled apart, they take on the shape of a 3D spring whose contours are similar to the input object. We devise an optimization problem that aims to minimize the number of required parts, thus reducing costs and fabrication time, while at the same time ensuring that the resulting spring mimics the shape of the original object. In addition to rapid fabrication and assembly, our method enables compact packaging and storage as flat parts. We also demonstrate its use for creating armatures for sculptures and moulds for filling, with potential applications in architecture or construction.

1 Introduction

Digital fabrication enables fast creation of real world objects from digital models. Many fabrication techniques exist today, ranging from state-of-the-art technologies such as 3D printing, to more traditional CNC milling or hot wire foam cutting. Laser cutting is a particularly appealing fabrication method, since it is extremely fast and affords a large variety of materials, including cheap yet durable options like plywood or acrylic glass. The main drawback of laser cutting is its restriction to flat shapes. Consequently, a multitude of research works explore design for laser cutting based fabrication and offer computational approaches to help users navigate the design space to reach their creative goals. For example, Hildebrand et al. [HBA12] use planar slices with prefabricated slits and assemble a representation of the input 3D shape by sliding them together. Another example can be found in [CSaLM13], where the authors

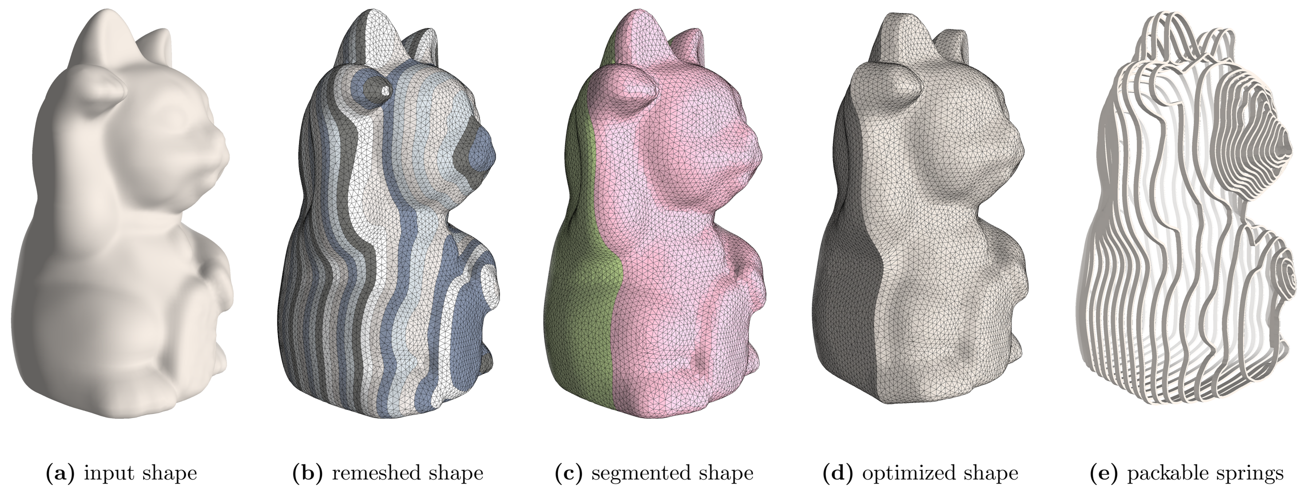

Figure 2: *To construct a spring representation of an input shape* (a)*, we first remesh it to create horizontal layers w.r.t. the spring direction* (b) *and then optimize for a desired normal orientation for each layer* (c)*. This results in n segments (red and green areas). We repeat* (b) *and* (c) *to find an optimal spring direction to reduce the number of segments and thus the required number of packable parts. To create the springs, we optimize the shape* (d) *such that each segment becomes "monotonic" by only minimally straying from the original appearance of the shape. Any spring* (e) *that lies on the optimized surface consists of at most n packable sub-springs. More than one spring might be needed to cover several "hills" of a segment. The spring representation in* (e) *consists of 2 springs or 3 packable parts.*

use a small number of planar polygonal primitives that form a closed surface. Many more techniques exist for attaching, interlocking and folding the flat panels to approximate a given 3D object, and each technique reveals its own design space.

In this paper, we present a radically different fabrication method based on laser cutting. We propose *packable springs*, a spring-like representation of an object that can be compressed to a flat state (Figures 1, 3). We draw inspiration from the field of spring design in mechanical engineering, where in some cases it is desirable for a spring to take up minimal space when compressed. For example, the packability of a cylindrical spring is limited, but a conical spring can be perfectly packed (Fig. 3, right). The packable spring representation thus has an advantage with regard to storage and shipping of the 3D object. However, a more surprising result is that packable springs can be laser cut using a small number of seemingly rigid material sheets: cutting a thin spring out of a flat panel produces an elastic object that can be lifted out of the plane to assume a 3D shape that approximates the input surface.

We define the packability of a spring as the minimal number of layers it has when compressed. The packability of a cylindrical spring is equal to the number of its turns, while the packability of a (sufficiently thin) conical spring is 1. An hourglass shaped spring (i.e., two cones connected at their apexes) has a packability of 2. In our setting, the packability of a spring directly influences the number of panels required to be cut, each panel forming one *sub-spring*. The entire fabricated object is assembled out of one or several springs, each consisting of possibly multiple sub-springs. Our goal is, given a 3D object, to find the best packable spring representation that strikes a balance between packability and similarity to the original shape (Fig. 1).

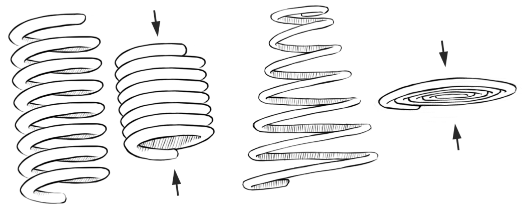

Figure 3: *Illustration of the packability of a cylindrical spring (left) and a conical spring (right). The cylindrical spring cannot be completely flattened and its packability value is the number of its turns. The conical spring can be flattened completely in this case, hence its packability value is 1.*

A naive approach to this problem is to initialize a spring directly on the shape as a spiraling curve on its surface, and then attempt to improve its packability. This quickly proves to be a rather difficult task, since it calls for an optimization of challenging quantities, such as the number of self intersections of the spring when projected to the ground plane. Furthermore, whenever one varies the desired parameters of the spring, such as the slope, direction, density of windings, etc., a new instance of the optimization problem must be solved. We propose a different approach: Instead of optimizing the spring, we optimize the *surface* in such a way that *any* spring designed on it is well packable.

Cutting our springs from a small number of material sheets results in just a few pieces that need to be put together, making manual assembly fast and simple. To hold the springs in place, additional holders along the silhouette of the object can be used. The fabricated objects preserve the creases and surface texture of the original object well, and can be used as a basis for traditional methods in arts and construction. One application that we found particularly appealing is armatures for sculpting. An armature is a simple structure used as a solid base for a sculpture. Artists usually construct wooden armatures and then place the material on them (e.g. clay) and form it to shape before gradually adding more detail. Our fabricated springs are well suited to be used as armatures, particularly for novices, thanks to their simplicity and detailed surface texture approximation. We found that by flattening a ball of clay and draping it over our springs we can easily create a sculpture that preserves many of the details of the original shape (Fig. 1 (e), Fig. 22 and Fig. 19 (left)).

Another application for our springs is moulds or formwork for creating solid shapes (Fig. 22, second object). This is done simply by extruding material into the spring object. The windings of the spring can be made tight enough such that viscous materials do not excessively leak out. Here, we only experiment with construction foam, but as shown in [Mes17], concrete can also potentially be used to create load bearing structures in architectural scale.

2 Related work

We review the most related literature on fabrication by cutting and wire based shape representations, as well as works concerned with the packing and stacking of objects.

Fabrication by laser cutting. Laser cutting is a fast and cheap method that has been explored as a fabrication tool in several works. A common strategy to utilize it for general shapes is to divide them into planar components that are assembled by different means. McCrae et al. [MSM11] create shape representations by arranging planar slices to optimize the perception of the original object, but they do not investigate the assembly of these slices. We compare to this method in Fig. 4. In [HBA12] objects are semi-automatically fabricated by cutting planar slices with slits and sliding them together to assemble a representation of the volume of the input shape, favoring arrangements of orthogonal slices. The works [SP12, SP13] also use a slit mechanism to compose 3D models, taking into account constraints on fabrication, assembly, rigidity and the resulting structure, and extending the approach to provide a more detailed formulation of the assembly of nonorthogonal slices. In [CPMS14], the planar slices are aligned to smooth cross fields, visually enhancing the approach.

Another class of approaches focuses on fabricating surfaces. Chen et al. [CSaLM13] form a closed surface by using a small number of laser cut planar polygonal primitives. We use this method on the Cat model in Fig. 4 for comparison. Song et al. [SDW*16] combine 3D printing and 2D laser cutting to support the fabrication of large 3D objects by an interior structure assembled from planar pieces. A method to approximate arbitrary 3D surfaces with auxetic materials, which can be cut from planar sheets, is proposed in [KCD*16], but also here a 3D printed model is needed for reference to form the object. Several works also allow for bending of the prefabricated pieces: works such as [MS04, STL06, MGE07] represent the input model through multiple planar, but foldable strips, usually made of paper, which can be glued together. An interactive tool for designing surfaces made from flexible interlocking quadrilateral elements of a single size and shape was presented in [SCGT15], while Pottmann et al. [PHD*10] propose a method to fit a freeform shape with

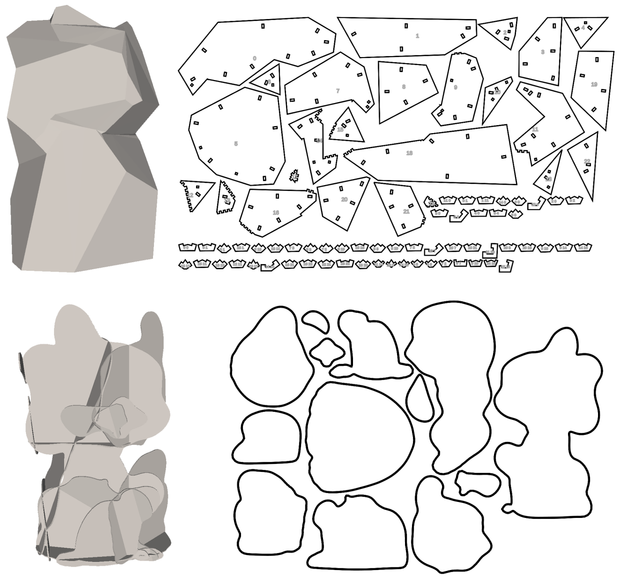
Figure 4: *Representations of the Cat model from Fig. 1 by two previous methods for fabrication using laser-cut pieces. Top row: [CSaLM13] utilizes 24 surface- and 58 connector-pieces to form a closed surface. Bottom row: [MSM11] creates an abstraction of the mesh by combining 13 planar slices. Both methods do not guarantee that their results are suitable for fabrication. In comparison, our method uses 3 laser-cut pieces for the spring and 3 for the holder to fabricate the Cat shape in Fig. 1 (d).*

single-direction bendable panels. All these methods utilize cutting to make 2D planar pieces, which then necessitate manual sorting and assembly into the 3D shape. In contrast, we propose to directly lift the cut spring shape out of the 2D plane, making the manual assembly easier and quicker.

Several methods utilize laser cutting for fast fabrication. The methods in [MKB13, UCM*15] create a 3D object by folding and stretching a laser-cut workpiece and additionally use the laser cutter to weld. This eliminates the need for manual assembly. The framework of [BGM*15] allows for fast fabrication of 3D objects by extracting straight and curved plates from a 3D model and substituting them with laser cut parts. In contrast to our method, these techniques either require the design of suitable objects or the existence of certain structures in the 3D object that can be substituted.

Fast fabrication has also been explored in the field of 3D printing. We only mention a few papers here that address similar problems. Vanek et al. [VGB*14] develop a framework to convert a shape into a segmented shell by hollowing its inner parts to save 3D printing time and support material. Hu et al. [HLZCO14] decompose a 3D volume into pyramidal building blocks, which can be 3D printed without support structures. Such a decomposition generally cannot be used to create a spring representation, but conversely, the decomposition of our approximated shapes (Fig. 1 (d)) is pyramidal.

Wire based representation. Wires are a low-cost and easily deformable material and have been used in several works to represent planar shapes and freeform surfaces. Iarussi et al. [ILB15] present a method to assist the creation of wire wrapped jewelry. Igarashi

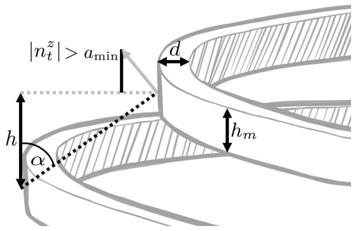

Figure 5: *Parameters: h (spacing of spring turns), d (spring width), h_m (material thickness), n_t^z (z-component of the triangle normal n^z, which needs to be bigger than a minimal value $a_{\min}$).*

et al. [IIM12] introduce a system for designing and constructing 3D beadwork, the art of stringing beads together using a wire to make a 3D shape. Regular wire meshes are used in [GSFD*14] to represent freeform surfaces, but again, a fabricated 3D reference model is needed to guide the forming deformation of the wire mesh. An assembly of planar wire shapes has been used to create stable structures in [MLB16]. Existing methods use simply bent wires, but due to the limitations of currently available consumer-grade wire bending machines, it is not possible to produce springs like the ones presented in this paper. We therefore do not currently use metal wires, although more advanced industrial machines or robotic wire bending could be potentially employed.

Instead of using metal wire, the works of [HZH*16, WPGM16] address the creation of feasible fabrication sequences for general rod meshes produced via thermoplastic extrusion. In [SFG*13, PTC*15, RA15], surfaces are represented with rod and beam structures of different properties. However, these approaches do not consider packability or spring representations.

Packability. Tightly packing several objects or parts of an object in a space that is as small as possible has obvious advantages for storage and shipping, and is therefore of great interest. Li et al. [LAZ*12] regard the problem of stacking multiple instances of the same object (e.g., a chair) along a stacking direction and optimize the object's shape with respect to maximal stackability. In [LHAZ15], space-saving furniture is designed by applying a minimum amount of modification to an object such that it can be folded. Jacobson [Jac17] generalizes the self-similar nesting of Matryoshka dolls to arbitrary solid objects and finds the largest scale replica of an object that nests inside itself. Yu et al. [YCC17] approximate shapes by telescopic structures for applications where mechanisms must be compact in size. Luo et al. [LBRM12] address how to partition large objects into smaller parts that are feasible for 3D printing. Yao et al. [YCL*15] focus on the partitioning and packing of a single 3D object, taking into account several metrics like printability, assembly and interface area. Guseinov et al. [GMB17] 3D print a flat piece of material, which takes on the shape of a smooth, doubly curved surface in 3D. Similarly, we consider the usage of as few sheets of material as possible.

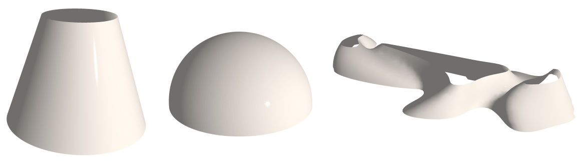

Figure 6: *Different generalized cones. Note that the shape of a generalized cone can differ significantly from a normal cone (right).*

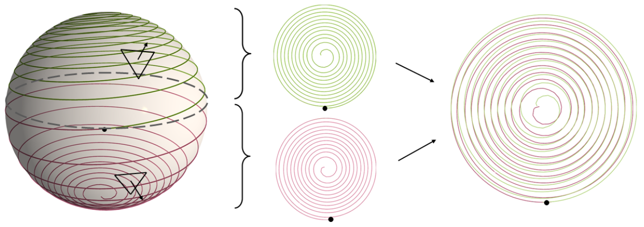

Figure 7: *The two packable parts of a sphere. The top half is a contracting generalized cone, and the bottom half is an expanding one. This is reflected in the normal orientation w.r.t. the z-axis (left): the normals of the contracting cone face upward and those of the expanding cone face downward. An arbitrary spring on the sphere has two packable sub-springs (middle). The whole spring is not packable (right). When the spring crosses the horizontal boundary between areas of different normal orientation (small black dot), it switches from one packable sub-spring to the next.*

3 Method overview

Our goal is to create a spring representation of a given 3D object that is as packable as possible, i.e., it can be cut from as few flat material sheets as possible. We assume force is applied perpendicularly to the flat panel the spring is cut out of, i.e., in the vertical z direction. Although springs can be deformed in other directions, potentially expanding the design space for packability, they tend to buckle in this case. Using a straight extension/contraction path also enables us to easily construct the final object by letting the spring fall onto the holders and then slightly tweak its positioning.

Packable springs. We define a *spring* as a space curve $\gamma(t) = (x(t), y(t), z(t))^\mathsf{T}$ such that its vertical component is monotonically increasing: $z(t_1) < z(t_2)$ for $t_1 < t_2$. Since we are interested in fabricating these springs, d denotes the desired horizontal width of the rectangular profile (see Fig. 1 (d) and Fig. 5). But for now, to simplify the method introduction, we assume that springs are 1D, i.e., have vanishing width d. A *packable* spring is a spring whose projection on the xy plane is not self-intersecting. An n-packable spring is a piecewise packable spring with n packable *sub-springs*. A given curve can be described as an n-packable spring in several ways, but there exists a minimal integer n such that the curve cannot be an $(n-1)$-packable spring. The question we ask is, how to modify the curve as little as possible in order to reduce the minimal n as much as possible. Note that more than one spring is needed for objects that are topologically not similar to a cylinder and that each of these springs can consist of several packable parts.

Generalized cones. Instead of attempting to optimize a single curve, we propose a different strategy. We optimize the surface itself, such that *any* spring lying on it will be n-packable. Our approach is based on the observation that any spring on an upright cone is packable. More generally, we can define the notion of *generalized cones*, which can be *expanding* or *contracting*. Let $\mathcal{S}$ be a shape and denote by $\Omega(t)$ the intersection of the shape with the plane $z = t$. Then $\mathcal{S}$ is an expanding generalized cone if for every $t_1 < t_2$, $\Omega(t_1) \subset$

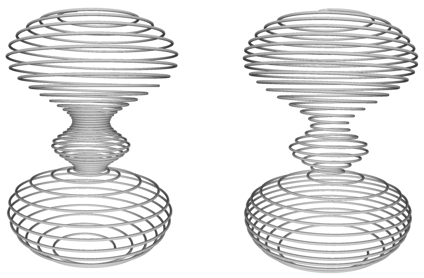

Figure 8: *Constant slope (left) vs. constant spacing of turns (right). The difference is particularly apparent in the middle part.*

$\Omega(t_2)$, and it is a contracting generalized cone if $\Omega(t_2) \subset \Omega(t_1)$. As with regular cones, any spring on a generalized cone is packable. The space of generalized cones is rather limited (see Fig. 6), but we can easily interleave contracting and expanding cones to obtain the much larger shape space of piecewise generalized cones (see Fig. 7 and Fig. 21). Any spring designed on a shape made of m stacked cones is n-packable, with $n \leq m$ ($n = m$ for shapes that are topologically similar to a cylinder). Thus, our approach is to optimize the input shape such that the result is comprised of a small number of generalized cones, while remaining similar to the original shape.

Spring design. The spring shape is determined by several properties, including the slope (defined by an angle α), the space between turns h, the direction (clockwise or counterclockwise) and the material thickness h_m (see Fig. 5 for an illustration of these parameters). Springs with constant slope can be easily defined, but only springs with even spacing between turns guarantee a minimum distance of windings when projected to the xy plane and thus a minimal material thickness of windings. Moreover, we consider springs with uniform spacing to be more visually pleasing (Fig. 8, right). This leaves the thickness d and the spacing of turns h, which both influence the maximal possible slope in the shape optimization (Sec. 4.2). Before optimizing the input shape, we define a minimal value h for the spacing between windings and a maximal value d for the wire width, which may not be exceeded when generating the spring. To create an even spacing between windings, we locally adjust the slope of the spring while computing its edges (Sec. 5).

Fabrication. Once a spring on the optimized shape is designed, we cut each packable sub-spring with a laser cutter from a flat material sheet. To assemble the pieces, we add a holder that keeps the desired shape of the spring in 3D (see Fig. 1 (c)-(d)).

Summary. In the following section, we separate the shape optimization for packability (Fig. 2 (a)-(d) and Sec. 4) from the computation of the spring on the surface of the object (Fig. 2 (e) and Sec. 5). This allows for fast design of the spring, postponing decisions like material thickness and spacing of windings, which can radically change the overall appearance of the spring. Once the shape is optimized for packability, we can guarantee for any spring designed on the shape's surface to consist of not more than the defined maximum number of packable sub-springs.

4 Packability optimization

In this section we explain our approach for packability optimization. The process consists of two steps: First, we segment the shape into regions that should become generalized expanding and contracting cones after optimization, and at the same time we optimize for the orientation of the spring axis, i.e., the z direction. In the second stage, we solve an optimization problem to modify the shape and project it onto the space of generalized cones determined by the first step. An important aspect of our algorithm is that we only modify the x and y coordinates of the points on the surface, keeping their z coordinates fixed. Allowing surface points to move only in the x and y directions is sufficient in order to span the entire space of applicable shapes of joined generalized cones, and it considerably reduces the number of variables in the optimization.

4.1 Segmentation

For each point $\mathbf{p}$ on the surface, with normal $\mathbf{n_p}$, we assign a value

$$f_\mathbf{p} = \operatorname{sign}(n^z_\mathbf{p}) = \pm 1,$$

where $n^z_\mathbf{p}$ stands for the z component of $\mathbf{n_p}$ (the z-axis points up and equals the axis of the spring). The scalar $f_\mathbf{p}$ induces a segmentation of the surface into up- and down-facing regions (see Fig. 9). A generalized cone has the same value $f_\mathbf{p}$ everywhere: $f_\mathbf{p} = +1$ if it is contracting and $f_\mathbf{p} = -1$ if it is expanding. As mentioned, any spring on a generalized cone surface is packable. A spring on a general surface is packable as long as its windings do not cross a boundary between segments or contain another segment between them. Otherwise, there is a risk that the spring's projection onto the xy plane self-intersects (see Fig. 10).

A shape consisting solely of contracting and expanding generalized cones has horizontal boundaries between segments of different $f_\mathbf{p}$ values (see Fig. 7). Whenever a spring crosses a boundary between regions of different $f_\mathbf{p}$, it switches from one packable sub-spring to the next. But when considering general input shapes, these boundary lines are not necessarily horizontal (see Fig. 9), which means a spring on the surface can cross the same boundary line more than once, potentially increasing the number of needed packable sub-springs. This implies that an ideal shape is made of stacked generalized cones, which is equivalent to having only horizontal boundary lines between $f_\mathbf{p}$ regions. In this case we can guarantee that these boundary lines are crossed only once by any spring on

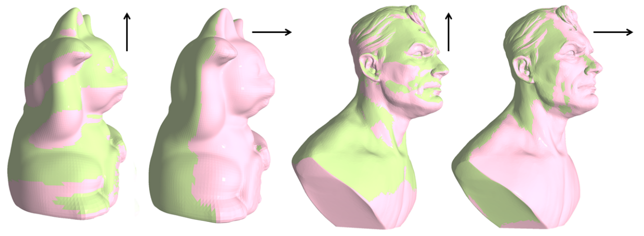

Figure 9: *Examples of boundaries between regions with different values $f_\mathbf{p}$ for different spring axes (z directions) shown by the arrows. The Cat and Superman are colored w.r.t. a vertical axis (left) and a horizontal axis (right). Regions with $f_\mathbf{p} = 1$ are in green, $f_\mathbf{p} = -1$ in red. The borders between regions are not planar.*

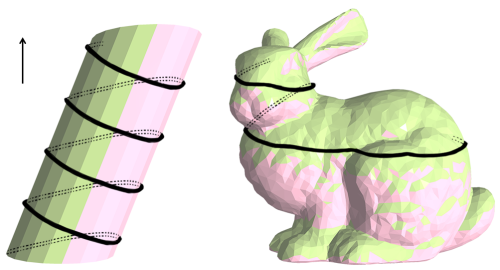

Figure 10: *When the boundary between different regions is not horizontal, a spring can cross it many times, increasing the number of necessary packable sub-springs (left). Even if a spring does not cross boundaries (right), but encloses regions of different $f_{\mathbf{p}}$ (the chin of the bunny), packability is not guaranteed. The arrow marks the z direction for both meshes.*

the surface, so that the number of packable sub-springs is smaller or equal to one plus the number of horizontal boundary lines (equal for shapes topologically similar to cylinders).

Our first step is to decide which parts should be expanding and which contracting, which is essentially equivalent to placing horizontal boundary lines. This step depends solely on the original surface normals, and is separated from the shape optimization stage.

Layer orientation optimization. We begin by discretizing the problem. Given an object, we remesh it such that it consists of horizontal layers (see Fig. 2 (b)). We emphasize that the specific remeshing technique is not critical, as long as most vertices of the resulting mesh lie on evenly spaced horizontal slices. In our case, we first intersect the mesh with horizontal planes, which results in a planar polygon for each slice. We then use adaptive remeshing as described in [DVBB13], while using the created polygons as constraints for edge positioning. Note that this can also create additional vertices that lie in between slices to create a more even mesh. The number of slices depends on the desired resolution. We used 30 slices on average in our examples. We define a *slice* as all horizontal edges of the same z-coordinate and the same connected component, and a *layer* refers to all connected faces between two slices. We use s to refer to a slice and l to refer to a layer. See Fig. 2 (b) for a visualization of the layers. Each layer l is assigned a variable $f_l \in \{-1, 1\}$ that represents whether the layer will lie on an expanding or contracting general cone after shape optimization. In other words, all triangles in the same layer should face the same way, either upwards or downwards, depending on f_l. We optimize the values f_l w.r.t. two objectives:

- Small amount of *switches* of the values f_l between consecutive layers, to limit the total number of required generalized cones.
- Minimal *mismatch* between the original normal directions of the triangles in a layer and the chosen f_l.

We solve the following meta-problem:

$$\begin{aligned} \min_{\mathbf{f}} \quad & E_{\text{mismatch}}(f_l, l) \\ \text{s.t.} \quad & E_{\text{switches}}(\mathbf{f}) < K, \end{aligned} \tag{1}$$

where K is a positive integer, $\mathbf{f} = (f_1, \ldots, f_L)^{\mathsf{T}}$, the function $E_{\text{mismatch}}(f_l, l)$ measures the total mismatch between the orientations of the triangles in layer l and f_l, and $E_{\text{switches}}(\mathbf{f})$ counts the number of sign switches in $\mathbf{f}$. There are many ways to formulate the two energies; we propose rather straightforward definitions:

$$E_{\text{mismatch}}(f_l, l) = -\frac{1}{F} \sum_l f_l \sum_{t \in l} n_t^z \tag{2}$$

$$E_{\text{switches}}(\mathbf{f}) = \sum_{p=1}^{L_{\text{pairs}}} \frac{1}{2} \left| f_{p_1} - f_{p_2} \right|,$$

where n_t^z is the z component of the unit normal of triangle t in the remeshed input mesh. Note that we measure the mismatch between the estimated orientation of a layer f_l and a triangle normal $\mathbf{n}_t$ only along the z-axis. To count the number of switches we regard the number of adjacent pairs of layers L_{pairs} and the layer values f_{p_1} and f_{p_2} for each pair p; F is the total number of triangles in the mesh. Dividing E_{mismatch} by F allows us to compare E_{mismatch} for different mesh orientations and thus for different remeshings. The inner sum $(\sum_{t \in l} n_t^z)$ in (2) is constant, hence the minimization is a pure linear binary program. We use [Gur16] to solve it.

Spring orientation optimization. In the above segmentation optimization, we assume a given orientation of the model w.r.t. the z-axis; the orientation of slices and the number of segments determined by the optimization relies on this orientation. As the segmentation takes less than a second for most of our meshes, we randomly sample 100 points on a unit sphere as different z directions, and for each direction we repeat the segmentation. We then pick the direction with the smallest E_{mismatch} value and treat it as the z axis.

4.2 Shape optimization

Once the target orientation of each layer is fixed (Fig. 2 (c)), the next step is to adjust the shape to match the layer orientations (Fig. 2 (d)) while staying as close as possible to the original shape and maintaining its appearance. An additional consideration is that physically fabricated springs are not just 1D curves, but have a certain width d. We therefore formulate a constraint for the maximal slope of each generalized cone, such that we can guarantee packability for each *physical* sub-spring. In this section, we use $\mathbf{v}_i = (x_i, y_i, z_i)$ to denote vertex positions, and stack them in $V = (X, Y, Z)$. Recall that only X and Y vary, while Z remains fixed.

Slope constraints. To guarantee the physical packability of sub-springs, we need to limit the slope of each generalized cone, which is equivalent to constraining the z-component of the unit normal of each triangle n_t^z to be greater than a given minimum $a_{\min}$. This $a_{\min}$ depends on parameters such as the expected width of the spring cross section d and the spacing between the windings of the spring h. We describe how to derive $a_{\min}$ at the end of this section.

The slope constraints are formulated for each triangle t by

$$\left| n_t^z \right| = \left| n_t^z(\mathbf{v}_{t_1}, \mathbf{v}_{t_2}, \mathbf{v}_{t_3}) \right| > a_{\min}, \tag{3}$$

where n_t^z is now a function of the vertex positions $\mathbf{v}_{t_1}, \mathbf{v}_{t_2}, \mathbf{v}_{t_3}$ of triangle t. However, we also need to ensure that the triangle normals of a layer are in the half-space defined by f_l, i.e.,

$$n_t^z f_l > 0, \quad \forall t \in l. \tag{4}$$

We can combine (3) and (4) into one constraint and write it more explicitly:

$$n_t^z f_l > a_{\min} \;\Rightarrow\; \frac{1}{2A_t}(\mathbf{e}_t^1 \times \mathbf{e}_t^2)^z f_l > a_{\min} \;\Rightarrow$$
$$C_t := (\mathbf{e}_t^1 \times \mathbf{e}_t^2)^z f_l - 2\,a_{\min} A_t > 0, \quad \forall t \in l, \tag{5}$$

where A_t is the area of triangle t, and $\mathbf{e}_t^1, \mathbf{e}_t^2$ are two of its edge vectors, e.g. $\mathbf{e}_t^1 = \mathbf{v}_{t_2} - \mathbf{v}_{t_1}$ and $\mathbf{e}_t^2 = \mathbf{v}_{t_3} - \mathbf{v}_{t_1}$; $(\mathbf{e}_t^1 \times \mathbf{e}_t^2)^z$ is the z component of their cross product. We note that the constraints in (5) are quadratic inequalities w.r.t. X and Y.

Energy. While the optimized shape must adhere to (5), it should stay close to the input surface $\mathcal{S}$ not only in terms of distance, but also the overall appearance. We propose to model this as follows:

$$\min_{X,Y} \; w_{\text{prox}} E_{\text{prox}}(V) + w_{\text{smooth}} E_{\text{smooth}}(V) + w_{\text{sim}} E_{\text{sim}}(V)$$
$$\text{s.t. } C_t > 0 \;\; \forall t, \tag{6}$$

where $E_{\text{prox}}(V)$ measures geometric proximity to the original shape, $E_{\text{smooth}}(V)$ measures the quality of the optimized shape, $E_{\text{sim}}(V)$ is a similarity measure to be discussed below, and $w_{\text{prox}}, w_{\text{smooth}}, w_{\text{sim}}$ are scalar weights. We define $E_{\text{prox}}(V)$ as the simple point-wise distance from the original shape:

$$E_{\text{prox}}(V) = \sum_i A(\mathbf{v}_i) \|\mathbf{v}_i - \bar{\mathbf{v}}_i\|^2, \tag{7}$$

where $A(\mathbf{v}_i)$ is the Voronoi area of vertex i, and $\bar{\mathbf{v}}_i$ are the original vertex positions. For $E_{\text{smooth}}(V)$ we use the Laplacian energy, see the formula in [BS08]. The similarity term requires a little bit more thought, and we devote the following paragraph to this discussion.

The similarity energy. Optimizing for packability distorts the original shape, and we always run the risk of getting a result that is distant from the input shape. In such cases we still want to preserve the fine surface details (see Fig. 15). One approach to cope with this problem is to add a distortion measure to the optimization. A full discussion on distortion measures is outside the scope of this paper; we refer the reader to [RPPSH17]. In a nutshell, distortion optimization aims to preserve the shape of individual triangles as much as possible, while the whole shape is deformed due to other energies and constraints. A typical consideration for shape optimization with minimal distortion is whether to treat the shape as a surface or a volume. Using volumes is usually visually preferable, as it better describes the full object, while a surface approach might cause the shape to look like a thin shell. However, volumetric meshes require much more resources due to higher complexity. Another consideration is the type of distortion measure to use, where the two most common choices are isometric and conformal distortion. Isometric distortion measures the change in distances, and is most commonly used to mimic elastic materials. Conformal distortion measures the change in angles, but completely ignores scaling.

Various combinations of distortion measures and shape representations are used in literature, but, we argue that none of them are quite appropriate for the space of generalized cones. In this space, it is reasonable to ask a cylinder and a cone to have similar energy values, since both of them have perfectly round slices, and neither isometric nor conformal distortion allows this. We therefore propose a different formulation that allows exactly that. Our energy is conformal, but applied not on the surface or the volume, but *per slice*.

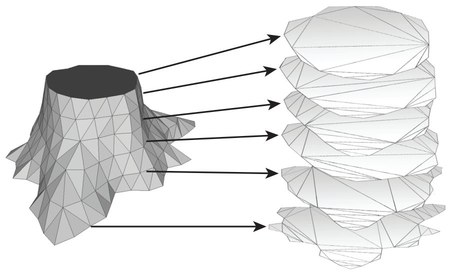

Figure 11: *Our similarity energy is not applied to the surface or the volume, but on the triangulated slices.*

We consider each slice of the shape separately, and ask to minimize the conformal distortion introduced to it during deformation. In practice, we triangulate each layer and use the discretized conformal distortion from [Lip12] (see Fig. 11).

Optimization. Our optimization problem (6) is sparse, non-convex due to E_{sim} and with quadratic constraints. We solve it by sequential quadratic programming using WORHP [BW13]. We provide the solver with analytical gradients and Hessians for the constraints and the energies E_{prox} and E_{smooth}. For E_{sim} we compute the gradients using the SVD formulas from [RPPSH17] and use the cotangent Laplacian as an approximation for the Hessian, as suggested in [KGL16]. We experimented with different sets of parameters, and found that the defaults work best.

Maximal slope. To guarantee packability for physical springs designed on generalized cones, the minimum distance between neighboring winding profiles must be equal to the thickness of the wires d (see Fig. 5). Therefore the angle α must satisfy $\alpha \geq \arctan(h/d)$. Since $\alpha = \cos(a_{\min})$, we can relate α to $a_{\min}$ as follows:

$$a_{\min} = \cos(\arctan(h/d)) = \left(1 + (h/d)^2\right)^{-1/2}. \tag{8}$$

We usually choose the smallest possible d (1-2 mm for wood) and set $a_{\min}$ to a small value (0.05-0.2) to avoid large shape deformations. We then calculate h from the equation above. The relation of these values is further discussed in the next section.

5 Spring design

After optimizing the shape of our object, we can guarantee a maximum number of n packable sub-springs for any spring we design on the optimized shape. Springs can then be created in many different ways. Springs with a uniform slope are easy and fast to compute given a starting point and an orientation (clockwise or counter-clockwise), as we show below. However, the distance between the windings in such a spring (Fig. 8, left) is related to the horizontal circumference of the object and therefore not constant. The resulting spring may look irregular and, most importantly, does not guarantee a minimal distance between the windings when projected to the xy plane, which we must ensure in order to be able to physically fabricate the spring while avoiding burning through and breakage. We thus focus on springs that give the impression of uniform vertical spacing between windings. Let us first describe how to generate a simple spring with a uniform slope, and then adjust the slope locally, such that we obtain uniform vertical distances between windings.

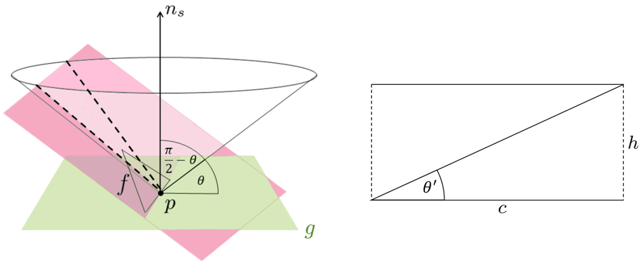

Figure 12: *On the left we show the intersection of a plane of a face (red) with a cone that represents all possible spring directions with an angle θ to the ground plane (green). On the right we show a cut and flattened cylinder to determine the needed angle θ′ to guarantee a minimal spacing h between windings.*

Uniform slope. A spring with a uniform slope w.r.t. the ground plane g (or, more precisely, a constant angle between the spring tangent and the spring axis n_s) is called a helix. We create such a spring (a polyline) by consecutively generating line segments on the mesh faces. Let θ be the chosen constant angle of incline ($\pi/2-\theta$ is then the angle between the tangent and the axis). Given a face f and a point p on its border, we can now represent all possible directions of the next spring edge by a cone with its apex at p, its axis equal to the spring axis and an opening angle of $\pi/2-\theta$ (see Fig. 12, left). When we intersect the plane of the face f with this cone, we get zero, one or two straight lines. If there is no intersection, the face is too flat to continue the spring creation and p is therefore one of the spring's endpoints. If we get one or two intersections, these represent the next spring segment, clockwise or counterclockwise, and we consistently choose one of them.

Local slope adjustments. As discussed before, a spring with uniform slope cannot guarantee that the physical, flattened spring of a given width d does not self-intersect. We must guarantee a given minimal distance h between windings (Fig. 8), so that $a_{min} \leq \cos(\arctan(h/d))$ holds (or $h \geq d\tan(\arccos(a_{min}))$, see Eq. (8)). We thus locally adapt the angle θ for each face. If our shape were a simple cylinder with circumference c, we could calculate the angle as $\theta' = \arctan(h/c)$ (see Fig. 12, right). For an arbitrary shape, we can calculate the circumference c of the object at p by horizontally slicing the mesh and computing the length of the resulting polygon. We then recalculate θ' for each face as above, intuitively approximating our shape by thin cylindrical layers.

In some areas of the shape, we do not need to enforce the minimal distance h if the shape is sufficiently flat in the corresponding layer. We can relax this requirement in favor of better shape coverage by the spring. Each layer is assigned a minimal face slope α_l, which is the minimum of all its face's slopes and which is equal to α in most layers. In layers with $\alpha_l > \alpha$, we can calculate a local winding height $h_l = d/\tan(\alpha_l)$ and compute the local slope as described above w.r.t. h_l. See the Cat's face in Fig. 1 (c) and (d).

Fabrication. To fabricate the generated spring, we project each packable sub-spring onto the xy-plane and cut it from a sheet of material (e.g., plywood) with a laser cutter. Due to small deformations of the material during cutting, we do not allow for "wire" diameters d smaller than 1 mm. After cutting, the sub-springs exhibit spring-like properties and can be elastically deformed to take on their intended 3D shape. To keep the springs in this position, we additionally laser cut a holder (see Fig. 1 (d) and Fig. 17). We manually select two or more orthogonal planes with their intersection axis approximately in the center of the structure, such that all spring parts are supported. The holders have small indents on the sides, which fixates the spring's position vertically. The spring itself is engraved at the top to prevent horizontal movement. The laser cutting pattern, including the engravings, is automatically generated.

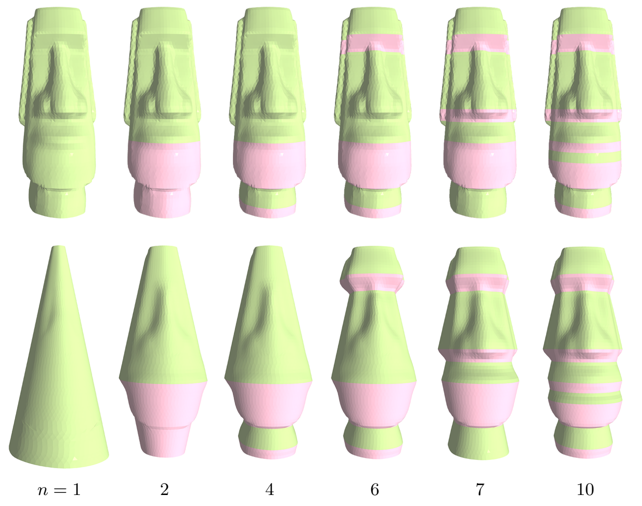

Figure 13: *Different amounts n of generalized cones used to approximate the input shape. Top: different results of the layer orientation optimization. Bottom: the object after shape optimization. To make the effect of choosing a higher n more visible, we chose a non-optimal spring axis here.*

6 Experimental results

Performance. Since we remesh all input objects, the running time depends on the chosen number of slices instead of the size of the input mesh. For the examples included in this work, the number of slices ranges from 20 to 120, resulting in 8k to 13k triangles per mesh. The following running times were computed on a 64 bit machine with 32 1.2 GHz CPUs and 64 GB RAM. The layer orientation optimization is fast thanks to being a small linear binary problem and takes not more than a second for all models. The shape optimization takes about 20 seconds to 30 minutes, depending on the number of slices and the chosen parameters.

Number of sub-springs. When segmenting the shape and optimizing the target orientation of each layer, we can choose a K to constrain the number of sub-springs in Eq. (1). Choosing a big K leads to more sub-springs, which in general results in a more faithful representation of the shape. Choosing K bigger than the number of slices leads to the maximum possible amount of sub-springs. Nevertheless, in many cases it is preferable to choose fewer, e.g. for artistic reasons or to decrease the number of needed material sheets. Fig. 13 shows the influence of the chosen number of generalized cones on the appearance of the object after shape optimization.

Maximal slope. Choosing the maximal slope, or equivalently the value of a_{min}, influences the maximal possible spring cross section

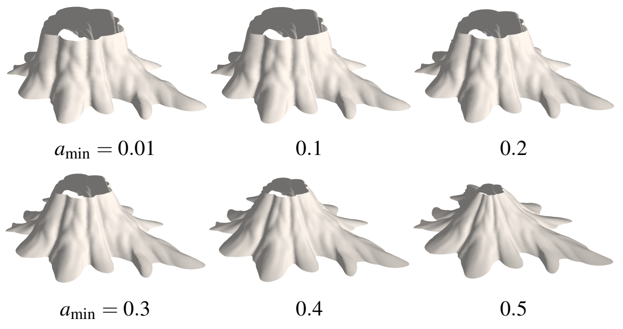

Figure 14: *Results for different slope values $a_{\min}$. A value of 0.2 is typically sufficient to fabricate a laser cut physical spring. The input mesh is shown in Fig. 17.*

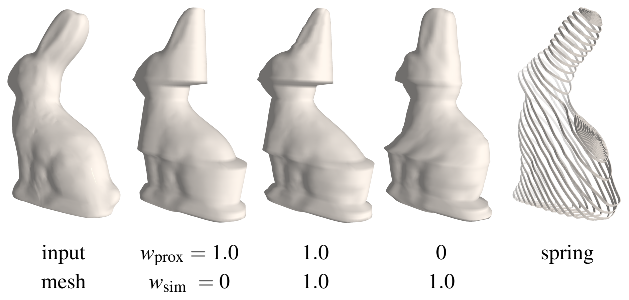

Figure 15: *Different combinations of the geometric proximity weight w_{prox} and the similarity weight w_{sim}. With higher w_{sim} we are able to retain more small details of the shape (middle, especially visible at the ear and main body), but with w_{prox} being much smaller than w_{sim} the shape drifts away from the original (middle right). For better visualization of this effect we chose a non-optimal spring axis. For comparison, we also show the spring with optimal spring orientation for this model (right).*

width d and smallest allowable spacing of windings h. A bigger value $a_{\min}$ leaves more freedom in the choice of d and h, which in turn influences fabrication and appearance: a smaller h results in a better "coverage" of the shape. On the other hand, a bigger $a_{\min}$ also leads to greater distortion of the shape, see Fig. 14. A bigger d is favorable due to the following reasons: Laser cutters are able to cut with very high precision, but we found that the material can deform slightly under heat, which results in broken pieces for d smaller than 2 mm. A higher d therefore minimizes these problems and results in a more stable fabricated object.

Proximity vs. similarity. When optimizing the shape in Eq. (6), we can set three scalar weights $w_{\text{prox}}, w_{\text{sim}}, w_{\text{smooth}}$ to influence the geometric proximity to the input shape, the similarity and the smoothness of the optimized shape, respectively. Fig. 15 shows how the choice of w_{prox} and w_{sim} influences the appearance of the optimized shape. Big values of w_{sim} retain more fine details after the shape optimization, but the similarity energy only preserves the overall appearance of each layer, and setting $w_{\text{prox}} = 0$ leads to layers shifting against each other. Therefore, a balance between both values (e.g., $w_{\text{sim}} = w_{\text{prox}}$) often gives the best results.

Smoothness. Without any smoothing, the optimized shape often exhibits sharp rims at the transition between contracting and expanding general cones (Fig. 16, left). These non-smooth transitions happen due to the slope constraints, which can force the optimized surface away from the original, but suddenly change at the boundaries of generalized cones. E_{prox} forces the optimized surface back to the input, creating rims. These are almost invisible in the final spring object (see e.g. Fig. 2 (d) and (e)), since the spring crosses these discontinuities in a single point and at an angle. Nevertheless, a small w_{smooth} is often sufficient to counter these rims if necessary. Too large w_{smooth} instead smooths out too many details.

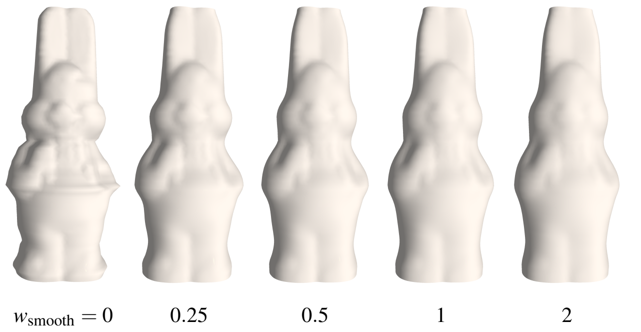

Figure 16: *Results for different amounts of smoothing. Without any smoothing, hard edges between the generalized cones can appear (far left), which can be countered by just a bit of smoothing. However, too high smoothing parameter w_{smooth} removes too many fine features (far right).*

Fabricated results. We show fabricated results for the Cat, Tree Stump, Venus and Dwarf before and after assembly in Fig. 1 and Fig. 17. We also fabricated a 90 cm large-scale branching structure (Fig. 18). Laser cutting the springs and the holders takes ca. 20-30 minutes, depending on the size of the model. The additional engraving of the spring at the locations touched by the holder adds about 5 minutes to the processing time. Manual assembly also takes approximately 10 to 20 minutes for each model. We provide some of the generated cutting plans in the additional material. We chose the materials in our examples for visual purposes, but other materials (e.g., acrylic glass, cardboard or even metal) would be suitable as well. Our springs can also be used as formwork for further production, when the volume of the shape should be filled with a certain material or as an armature to be covered with different materials. Fig. 22 shows how our springs can be used with different materials, like heat-drying polymer clay, construction foam, gypsum, wallpaper or air-drying clay. When using the spring as formwork, it serves as both the mold when pouring in the filler material, and a guide when cutting away excess material. As an armature, it is a helpful guide for the placement of proportions and detail (see Fig. 19, left). The springs are also useful for architectural landscape modeling (Fig. 20) and as decorative elements (Fig. 19, right).

7 Conclusions

We have shown how to optimize an input shape such that a spring representation can be fabricated from as few material sheets as possible while staying close to the input shape. Our spring representation is advantageous compared to e.g. separate loops, as the low number of pieces makes manual assembly much faster and the alignment and ordering of pieces easier.

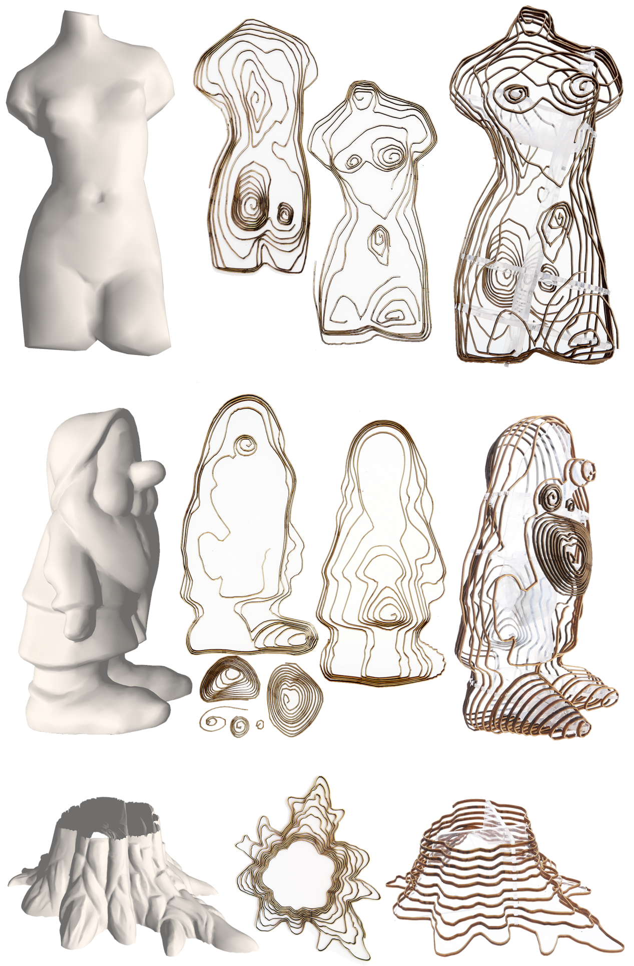
Figure 17: *Packable springs for different input meshes. We show the input mesh (left), the resulting packable springs (middle) and the fabricated object (right). We need 9 packable parts for the Venus, 8 for the Dwarf and 1 for the Tree Stump.*

Our method is limited to the straight vertical z direction when compressing or pulling the springs, but we would like to explore how to incorporate curved directions, and especially individual spring directions for objects where more than one spring is needed to cover the shape. The windings of the springs are often perceived as feature lines. Currently our approach is targeted to maximize surface coverage, but adapting the spring to follow features instead might improve the visual perception of the shape. While we still need an additional holder to fixate the spring to its target 3D shape, there are other solutions to explore in future work. When the individual sub-springs are deformed by gravity, they clearly do not deform uniformly, since more weight pulls on the upper parts of the spring. The work on inverse elastic shape design, e.g., [CZXZ14], could be applied to calculate the optimal radius of the spring, such that it deforms into its target 3D shape when lifted. For applications in architecture, where metals are needed for their necessary strength, one could use sophisticated wire bending machines or robotic arms to fabricate the springs in their 3D form, without dividing into sub-springs. Packability would still be needed for storage and shipping, with the advantage of "self-assembly" when unpacked, as the compressed spring springs back into shape. Alternatively to wire bending, cutting metal sheets is possible. The springs can be wrapped in cloth for tent-like structures or used as formwork for concrete, as a low-cost fabrication technique in architecture.

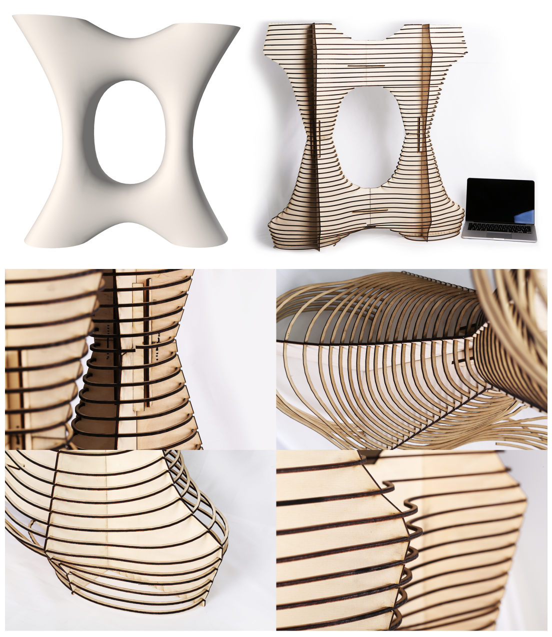
Figure 18: *Here we show how our method can be used to fabricate large scale objects with holes. The object is 90 cm tall and consists of 4 springs or 8 packable parts.*

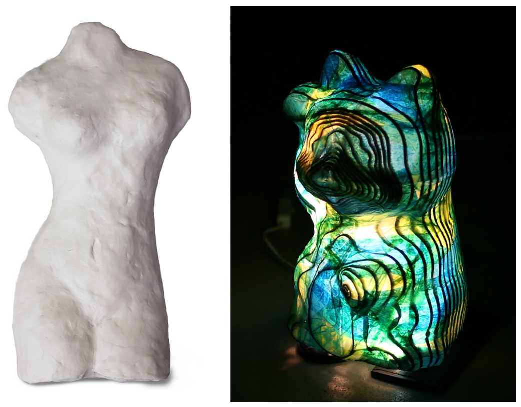
Figure 19: *When used as armatures, our springs allow even amateur artists to create clay objects with correct proportions and help with the placement of details. When covered with wallpaper, our springs can be used as decorative lamps.*

Acknowledgements

This work was supported in part by the NCCR Digital Fabrication, funded by the Swiss National Science Foundation Agreement #51NF40-141853, and by the European Research Council under Grant No. StG-2012-306877 (iModel). We thank Christian Schüller for providing us with his code for adaptive remeshing.

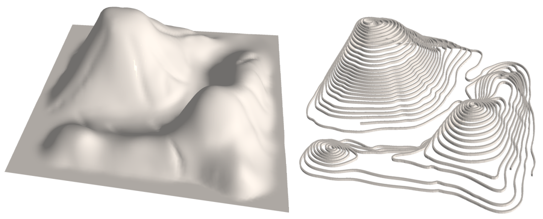

Figure 20: *Our springs can also be used for architectural landscape modeling. Usually these models are fabricated from stacked flat material sheets, using a lot of heavy material. Our approach can be a lightweight alternative produced from few pieces (here 3).*

References

[BGM*15] BEYER D., GUREVICH S., MUELLER S., CHEN H.-T., BAUDISCH P.: Platener: Low-fidelity fabrication of 3D objects by substituting 3D print with laser-cut plates. In *Proc. ACM CHI* (2015). 3

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graph. 14*, 1 (2008). 7

[BW13] BÜSKENS C., WASSEL D.: *Modeling and Optimization in Space Engineering*, vol. 73 of *Optimization and Its Applications*. Springer Verlag, 2013, ch. The ESA NLP Solver WORHP. 7

[CPMS14] CIGNONI P., PIETRONI N., MALOMO L., SCOPIGNO R.: Field-aligned mesh joinery. *ACM Trans. Graph. 33*, 1 (2014). 3

[CSaLM13] CHEN D., SITTHI-AMORN P., LAN J. T., MATUSIK W.: Computing and fabricating multiplanar models. *Comput. Graph. Forum 32*, 2 (2013), 305–315. 1, 3

[CZXZ14] CHEN X., ZHENG C., XU W., ZHOU K.: An asymptotic numerical method for inverse elastic shape design. *ACM Trans. Graph. 33*, 4 (2014). 10

[DVBB13] DUNYACH M., VANDERHAEGHE D., BARTHE L., BOTSCH M.: Adaptive remeshing for real-time mesh deformation. In *Proc. Eurographics, Short Papers* (2013). 6

[GMB17] GUSEINOV R., MIGUEL E., BICKEL B.: CurveUps: Shaping objects from flat plates with tension-actuated curvature. *ACM Trans. Graph. 36*, 4 (July 2017), 64:1–64:12. 4

[GSFD*14] GARG A., SAGEMAN-FURNAS A. O., DENG B., YUE Y., GRINSPUN E., PAULY M., WARDETZKY M.: Wire mesh design. *ACM Trans. Graph. 33*, 4 (July 2014), 66:1–66:12. 4

[Gur16] GUROBI OPTIMIZATION INC.: Gurobi optimizer reference manual, 2016. http://www.gurobi.com/. 6

[HBA12] HILDEBRAND K., BICKEL B., ALEXA M.: crdbrd: Shape fabrication by sliding planar slices. *CGF 31*, 2 (2012). 1, 3

[HLZCO14] HU R., LI H., ZHANG H., COHEN-OR D.: Approximate pyramidal shape decomposition. *ACM Trans. Graph. 33*, 6 (2014). 3

[HZH*16] HUANG Y., ZHANG J., HU X., SONG G., LIU Z., YU L., LIU L.: FrameFab: Robotic fabrication of frame shapes. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 224:1–224:11. 4

[IIM12] IGARASHI Y., IGARASHI T., MITANI J.: Beady: Interactive beadwork design and construction. *ACM TOG 31*, 4 (2012). 4

[ILB15] IARUSSI E., LI W., BOUSSEAU A.: WrapIt: Computer-assisted crafting of wire wrapped jewelry. *ACM Trans. Graph. 34*, 6 (2015). 3

[Jac17] JACOBSON A.: Generalized matryoshka: Computational design of nesting objects. *Comput. Graph. Forum 36*, 5 (2017). 4

[KCD*16] KONAKOVIĆ M., CRANE K., DENG B., BOUAZIZ S., PIKER D., PAULY M.: Beyond developable: Computational design and fabrication with auxetic materials. *ACM Trans. Graph. 35*, 4 (2016). 3

[KGL16] KOVALSKY S. Z., GALUN M., LIPMAN Y.: Accelerated quadratic proxy for geometric optimization. *TOG 35*, 4 (2016). 7

[LAZ*12] LI H., ALHASHIM I., ZHANG H., SHAMIR A., COHEN-OR D.: Stackabilization. *ACM Trans. Graph. 31*, 6 (Nov. 2012). 4

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph. 31*, 6 (Dec. 2012). 4

[LHAZ15] LI H., HU R., ALHASHIM I., ZHANG H.: Foldabilizing furniture. *ACM Trans. Graph. 34*, 4 (2015). 4

[Lip12] LIPMAN Y.: Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31*, 4 (July 2012), 108:1–108:13. 7

[Mes17] MESH MOULD:. http://www.dfab.ch/portfolio/mesh-mould/, 2017. Accessed: 2017-10-10. 3

[MGE07] MASSARWI F., GOTSMAN C., ELBER G.: Papercraft models using generalized cylinders. In *Proc. Pacific Graphics* (Oct 2007). 3

[MKB13] MUELLER S., KRUCK B., BAUDISCH P.: LaserOrigami: Laser-cutting 3D objects. In *Proc. ACM CHI* (2013). 3

[MLB16] MIGUEL E., LEPOUTRE M., BICKEL B.: Computational design of stable planar-rod structures. *ACM TOG 35*, 4 (2016). 4

[MS04] MITANI J., SUZUKI H.: Making papercraft toys from meshes using strip-based approximate unfolding. *ACM TOG 23*, 3 (2004). 3

[MSM11] MCCRAE J., SINGH K., MITRA N.: Slices: A shape-proxy based on planar sections. *ACM Trans. Graph. 30*, 6 (2011). 3

[PHD*10] POTTMANN H., HUANG Q., DENG B., SCHIFTNER A., KILIAN M., GUIBAS L., WALLNER J.: Geodesic patterns. *ACM Trans. Graph. 29*, 3 (2010). 3

[PTC*15] PÉREZ J., THOMASZEWSKI B., COROS S., BICKEL B., CANABAL J. A., SUMNER R., OTADUY M. A.: Design and fabrication of flexible rod meshes. *ACM Trans. Graph. 34*, 4 (2015). 4

[RA15] RICHTER R., ALEXA M.: Beam meshes. *Computers & Graphics 53* (2015), 28–36. 4

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. Graph. 36*, 2 (2017), 16:1–16:16. 7

[SCGT15] SKOURAS M., COROS S., GRINSPUN E., THOMASZEWSKI B.: Interactive surface design with interlocking elements. *ACM Trans. Graph. 34*, 6 (2015). 3

[SDW*16] SONG P., DENG B., WANG Z., DONG Z., LI W., FU C.-W., LIU L.: CofiFab: Coarse-to-fine fabrication of large 3D objects. *ACM Trans. Graph. 35*, 4 (July 2016), 45:1–45:11. 3

[SFG*13] SONG P., FU C.-W., GOSWAMI P., ZHENG J., MITRA N. J., COHEN-OR D.: Reciprocal frame structures made easy. *ACM Trans. Graph. 32*, 4 (July 2013), 94:1–94:13. 4

[SP12] SCHWARTZBURG Y., PAULY M.: Design and optimization of orthogonally intersecting planar surfaces. In *Computational Design Modelling*. Springer, 2012, pp. 191–199. 3

[SP13] SCHWARTZBURG Y., PAULY M.: Fabrication-aware design with intersecting planar pieces. *Comput. Graph. Forum 32*, 2 (2013). 3

[STL06] SHATZ I., TAL A., LEIFMAN G.: Paper craft models from meshes. *Vis. Comput. 22*, 9 (Sept. 2006), 825–834. 3

[UCM*15] UMAPATHI U., CHEN H.-T., MÜLLER S., WALL L., SEUFERT A., BAUDISCH P.: LaserStacker: Fabricating 3D objects by laser cutting and welding. In *Proc. UIST* (2015), pp. 575–582. 3

[VGB*14] VANEK J., GALICIA J. A. G., BENES B., MECH R., CARR N., STAVA O., MILLER G. S.: PackMerger: A 3D print volume optimizer. *Comput. Graph. Forum 33*, 6 (2014). 3

[WPGM16] WU R., PENG H., GUIMBRETIÈRE F., MARSCHNER S.: Printing arbitrary meshes with a 5DOF wireframe printer. *ACM Trans. Graph. 35*, 4 (2016). 4

[YCC17] YU C., CRANE K., COROS S.: Computational design of telescoping structures. *ACM Trans. Graph. 36*, 4 (2017). 4

[YCL*15] YAO M., CHEN Z., LUO L., WANG R., WANG H.: Level-set-based partitioning and packing optimization of a printable model. *ACM Trans. Graph. 34*, 6 (2015). 4

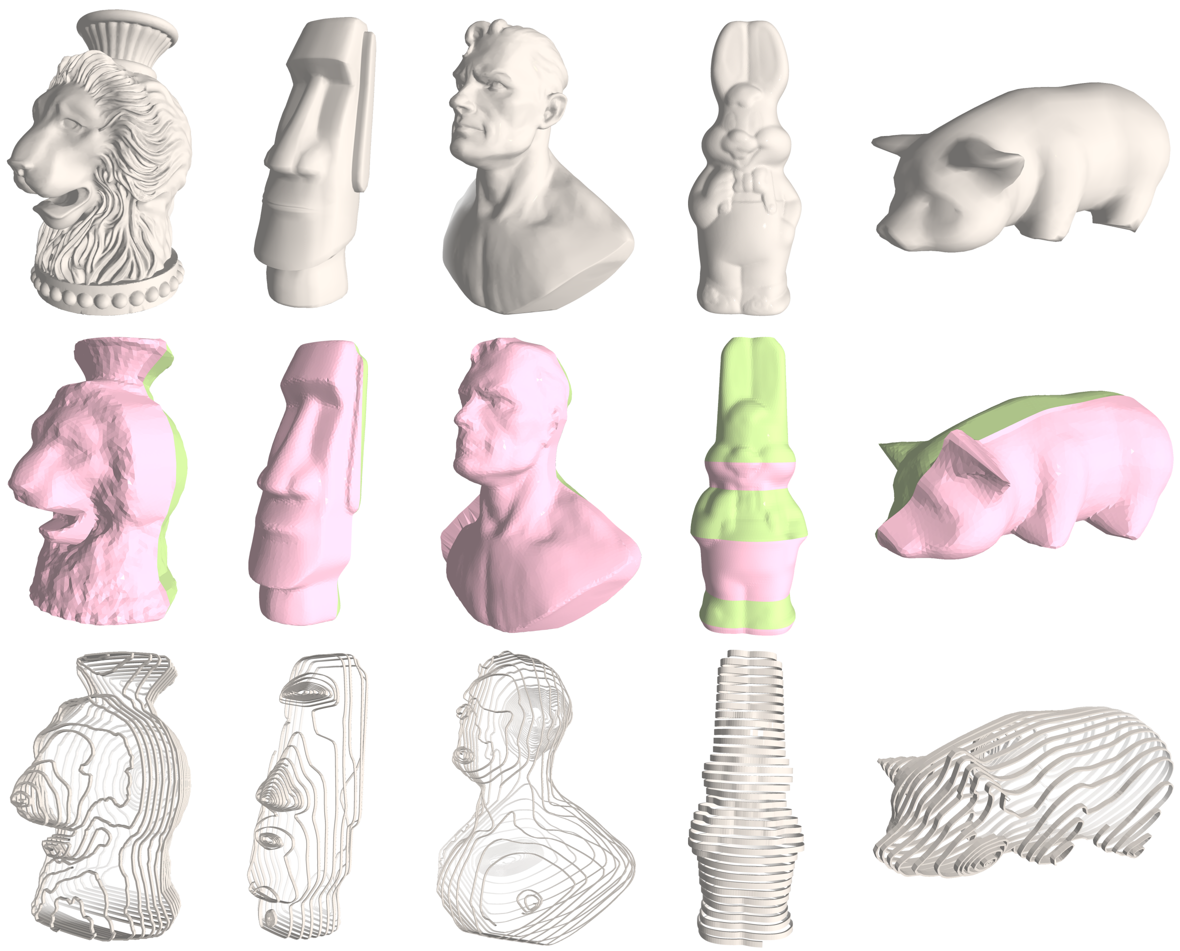

Figure 21: *Optimized shapes and final springs for various input models. We highlight the regions with different values $f_\mathbf{p}$ in green and red for the optimized shapes. The Easter Island statue, Superman and Chocolate Bunny all consist of 6 packable parts. The Lion Vase and the Pig consist of 5.*

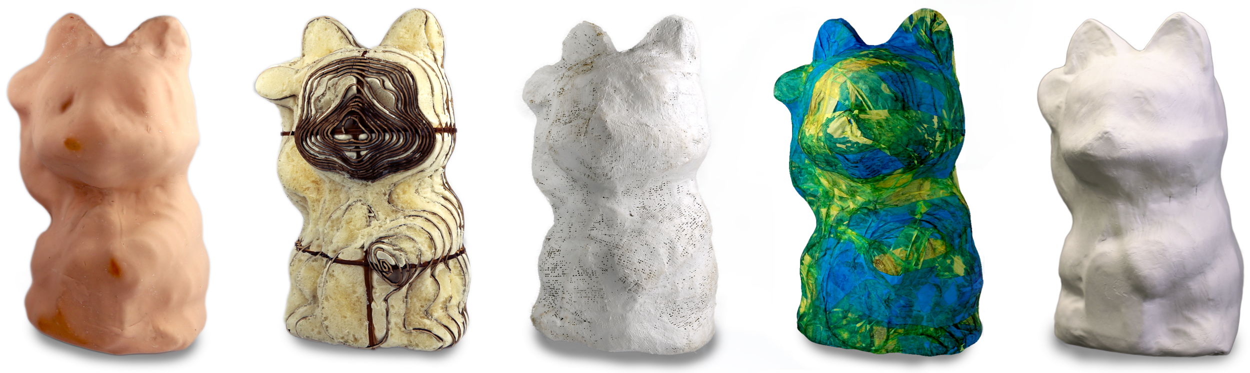

Figure 22: *Different possible applications of packable springs. From left to right we used plastic clay, construction foam, gypsum, wallpaper and clay to cover or fill the shape.*