Shape Representation by Zippables

CHRISTIAN SCHÜLLER, ROI PORANNE, and OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland

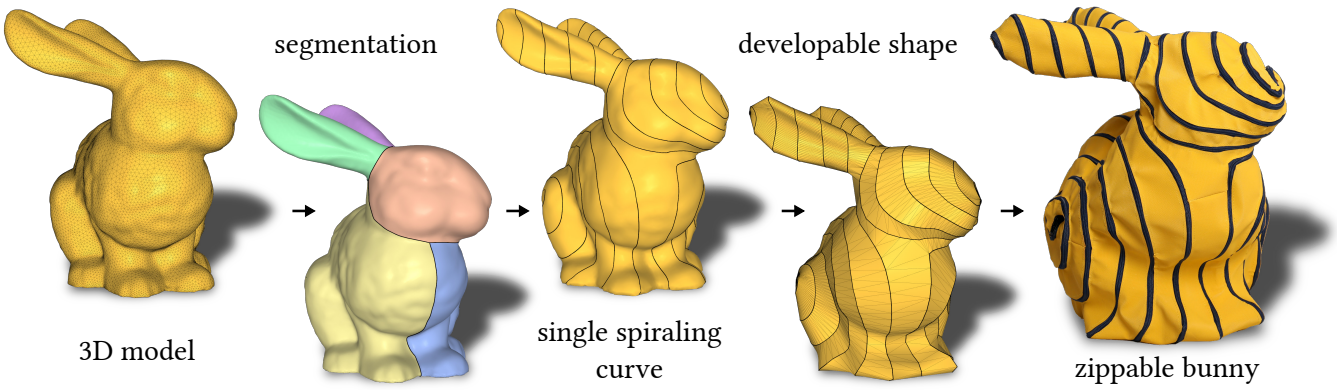


Fig. 1. The pipeline of our approach. Starting from a 3D model, the user decomposes the shape into topological cylinders. Our algorithm automatically produces a single continuous curve on the shape that spirals along the cylinders. It proceeds to cut the shape along the curve and creates a developable surface that can be trivially unfolded into a single 2D shape – the so called *zippable*. Based on the flattening, plans for laser cutting it from fabric are generated. Finally, we attach a zipper with a single slider to the boundary of the zippable. Zipping it up reproduces a faithful approximation of the input model.

Fabrication from developable parts is the basis for arts such as papercraft and needlework, as well as modern architecture and CAD in general, and it has inspired much research. We observe that the assembly of complex 3D shapes created by existing methods often requires first fabricating many small parts and then carefully following instructions to assemble them together. Despite its significance, this error prone and tedious process is generally neglected in the discussion. We present the concept of *zippables* – *single*, two dimensional, branching, ribbon-like pieces of fabric that can be quickly zipped up without any instructions to form 3D objects. Our inspiration comes from the so-called *zipit* bags [zipit 2017], which are made of a single, long ribbon with a zipper around its boundary. In order to "assemble" the bag, one simply needs to zip up the ribbon. Our method operates in the same fashion, but it can be used to approximate a wide variety of shapes. Given a 3D model, our algorithm produces plans for a single 2D shape that can be laser cut in few parts from fabric or paper. A zipper can then be attached along the boundary by sewing, or by gluing using a custom-built fastening rig. We show physical and virtual results that demonstrate the capabilities of our method and the ease with which shapes can be assembled.

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

Additional Key Words and Phrases: Digital fabrication, zippers, developable surfaces, mesh parameterization

This work was supported in part by the ERC grant iModel (StG-2012-306877).
Authors' address: Christian Schüller; Roi Poranne; Olga Sorkine-Hornung, ETH Zurich, Department of Computer Science, Universitätstrasse 6, Zurich, 8092, Switzerland.

0730-0301/2018/8-ART78
https://doi.org/10.1145/3197517.3201347

ACM Reference Format:
Christian Schüller, Roi Poranne, and Olga Sorkine-Hornung. 2018. Shape Representation by Zippables. *ACM Trans. Graph.* 37, 4, Article 78 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201347

1 INTRODUCTION

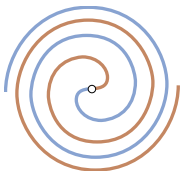
Representing shapes using developable surfaces is a problem with numerous applications, ranging from recreational activities like papercraft and plush toy fabrication, to large scale industrial design and modern architecture. In this paper we introduce the concept of

Fig. 2. The star model fabricated with our fastening rig. The insets on the left and in the middle show the developable model from a front and side perspective. Note how the fabricated star perfectly resembles the predicted shape. The segmentation is shown in the top right corner; below it a visualization of the zipper tape and the flattened *zippable*. The physical result has a height of 27 cm.

zippables – two dimensional, branching, ribbon-like pieces of fabric that can be zipped up to form 3D shapes. Our interest in this problem is inspired by a product commercially known as the *zipit* bag [zipit 2017]. This bag is made from a single, long piece of zipper. When zipped up, the ribbon folds and wraps around to form a simple bag. It takes only a few seconds to zip up and down, and no instructions are necessary. The simplicity of this concept immediately propelled us to ask whether this idea can be employed and generalized to facilitate the fast fabrication of *arbitrary* shapes. To this end, we devise a computational method to create a developable shape that approximates a given target 3D model when zipped up; see, e.g., Fig. 1 and the accompanying video. Our approach generalizes the simple straight ribbons that make up the *zipit* bags by allowing the zippables to turn, have varying width, and branch (see Fig. 2, bottom right inset).

We approach the problem in two stages. First, we compute a single curve on the surface, which represents the zipper-curve, i.e., the 3D path that the zipper should take. We then approximate the original surface geometry by a single developable surface – the zippable – whose boundary interpolates the zipper-curve (see Figures 2 and 4). Since resulting shape is largely determined by the first stage, several considerations must be taken into account when planning the zipper-curve: first, it should cover the surface as uniformly as possible, in order to get a uniform approximation. Equivalently, the zippable should have as little variation of width as possible. Second, the zipper-line should not curve excessively, as zippers tend to resist bending in the plane (see Fig. 18) and attaching them to a sharply turning curve is challenging.

There are several sensible strategies for tracing a zipper-curve. We discuss them and their limitations in Sec. 2.1. Our main observation is that the two aforementioned properties, uniformity and curvature, are trivial to achieve when the target 3D model is a cylinder: simply draw a spiraling curve on the cylinder from one boundary to the other, such that if cut along that curve, the resulting shape is a straight ribbon of constant width. For general target surfaces, our approach is based on first decomposing the shape into topological cylinders, and then mapping them onto cylindrical domains with low isometric distortion and in a seamless manner. We then draw "perfect" spirals on the cylinders and map these spirals back onto the input shape. Since the mappings minimize isometric distortion, the mapped curves tend to exhibit low curvature and low variation of distance between windings. Inspired by [Zhao et al. 2016], we connect the spirals on the different cylinders into a single, long curve using *Fermat spirals* (see inset). The shape is then cut along the computed curve to create a single, possibly bifurcated, but not yet developable, ribbon-like surface. A simple remeshing process transforms this surface into a developable one, essentially making a zippable. It can be trivially unfolded onto the 2D plane to create a cutting pattern. The resulting strands of the flat ribbon might overlap in the plane, and the pattern might take up too much space to be cut with an available laser cutter; in both cases we simply divide the ribbon into a few separate pieces before cutting them from fabric and attaching a zipper.



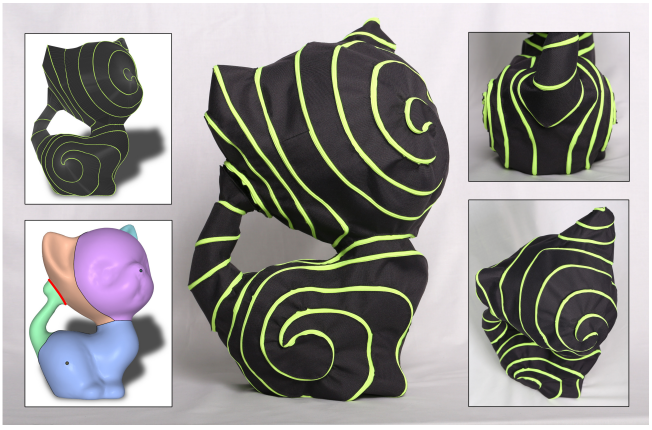


Fig. 3. A zippable shape of a kitten. Since it is topologically equivalent to a torus, an additional cut is needed (bottom left inset: marked in red where the tail touches the head). Another zipper could be used to close up this cut, but we opted for using Velcro instead.

In addition to the final zipping up being easy and entertaining, our assembly and fabrication process has distinct advantages over papercraft and many other similar methods in this domain. Most importantly, attaching the zipper to the zippable can be done by working solely in the flat plane. In contrast, attaching multiple parts in papercraft or sewing plush toys from multiple charts usually cannot be done in a flat configuration, but rather requires a certain assembly order and sewing in 3D, especially in the final stages, where all the parts have to come together for the shape to close up. The makers usually must refer to a manual, find the next piece and understand how to attach it to their work. In our case, the final assembly is *linear*, i.e., at every instant of the assembly process, the next action is unique and unambiguous, and it requires almost no instructions. In case our method generates self-overlapping strands that must be severed in order to laser-cut them from fabric, sewing the pieces together is simple, since both ends are flat, perfectly matching in length and only requiring flat stitching along a straight line. Furthermore, the zipper replaces the gluing lashes, adhesive tape or other connectors, which can be challenging to work with in free space. To further simplify the alignment of the zipper to the zippable, we propose an optional fastening rig that enables sliding the zipper in and keeping it in place before attaching to the fabric. This could be of particular importance in the context of industrial fabrication, where processes must be streamlined and automated.

We demonstrate our method on various shapes, see e.g. Figs. 3, 8. We show virtual results and physically fabricated objects, assembled and disassembled simply by zipping up and down.

2 RELATED WORK

Our work relates to the general field of computational fabrication and digital geometry processing. We briefly review the most relevant previous works below.

Papercraft and needlework. Some objects, typically of limited size and/or with certain constraints on the shape, can be directly manufactured in one piece using e.g. 3D printing or CNC milling. However, a large number of approaches propose to create shapes composed

of several individually fabricated parts. In this space, the most related approaches to ours are papercraft based on cutting and gluing [Massarwi et al. 2007; Mitani and Suzuki 2004; Shatz et al. 2006; Straub and Prautzsch 2011; Takahashi et al. 2011], and manufacturing by sewing [Igarashi and Igarashi 2008; Igarashi et al. 2009; Julius et al. 2005; Mahdavi-Amiri et al. 2015; Mori and Igarashi 2007; Wang 2010]. Both types of methods require the approximation of a given 3D shape by pieces of developable surfaces (in the case of paper) or highly stretch-resistant material (in the case of fabric). This is typically achieved by segmenting or cutting the shape into parts with low Gaussian curvature and parameterizing each part onto the plane. Alternatively, the shape can be a priori modeled or approximated as a piecewise developable surface, which is a current topic of active research [Chandra et al. 2015; Fondevilla et al. 2017; Jung et al. 2015; Kilian et al. 2008; Kolmianič and Guid 2002; Liu et al. 2009; Rose et al. 2007; Tang et al. 2016; Zeng et al. 2012]. The assembly by gluing or sewing the pieces together requires precision and carefully following the instructions. In contrast, in our case, due to the few parts it is mostly a straightforward and nearly mindless task.

Soft materials. Designing for fabrication using soft material like fabric or rubber is challenging, since such materials easily deform under stress and gravity. The main goal is to predict the deformation and solve an inverse problem, so that the fabricated model assumes the desired shape when subjected to the stress. Examples include the generation of plush toys [Igarashi and Igarashi 2008; Mori and Igarashi 2007], inflatable structures [Skouras et al. 2012, 2014] and rubber objects [Bickel et al. 2010; Chen et al. 2014; Skouras et al. 2013], among others. As mentioned, although we use fabric in our results, the behavior is more reminiscent of papercraft, because we use a nearly inextensible cloth, and the zipper itself is rather stiff and completely inextensible. Regardless of the type of fabric used, our zipper based approach has one decisive advantage: The whole fabrication process can be done entirely in the flat. In contrast, previous methods result in spatially curved pieces once they become connected to each other, which makes sewing or gluing much harder, as confirmed by professional tailors. The final 3D assembly by zipping up needs no instructions, is fast and fun to do.

Parameterization. Our approach relies on mesh parameterization, which is an extensively studied topic, see e.g. the survey in [Hormann et al. 2007]. The more recent relevant parameterization literature is presented in [Kovalsky et al. 2016; Rabinovich et al. 2017; Smith and Schaefer 2015]. Our method introduces a *new type* of global parameterization, which extends cylindrical parameterization [Tarini 2012]. Global parameterization is primarily used for quad meshing, where parts of seams in the parameter domain are related to each other by a rotation of integer multiples of $\pi/2$. A recent review can be found in [Bommes et al. 2013]. In our specific case, we require a different form of seamless mapping, based on cylindrical domains [Knöppel et al. 2015; Martin et al. 2008; Ray et al. 2006; Thiery et al. 2012]. Our main inspiration is [Kälberer et al. 2011], where the authors propose an approach for drawing stripes on *tubular* shapes that can be used for generating textures (see also [Livesu et al. 2017]). Their approach is based on a seamless parameterization aligned with a 2-RoSy field obtained from the principal curvature directions. This causes problems near umbilical points, where the principal directions are unstable. In contrast, we employ a parameterization based on distortion minimization, which avoids this problem, and generally leads to less distorted mappings [Myles and Zorin 2012].

Fig. 4. Our design for a zippable star pillow, made of two differently colored fabrics flatly attached together. Zipping it up generates an interesting interleaving of the two parts. Two of the five Fermat spirals are clearly visible in the top right inset.

2.1 Alternative curve design strategies

We briefly sketch a few alternatives to our zipper-curve design method and discuss their drawbacks.

Mesh stripification. Finding a 2D shape that *perfectly* reproduces a 3D mesh can be achieved using so-called *mesh stripification* algorithms, which cut the mesh into triangle strips [Eppstein and Gopi 2004; Lubiw et al. 2010; O'Rourke 2015; Rossignac 1999]. However, in addition to being highly mesh dependent, the resulting strips have many sharp turns that make attaching a zipper and manipulating the zipper slider difficult, if not impossible, and result in an uneven strip width.

Space filling curves. A recently granted patent proposes a method for approximating shapes by developable surfaces passing through *labyrinths* on the surface [Pedersen 2011] (see also a related paper [Pedersen and Singh 2006]), reminiscent of space filling curves. While these curves are smooth and uniform, they bend excessively, which would make attaching a zipper very hard if not impossible.

We summarize the properties of each method in Table 1.

	Stripification	Paper craft	Labyrinth	Ours
Single part	✓	x	✓	✓
Uniformity	x	✓	✓	✓
Low curvature	x	✓	x	✓

Table 1. Comparison of alternative curve design approaches.

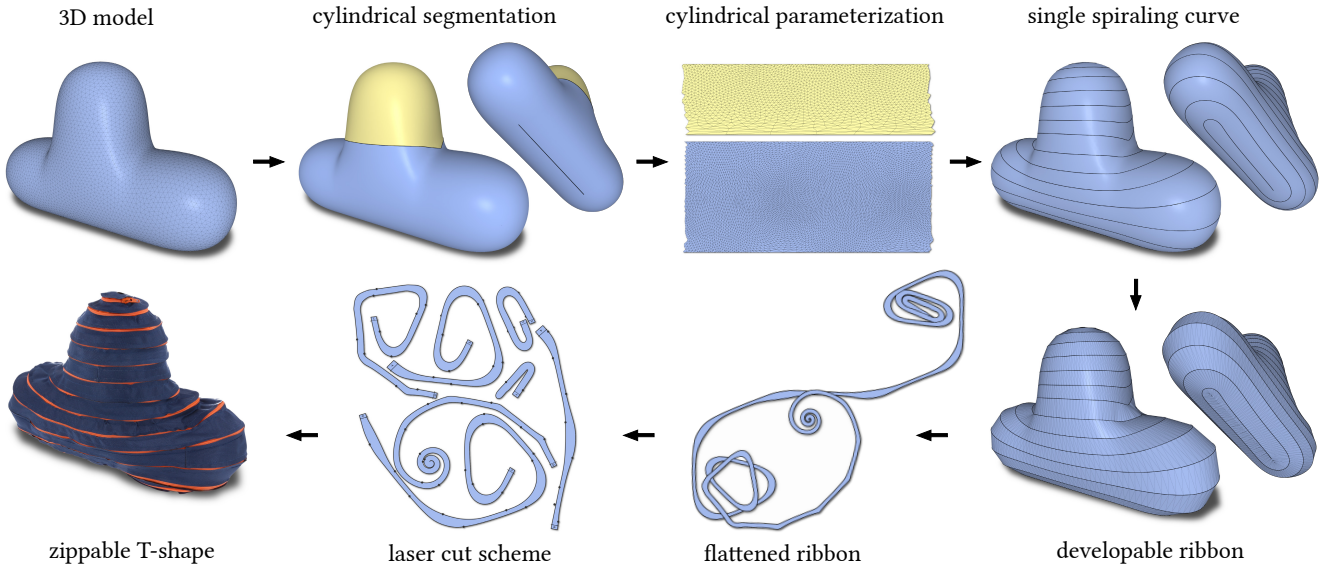


Fig. 5. Overview of our pipeline. We begin by segmenting a 3D model to cylinders, followed by a global cylinder parameterization. Using the parameterization, we trace a spiraling curve on the shape. The shape is then cut along the curve and approximated by a developable ribbon. The ribbon is then unfolded to the plane and offset. We proceed by packing the design in order to create a cutting program for a laser cutter. Finally, we cut the design from a piece of fabric and sew a zipper along its boundary. When zipped-up, the ribbon reproduces the original shape.

3 METHOD

Given a mesh representing the 3D object, our goal is to generate a single, flat, possibly branching and/or self-overlapping shape, referred to as the *zippable*, which approximates the object when "zipped-up". We can rigorously define zipping-up as an isometric deformation of the flat shape into a 3D shape such that the boundary exactly overlaps with itself. However, we assume that our intention is clear and avoid mathematical rigor at this point. In addition to being "zippable", we wish to enable some creative control by allowing the user to define where the zipper should pass and how it should be oriented or aligned.

Creating a zippable is equivalent to tracing a path – the zipper-curve – on the surface, which forms the boundary of the zippable. Our method is based on the observation that it is trivial to generate a uniform spiral on a cylinder with no caps. We can cut it from one boundary loop (the top) to the other (the bottom) and unfold it to a rectangular shape, where the two boundary curves become the top and bottom edges, and the cut becomes the two side edges. We then place "copies" of the unfolded cylinder side-by-side, and draw a straight line from the bottom corner of the leftmost edge to the top corner of the rightmost edge, see Fig. 6 for an illustration. Overlaying the copies on top of each other creates several disconnected, parallel line segments on the parameterization of the cylinder, and by rolling it back to a cylinder, these segments transform into a perfect connected spiral. Its number of turns depends on the number of copies we made.

The same approach, termed *cylindrical parameterization*, can be applied to general cylinder-like shapes, which we continue informally calling "cylinders". We start in the same manner by cutting

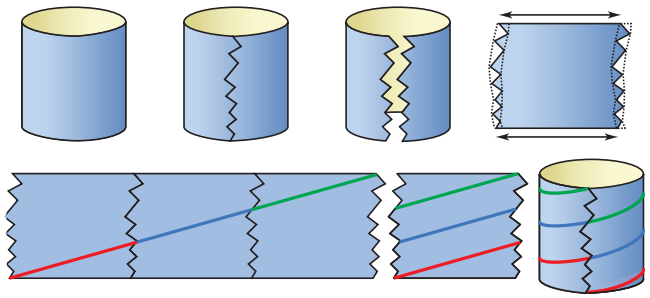


Fig. 6. Drawing a spiral on a cylinder can be done by cutting the cylinder from the top boundary to the bottom one and unfolding it to the plane. We then place copies of the flattened cylinder and draw a straight line that passes from the bottom leftmost corner to the top rightmost one. Overlaying the copies and folding back to a cylinder creates a spiral, where the number of windings is equal to the number of flattened copies.

the shape from one boundary loop to the other. The cut shape is then mapped to the plane by a distortion minimizing parameterization with *seam constraints*, which force the two sides of the cut to match like puzzle pieces. Minimizing distortion is necessary for the straight line in 2D to be mapped to a smooth and uniform spiral on the surface in 3D. The case of a more complex shape is slightly more involved: we decompose the shape into cylindrical parts and use a global parameterization scheme to smoothly map all the parts to cylinders. We discuss this in more detail in Sec. 3.2. Once the mapping is found, we turn to designing spirals on the cylinders. The main challenge is to synchronize the spirals, such that one spiral ends where another begins, resulting in a single, long zipper-curve

on the surface. The final task is to cut the surface along the zipper-curve and remesh it such that the result is developable. It is then trivially isometrically unfolded to generate the final 2D shape of the zippable.

To summarize, the design phase of our method consists of four stages (see Fig. 5 for a graphical overview):

(1) Decomposition into cylindrical parts.
(2) Seamless, global parameterization of the cylinders.
(3) Zipper-curve generation.
(4) Cutting along the zipper-curve, remeshing and flattening.

3.1 Decomposition into cylindrical parts

We partition the input shape S into N *topological* cylinders C_i, $i = 1, \ldots, N$, i.e., 2-manifolds with two boundary loops. The decomposition plays a substantial role in the final appearance of the spiral, since the resulting curve is aligned to the boundaries of the cylinders. It enables a flexible interface for artistic exploration and is achieved by interactively tracing the boundaries of the segmentation using our software (see the accompanying video). Since surfaces without boundaries cannot be decomposed into topological cylinders, we also allow the user to cut the shape open and place new boundaries, e.g., small circular holes or curves on the shape that act as a single cylinder boundary (see Fig. 7). Our methodology is to start by segmenting the shape into discs, which we feel is more intuitive, and then "puncture" them to obtain topological cylinders. Alternatively, more automatic ways of cylindrical decomposition such as [Livesu et al. 2017; Zhou et al. 2015] could be applied, but we have not explored this option. We distinguish between the *transition* boundary and the *open* boundary of a cylinder. The former being the boundary that separates it from adjacent cylinders, while the later is an actual boundary of the shape (a punctured hole or curve, as explained above). Further, we define an *interface* to be a shared, continuous edge sequence of two transition boundaries between two cylinders. Therefore, a transition boundary consists of at least one but usually multiple interfaces.

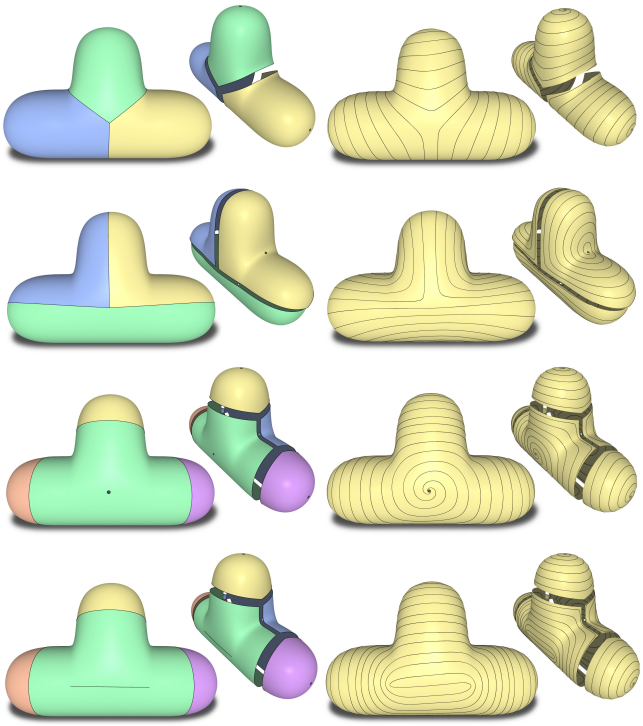


Fig. 7. Different cylindrical segmentations of a T shape. Each cylinder has one transition boundary and one open boundary. Note the small holes in the middle of the colored parts. These are the open boundaries for the corresponding cylinders. In the last row we show an example of a straight curve cut, serving as the open boundary of the green cylinder. Compare the resulting spiral to the segmentation in the third row to see the effect of the curve cut.

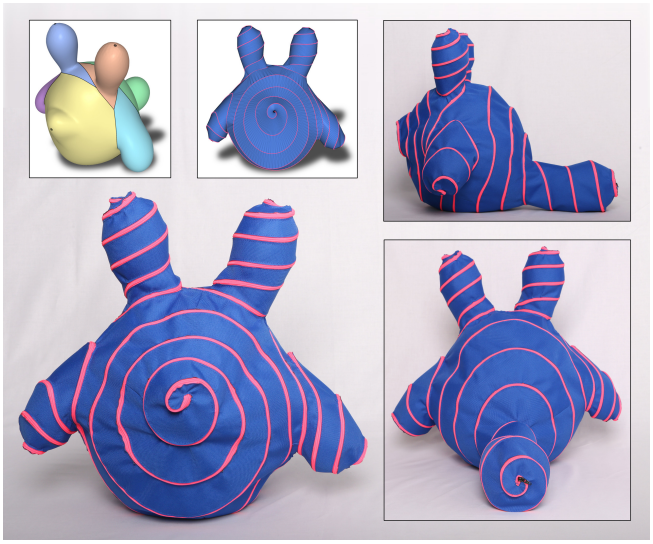


Fig. 8. A zippable model of an anime character. The zipper starts at the tail and spirals around all extruding parts until it ends at the tip of the nose.

3.2 Seamless parameterization

Once the cylindrical decomposition of S into parts C_i is available, we proceed to compute the parameterization. This step determines how the equally spaced, straight lines in the 2D parameter domain transform into a spiraling zipper-curve on S. To obtain a spiral that is as evenly spaced on the 3D shape as possible, the parameterization must minimize isometric distortion. Additionally, we require the parameterization to be bijective for the mapping from the 2D lines to the 3D curve to be well defined. Fig. 9 compares curves generated with the initial (i.e., suboptimal) and optimized parameterization. We assume that S has been cut along the boundaries of the C_i's, and each cylinder is cut from one boundary to the other, analogous to the example of one cylinder (see Fig. 6 top row). The edges and vertices along the cuts are duplicated, and we keep correspondences between the copies. We generate a seamless bijective parameterization of each C_i, with seamless transitions between adjacent C_i's. We first explain the case where there is only one cylinder, and the general case of multiple cylinders immediately follows.

Minimizing isometric distortion. An isometric distortion measure quantifies the difference between a given flattening of a shape and a perfect isometry; most formulations define it as a sum of the distortions of individual triangles. In this paper we use the recently proposed *symmetric Dirichlet* distortion measure; see [Kovalsky et al. 2016; Rabinovich et al. 2017; Smith and Schaefer 2015] for details.

We denote the coordinates of a vertex in the parameter domain by $\mathbf{x} = (x, y)$ and stack all the coordinates in a vector $\mathbf{X}$. The distortion of a triangle t is a function of the positions of its vertices in the plane. We denote the symmetric Dirichlet measure of triangle t by $D_t(\mathbf{X})$. Then the optimization problem to solve is

$$\operatorname*{argmin}_{\mathbf{X}} \sum_{\text{triangle } t} A_t \, D_t(\mathbf{X}), \tag{1}$$

where A_t is the area of t in the original mesh. In our work, we use a modified Newton's method [Shtengel et al. 2017] with a feasible starting point to solve this problem. Since we add several constraints in the following, we defer a detailed discussion to the end of this section.

Seamless cylindrical parameterization. To map a single topological cylinder C_i to the plane with minimal distortion in a seamless manner, it is cut to form a disk topology and then parameterized while adhering to seam constraints, whose role is to ensure that the parameterization is invariant to the cut [Myles and Zorin 2012]. In the cylinder case, the seam constraints call for each edge on one side of the seam to be a translation of its twin edge on the other side of the seam. More precisely, assume the cut contains n consecutive vertices and let $\mathbf{x}_j^{\mathrm{L}}$ and $\mathbf{x}_j^{\mathrm{R}}$, $j = 1, \ldots, n$, be the two copies of each vertex in the parameterization (superscripts L, R stand for Left and Right). Then the cylindrical seamlessness constraints are

$$[\,\mathrm{Cyl}(C_i)\,] \qquad \mathbf{x}_j^{\mathrm{L}} - \mathbf{x}_{j-1}^{\mathrm{L}} = \mathbf{x}_j^{\mathrm{R}} - \mathbf{x}_{j-1}^{\mathrm{R}}, \quad j = 2, \ldots, n. \tag{2}$$

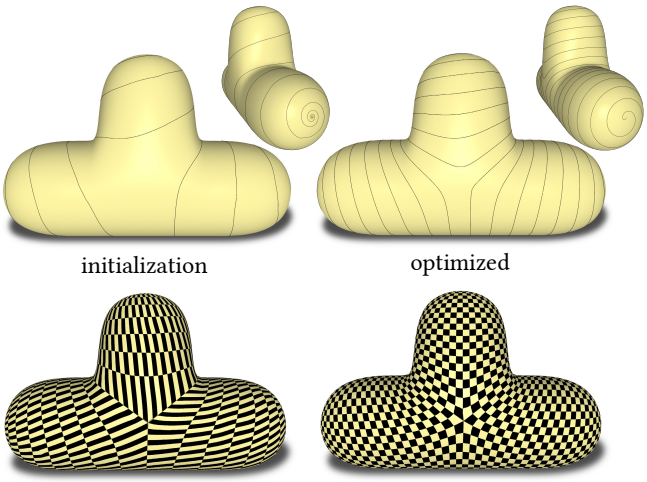


Fig. 9. Comparison between the curve obtained before and after minimizing isometric distortion of the parameterization. Note that the non-optimized spirals have a much greater variation in the spacing between the windings. We show a uniform grid texture to illustrate the difference in distortion.

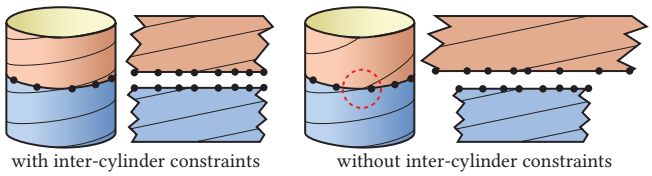
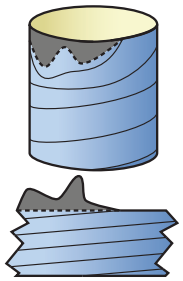


Fig. 10. Inter-cylinder constraints ensure that the transitions between cylinders are smooth for curves with the same slopes (left). Note the kink that appears in the curve when these constraints are missing (right).

We use the differential form of the seam constraints in order to avoid introducing auxiliary variables. The equivalent positional form is $\mathbf{x}_j^{\mathrm{L}} = \mathbf{x}_j^{\mathrm{R}} + \mathbf{t}$, $j = 1, \ldots, n$, where $\mathbf{t}$ is an unknown offset (the same for all vertices). For conciseness, we refer to the set of constraints in (2) as $\mathrm{Cyl}(C_i)$ for a given C_i, or Cyl in general.

Cylinder boundary constraints. In addition to the cylinder seam constraints Cyl, we also require the boundary loops of the cylinders to be mapped to straight lines. This serves two purposes: First, together with the Cyl, it guarantees bijectivity, and second, it allows for a better surface coverage by the spiral. Indeed, when the boundaries are not kept straight and allowed to "spill out" in the 2D domain, the spilled region is not covered by the zipper-curve (see illustration in the inset). Due to Cyl, the straight lines of the boundaries must be parallel, hence, without loss of generality, we can make them parallel to the horizontal x-axis. Let y_k^{Top}, $k = 1, \ldots, m^{\mathrm{Top}}$ and y_l^{Bot}, $l = 1, \ldots, m^{\mathrm{Bottom}}$ be the y-coordinates of the vertices of the top and bottom boundaries in the parameter domain. We again use the differential form for the straight line constraints, given by



$$[\,\mathrm{Str}(C_i)\,] \qquad \begin{aligned} y_k^{\mathrm{Top}} - y_{k-1}^{\mathrm{Top}} &= 0, \quad k = 2, \ldots, m^{\mathrm{Top}} \\ y_l^{\mathrm{Bot}} - y_{l-1}^{\mathrm{Bot}} &= 0, \quad l = 2, \ldots, m^{\mathrm{Bottom}} \end{aligned} \tag{3}$$

We denote the constraints of each C_i in (3) by $\mathrm{Str}(C_i)$, and the entire set of these constraints as Str. The constraints Cyl and Str together already result in a nice spiral on each C_i separately. However, without dedicated treatment, there could be a visible kink when transitioning between C_i's (see illustration in Fig. 10). Indeed, if copies of the same edge on a transition boundary are parameterized to two edges with different size and direction, then the parameterization does not appear smooth across that edge. We handle this issue in the following.

Inter-cylinder seamlessness. In order for the transition between C_i's to appear smooth, we apply seam constraints on cuts between each pair of neighboring C_i's. These constraints enforce a rigid transformation between the two sides of each seam (see Fig. 10). Since we have the freedom to define the exact transformation, we choose a rotation by π. Thus, for every two neighboring C_P and C_Q, we let $\mathbf{x}_r^P$ and $\mathbf{x}_r^Q$, $r = 1, \ldots, s$, be the coordinates of the two copies of each vertex along the seam between C_P and C_Q. Then the inter-cylinder seam constraints can be written in differential form

as

$$\left[\,\mathrm{Int}(C_P, C_Q)\,\right] \quad \mathbf{x}_r^P - \mathbf{x}_{r-1}^Q = \mathbf{x}_{r-1}^P - \mathbf{x}_r^Q, \quad r = 2, \ldots, s. \tag{4}$$

We denote the constraints in (4) for each pair C_P, C_Q by $\mathrm{Int}(C_P, C_Q)$.

Global cylindrical parameterization. With all the types of constraints defined, we can formulate the optimization problem for parameterizing the surface:

$$\begin{aligned} \underset{\mathbf{X}}{\text{argmin}} \quad & \sum_{\text{triangle } t} A_t\, D_t(\mathbf{X}) \\ \text{s.t.} \quad & \mathrm{Cyl}(C_i), && \forall\, C_i \\ & \mathrm{Str}(C_i), && \forall\, C_i \\ & \mathrm{Int}(C_P, C_Q), && \forall\, C_P, C_Q \text{ neighbors.} \end{aligned} \tag{5}$$

Eliminating degrees of freedom. The constraints in (5) are all sparse linear homogeneous equalities, but have some redundancies. For example, it is unnecessary to require all of the top and bottom boundaries to be on straight lines: half of them is sufficient, since the other half then must lie on straight lines due to the Int constraints. Similarly, the y part of (5) is redundant due to the Str constraints. We automatically remove all of these redundant constraints using Gaussian elimination.

Initialization. Our optimization is based on Newton's method and requires a feasible starting point with no triangle flips. We use Tutte's embedding with uniform weights, which guarantees bijectivity if the boundary is convex. We can therefore initially map each cylinder to a rectangle in the plane, where the top and bottom boundaries are parallel to the x axis, in order to satisfy Str. We set the height of each rectangle to have the length of the cylinder boundary to get a more isometric initial guess. To satisfy Int we require each edge of a transition boundary to have the same length in its parameterization. We can do the same for the cylinder boundary edges, which then completely determines the boundary.

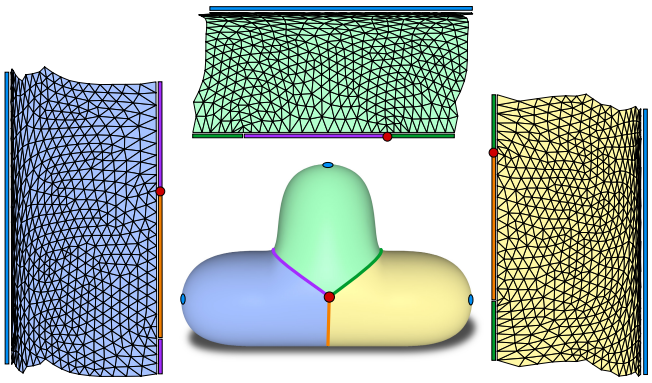


Fig. 11. The parameterization of the three topological cylinders of the T shape. See Fig. 7 for another perspective of the same segmentation. We mark the corresponding interfaces by matching colors, and the red dot represents one of the two points where all cylinders meet. We remark that this point has no particular significance and is only there as a visual guide. The blue sides of each flattening and the corresponding ellipses represent the open boundaries, while the unmarked sides represent the constrained cylinder seams.

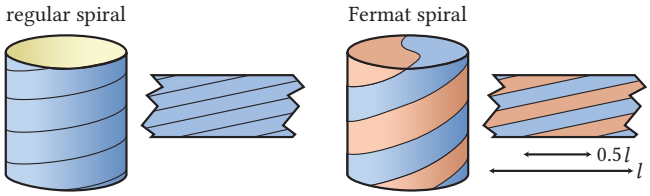


Fig. 12. Illustration of a Fermat spiral on a cylinder. The cap of the cylinder on the right represents the open boundary where the center of the Fermat spiral appears.

However, we note that we can in fact use Cyl, as they are in the form of an orbifold Tutte's embedding (see [Aigerman and Lipman 2015]), instead of specifying vertex positions directly, which results in a slightly less distorted initial guess.

Optimization. We use a modified Newton's [Shtengel et al. 2017] method with linear constraints to solve (5). We use the line search method suggested in [Smith and Schaefer 2015], which guarantees that no triangle flips are introduced during optimization. We show an example of the parameterization of the T shape in Fig. 11.

3.3 Spiraling zipper-curve

With the global seamless parameterization of the cylindrical decomposition available, the next stage in our algorithm is to generate a spiraling curve that represents the zipper-curve. This is done by drawing straight lines in the parameter domain and lifting them back into 3D using the inverse of the parameterization. A spiral on a single C_i can be created as discussed in the beginning of Sec. 3, by drawing a straight line on the cylinder's parameterization. For the case of multiple C_i's, in order to obtain a single continuous curve, we must make sure that the spirals of the individual C_i's connect. We make the following simplifying assumptions:

(1) The curve visits each C_i exactly once.
(2) The curve starts and ends at an open boundary and enters and leaves each cylinder through different interfaces.
(3) The curve traverses cylinders via Fermat spirals, except for the first and the last one.

Note that a Fermat spiral requires drawing *two* lines on a cylinder (see Fig. 12). The assumptions above mean w.l.o.g. that a curve must start at an open boundary in C_1, then cross the transition boundary through an interface to the adjacent C_2 and touch the open boundary of C_2. Next, leaving the open boundary at another point, passing back through the transition boundary via another interface to C_3. This continues until all cylinders are traversed, and the curve ends at the open boundary of the last C_i. See Fig. 13 for an illustration.

There are several choices we let the user make. The first is the traversal order of the C_i's. To assist the user, we enumerate all valid traversal orders, which are essentially all the Hamiltonian paths on the graph of segments, and let the user choose one by cycling through them. A valid path is guaranteed to exist if the number of segments is smaller than 11 [Barnette and Jucovič 1970]. Excluding pathological cases, the number of possible paths grows dramatically with respect to the number of segments. Many of them can be eliminated by letting the user pick a start and end segment.

Another choice is the location on an interface where the curve passes between C_i and C_{i+1}. Finally, the user can prescribe the number of windings of the spiral on each C_i. See Fig. 14 for an example of the different choices, also demonstrated in the accompanying video.

These choices impact the final appearance of the zipper-curve and should be based mostly on artistic considerations. In general, having even spacing between the windings greatly contributes to the aesthetics of the zipped-up shape. For a regular spiral, even spacing is ensured by the low-distortion parameterization. For Fermat spirals we would like to draw the two lines parallel, such that their copies are uniformly spaced. Therefore, the horizontal distance between the lines should be half the width of the boundary (Fig. 12). Unfortunately, this is not always possible, since the placement of interfaces between the different C_i's might not allow it. To illustrate this, consider a cylinder C_i with $i > 1$, where the zipper-curve comes in from C_{i-1} through the interface I_i^{in} and leaves to C_{i+1} through the interface I_i^{out} (see Fig. 13, e.g. $i = 2$). The two boundaries (open and transition) of C_i are parameterized to straight, parallel segments of the same length l_i. For all $C_i's$ with $1<i<N$, we need to pick two points on each segment that will be the end points of the incoming and outgoing lines. For the open boundary, we are free to pick any two points, and the best option is two points with a distance of $0.5\,l_i$. W.l.o.g. we can assume that the interface segments lie on the x-axis. For the transition boundary, we would like to pick two points $x_i^{\text{in}}, x_i^{\text{out}}$ such that

$$\left| x_i^{\text{in}} - x_i^{\text{out}} \right| = 0.5\, l_i. \tag{6}$$

However, since $x_i^{\text{in}} \in I_i^{\text{in}}$ and $x_i^{\text{out}} \in I_i^{\text{out}}$, this is clearly not necessarily possible. Even if $I_i^{\text{in}}, I_i^{\text{out}}$ do permit (6), we must recall that $x_{i-1}^{\text{out}} = x_i^{\text{in}}$, which thus couples all transition interfaces together. Nevertheless, we can solve an optimization problem to determine the best transition points. We formulate the problem as follows:

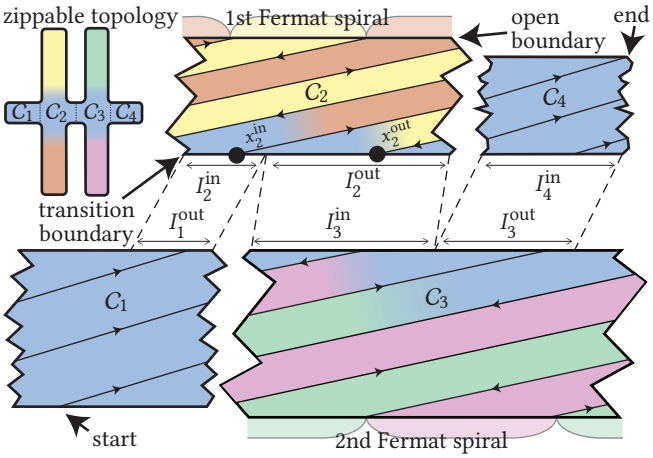


Fig. 13. Illustration of a spiraling curve traversing several cylinders. We mark the interfaces of C_i by I_i^{in} and I_i^{out} for $i = 1, 2, 3, 4$ (see Sec. 3.3). The colors help to visualize the zippable's connectivity and relate to the topology of the zippable shown in the top left corner. Every Fermat spiral creates a new branching part shown in red/yellow and green/pink.

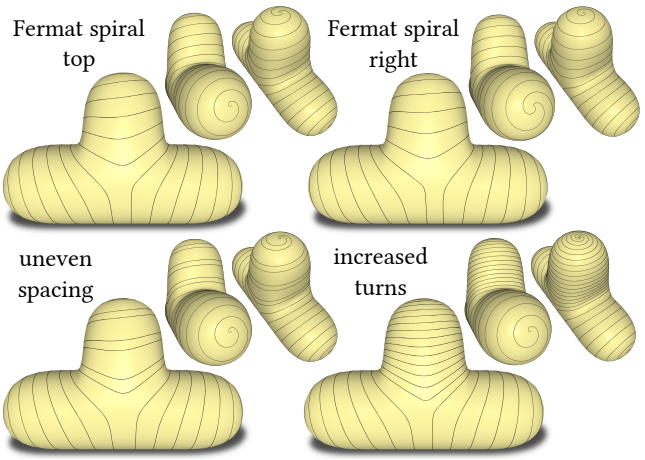


Fig. 14. We show several possible zipper-curves on the T shape. The design is up to the user's artistic choices.

$$\begin{aligned} &\underset{x_i^{\text{in}}, x_i^{\text{out}},\, i=2,\ldots,N-1}{\text{argmin}} && \sum_i \left(\left| x_i^{\text{out}} - x_i^{\text{in}} \right| - 0.5\, l_i \right)^2, \\ &\text{s.t.} && x_i^{\text{in}} \in I_i^{\text{in}},\ x_i^{\text{out}} \in I_i^{\text{out}} \quad \text{and} \\ & && x_{i-1}^{\text{out}} = x_i^{\text{in}}, \end{aligned} \tag{7}$$

where we exclude C_1 and C_N since they do not contain Fermat spirals. We have the freedom to pick the starting point of each I_i such that $x_i^{\text{out}} > x_i^{\text{in}}$ is always satisfied, allowing to drop the absolute value from the objective in (7). There still might be a translational degree of freedom, and we remove it by requiring transition points to be close to the middle of their intervals. In Fig. 15 we show the difference between a naive initialization, which tries to keep the zippable width as constant as possible in a greedy way, and the optimized solution.

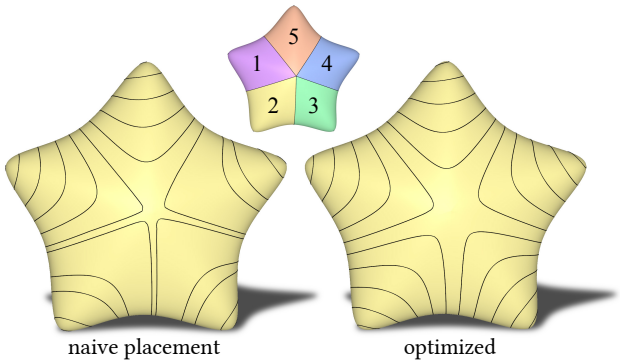


Fig. 15. The difference between a naive transition between cylinders, which tries to keep the zippable width as constant as possible with a greedy approach, and the optimized solution. Note that the optimized result appears to be more uniform. The numbers depict the cylinder transition order of the zipper-curve.

3.4 Cutting, remeshing and flattening

The goal of the final stage of our algorithm is to find a developable surface that approximates the input 3D shape well and has the curve computed in the previous stage as the boundary. This is a challenging task in general (see, e.g., [Rose et al. 2007; Tang et al. 2016]), but somewhat simpler in the discrete setting. It is well known that a triangle mesh in 3D that has no internal vertices, i.e., all its vertices are boundary vertices, is developable. Fitting a developable triangulation is still a difficult problem, where the challange lies in finding a meshing that appears smooth. Mitani and Suzuki [2004] proposed to use edge collapse and vertex removal operations until no internal vertices are left, and then to apply edge flip operations in order to improve the smoothness of the triangulation. We have implemented their method, but observed that the greedy edge flipping can introduce triangle fans near sharp curve turns (see Fig. 16). We instead propose a simple approach based on the parameterization we already have from previous stages. The idea is to define correspondences between points on adjacent windings of the spiral, which, when connected by straight lines, act as rulings of a developable surface. The simple correspondence we choose is based on the x coordinates of the lines in each C_i's parameterization (Fig. 17). We sample these lines uniformly and connect two samples in adjacent windings when their x coordinates are the same. We use Triangle [Shewchuk 1996] to complete the triangulation in the parametrization space in regions where there is no natural correspondence, that is, around the interfaces between cylinders and the open boundaries. Although this simple approach is not always optimal (see Fig. 24), we found it to deliver sufficient results for all cases. Once the zippable is triangulated, we can unfold it to the plane (see Fig. 5). We cut it into few smaller pieces in order to utilize the laser cutter bed better and resolve overlaps. The minimal number of cuts is related to the number of segments (parts from different segments might intersect) and the number of turns in each segment (different turns in the same part might intersect). See the attached cutting plans in the supplementary material.

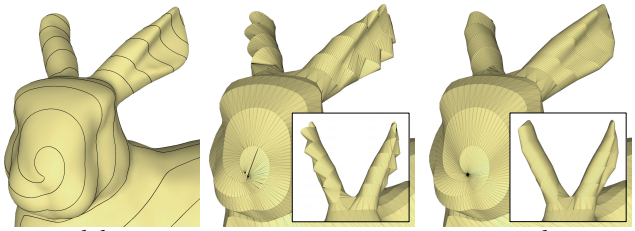


Fig. 16. A comparison of our simple ribbon meshing approach to the method in [Mitani and Suzuki 2004]: The greedy edge flipping step can generate non-optimal triangle fans (middle column), whereas ours results in a smoother and better approximation of the original surface. But generally, it is not guaranteed to create an optimal meshing (see Fig. 24).

4 FABRICATION

Zipper tape. The previous section describes how to design and compute a zipper-curve on the surface, but ignores the physical properties of the zipper itself. This is sufficient if one wishes to make papercraft, as in Fig. 22, since the zipper-curve has a negligible width. Accommodating a real zipper requires additional modeling. In general, zippers are made from two fabric tapes with a row of teeth on each (Fig. 18), which interlock or split when operating the slider. The common way to attach zippers to fabric is to sew the two tapes onto the matching edges of two parts. The teeth should slightly protrude to create a wide enough gap between the two pieces of fabric, so that the slider can move freely without getting stuck. This means that we must slightly offset the borders of the designed zippable in order to create that gap. We discuss offsetting in later subsections.

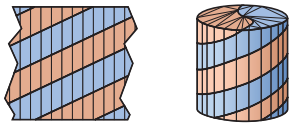


Fig. 17. Illustration of remeshing to a developable zippable. In the parameter domain, points on adjacent lines with the same x coordinate are connected by an edge.

Boundary alignment. To ensure that corresponding borders are aligned with each other through the zipper, we place markers (e.g., small circles) at regular intervals along the boundary of the zippable and on the zipper tape. We use 3 cm intervals; the markers can be etched by the laser cutter. When attaching the zipper, care must be taken to align the markers on the zippable with the ones on the zipper tape. After the zipper is attached, we align the starting points of the two sides of the zipper and put the slider in to obtain the final result.

Overlapping zippables. If the 2D layout of the zippable contains any overlapping regions or is too big for the available laser cutter or fabric, we separate it by cutting into intersection-free parts. These parts are then marked for etching with corresponding letters, arranged efficiently in the plane, laser-cut and sewn back together.

4.1 Attaching the zipper.

In this section we describe the fabrication process in more detail. We propose two different methods that differ by the offsetting procedure and the zipper attachment technique, with each method having its own benefits and drawbacks. The first approach is based on sewing the zipper to the fabric, and the second approach is based on gluing and requires the zipper and fabric to be mounted on a bespoke fastening rig. The sewing approach can be used for larger target sizes, where the width of the zipper can be largely neglected. It requires more expertise and manual work, but allows for more flexibility in fabrication. The second technique is particularly handy when the zipper width is not negligible, or for fabrication in an assembly line, since it comprises several sequential steps.

Attaching the zipper by sewing. To the best of our knowledge, there are currently no devices for automated sewing of zippers along a curved path, so this must be done manually using a sewing machine. Sewing on a curve is somewhat complicated since zippers are usually designed to have zero geodesic curvature. Their straight

tapes resist bending in the plane (although bending out of plane is possible, as we discuss in the following section). To afford this bending, the tape must be cut every few centimeters, as shown in Fig. 18.

2D-Offsetting. Since the zipper teeth have to protrude by a certain distance w, we must compensate for that by offsetting the boundary curve of the zippable by the same amount. This can be easily done in the plane using a standard curve offset in the normal direction by distance w. The only caveat is that this changes the boundary length, such that markers on the offset curve are no longer 3 cm apart. As a consequence, while sewing, the tailor needs to ensure that the markers on the zipper and the fabric still line up by slightly squeezing the fabric or the zipper tape, or cutting them, as shown in Fig. 18. For larger objects, the resulting inaccuracies are fairly small, as can be seen in the results in Figures 4, 8, 3, 5, which were all created using this approach. The fabricated shapes remain faithful to the design.

Attaching the zipper by welding or gluing. In modern clothing industry, zippers are often attached by welding or gluing to the underlying fabric, especially in outdoor, waterproof garments and equipment. This can be beneficial for the fabrication on an assembly line, since the different working steps can be split better than for sewing, which is still one of the most manual labor-intensive industries today. Gluing may also be a preferable technique for makers with no sewing experience or the necessary machines. It requires the fabric and the zipper to be first placed and secured in correct alignment before applying the necessary pressure to fixate it. Therefore, one can no longer assume that local squeezing is freely permitted, which calls for a different offsetting strategy.

Length preserving offsets. Forbidding the zipper (and the fabric) from stretching means that it can only undergo isometric deformations, which prohibits the planar curve offsetting we proposed for the sewing method. However, in practice and as shown in Fig. 18, the zipper has the ability to bend out of plane. Therefore, we consider the offsetting in 3D (see Fig. 19). In reality, the zipper tape can also easily twist and shear a bit, so instead of precisely isometric deformation, we make a simplification and assume that the two sides of the zipper, or equivalently the two sides of the offset boundary curves of the zippable, only need to have the *same length* to be a

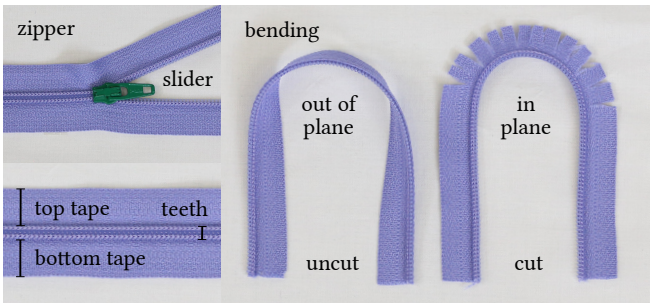


Fig. 18. A dissection of a zipper. Note that the zipper resists bending in the plane, unless cut every few centimeters to allow the zipper tape to stretch.

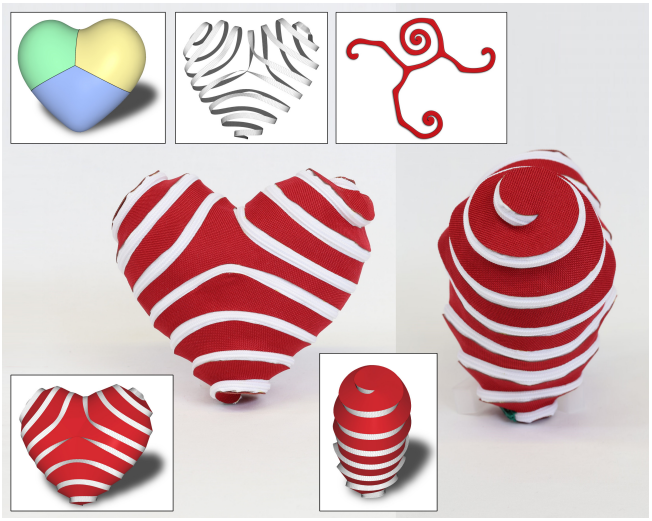


Fig. 19. The heart is our smallest physical result with a height of only 15cm. It was fabricated using our fastening rig in only 1 hour. The virtual model (insets in the bottom) is faithfully reproduced. The top insets show the segmentation, the visualization of the zipper tape and flattened zippable, from left to right.

valid zipper configuration. Given the zipper-curve on the surface in 3D, we seek two curves that are offset by the same amount in *opposite directions*. We propose to use the *binormal* of the zipper-curve as the offset direction. We prove in Appendix A that in the continuous setting, the offsetting in the positive and negative direction is guaranteed to keep the lengths of the two offset curves equal, and we bound the length difference between the zipper-curve and the offset curves. In practice, this small difference can be compensated by a slight buckling of the zipper tape. In the discrete setting, we estimate the binormal by the cross product of two adjacent segments of the zipper-curve. Note that the binormal direction for a straight line is not well defined, and so in regions where the polyline is almost straight, we might get numerically noisy results. To avoid this, we consider a bigger window of adjacent segments to find a stable estimation of the binormal at these locations, at the cost of a small deviation in length. To validate our results, we compute the relative change in length that occurs due to the approximation. For the heart shape in Fig. 19, we use 2.25 meters of zipper, and our error of 0.2 mm is negligible. Two results with this type of offsetting are shown in Figures 19, 2.

The fastening rig. Central to the gluing approach is a bespoke fastening rig we developed. The idea is to secure the zipper and the zippable in place before applying contact glue and pressing them together. In order to glue the zipper tape completely flat onto the zippable, one needs to cut it at regular intervals. The important difference to sewing is that this does not result in any buckling or deformation of the zippable and is completely hidden behind it. The rig is constructed from 3 layers of hard sheets (e.g., acrylic glass or plywood) as shown in Fig. 20. The bottom one serves as a baseplate to stabilize the others. We cut tracks into the middle sheet for sliding

in the zipper tapes. The tracks have the shape of the boundary of the flattened zippable and a width of about 4 mm. The teeth track is curved, but it only bends the zipper tape out of plane. The cut in the top layer is slightly offsetted, such that the zipper tape can pass through the gap but the teeth are held down. We also add an opening for the tracks to simplify the sliding in of the zipper in parts. The layers are connected and secured with screws. Even though the rig does a good job in aligning the zipper tape to the zippable, we still add markers in 5 cm intervals on the zipper tape and the rig as a detailed alignment guidance. If the zippable has overlaps, the tracks for the shape can also be split into multiple parts as before, which then only requires to slide in and glue the zipper to the zippable part by part (the result in Fig. 2 was produced in this way). We found that this adds no significant time to the fabrication process. See Fig. 20 and the accompanying video for more detail.

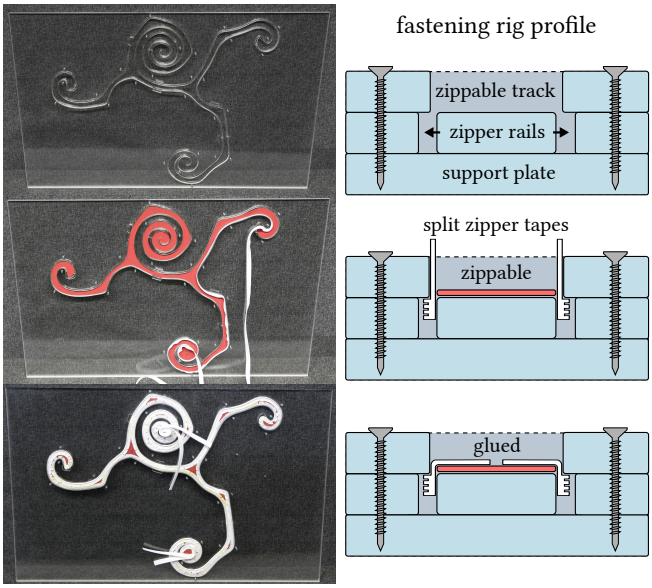


Fig. 20. An overview of the fabrication with the fastening rig. From top to bottom: the empty assembled rig; rig with inserted zippable and partly slid in zipper tape; the completed zippable. Sliding in the zipper tapes is easy and fast. The tracks in the fastening rig almost automatically take care of the correct alignment of the tape to the zippable. Gluing is straight-forward and we refer to the accompanying video for a demonstration.

5 RESULTS AND DISCUSSION

We implemented all parts of our algorithm in C++, except the parameterization, which was implemented in MATLAB. The modified Newton's method converges within at most 15 iterations and takes about 20 seconds for a mesh of 30k triangles on our Xeon CPU E5-2650 v2, 64 GB RAM machine.

Fabrication process. We fabricated seven of our designs in fabric, see Figures 1, 2, 3, 4, 5, 8, 19. Fig. 5 shows the steps of our method with our simplest model, the T shape. All fabricated results have a zipper length of about 10 meters and a maximum dimension of about 50 cm, except the heart (Fig. 19) with 3.5 m zipper and 15 cm height, and the star (Fig. 2) with 2.25 m zipper and 27 cm height. These last two are considerably smaller and were fabricated with our gluing technique. The sewn results were made by professional tailors and took 5 to 6 hours for each model. The time needed mainly depends on the length of zipper-curve. Even though the tailors had no experience with this special kind of fabrication, there were no problems in attaching the zippers to the cut fabric pieces thanks to the linearity of the assembly method. In comparison, the fabrication of the results made with the bespoke fastening rig approach took us 1 hour for the heart and 1.5 hours for the star shape. Cutting and constructing the fastening rig amounts to about 30 min overhead, and the rig can serve the making of many copies of the same shape.

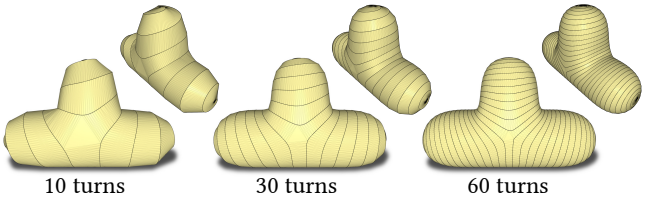


Fig. 21. Running our method on the T shape with different spiral densities.

Design process. We designed all the presented examples ourselves. Simple designs like the heart (Fig. 19) or the star (Fig. 2) can be made within only two to three minutes. More complicated models like the octopus (Fig. 23) can take up to 15 minutes. Most time is spent iterating and refining the cylindrical segmentation in order to optimize the zipper-curve shape, since the global parameterization has to be recomputed each time. In the future, we would like to explore splines instead of straight lines in the parameterization to enable a more flexible zipper-curve design, possibly aligning it to geometric features or user prescribed directions. While the transition point optimization (Sec. 3.3) finds the best possible solution for a selected traversal order, it is not guaranteed to always create a nice, uniformly spaced zippable. This is due to the limited degrees of freedom of the corresponding traversal interfaces. In this case, the user can iterate through the other traversal orders to pick a better choice. In the future, it would be interesting to incorporate an automatic search to find the best possible traversal order in terms of uniformity of the zippable. Another useful feature would be a way to bound the maximal curvature of the zipper-curve to make it smoother.

Papercraft comparison. Our method can be used to create papercraft models. In general, it produces fewer initial pieces than previously published methods, and the assembly by gluing is straightforward and does not require elaborate instructions. See Fig. 22 for a comparison of our bunny result fabricated from paper with the method of [Mitani and Suzuki 2004]. Note that even though our result is somewhat finer in terms of the width of the parts, it is fabricated from fewer pieces.

Approximation capabilities. Naturally, the thinner the designed zippable, the better its approximation power. However, this also results in a longer zippable, prolonging the fabrication. The decision

regarding the sizing is left to the user. See Fig. 21 for different widths and approximations of the T shape.

We attach cutting plans for our zippables, including the fastening rig for the heart shape, in the supplemental material.

6 CONCLUSION

We presented a method for shape representation by a single developable surface that can be fabricated from flat fabric. We show several examples to demonstrate the power and generality of our approach. Currently, we do not attempt to align the zipper-curve to the input shape's features. This means that regions with sharp corners may not be well represented by the zippable, unless the user manually specifies it. We plan to tackle this issue in the future by using feature detection and incorporating it into the zipper-curve design stage. Additionally, we are interested in targeting more global objectives, such as symmetry. Furthermore, we would like to combine the zipper-curve design with the final stage of remeshing to a developable. Currently these steps are strictly decoupled, and we expect to get better approximation quality by optimizing both parts simultaneously. Another interesting direction would be to integrate the optimization into the segmentation stage, such that the design of the final zipper-curve can be done more interactively.

Fabrication with textiles, especially woven fabrics, is a very prolific but currently notoriously human labor-intensive industry, in dire need of digitalization and automation. We plan to explore further automation and acceleration of our fabrication process and expand to other types of fabrication methods and applications. One application we are particularly interested in is *pipe cladding*, which is part of the process of insulating heated pipes with metal sheets, and is a labour intensive task. The process is similar to our zipping, but usually performed manually by an expert; our method could be potentially used to greatly simplify and speed up this task.

Fig. 22. Our laser cut plans of the bunny with 7 pieces (bottom row) for a single developable piece that can be assembled by linearly gluing its border, starting at the designated red point, compared to [Mitani and Suzuki 2004] (taken from their paper) with 15 pieces which require more detailed instructions to be glued together (top row). Note that even though our result is somewhat finer in terms of the width of the parts, it is fabricated from fewer pieces.

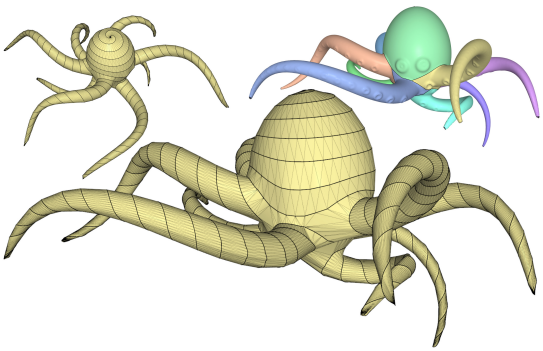


Fig. 23. A virtual result of an octopus model segmented into nine cylindrical parts. The zippable has a nice uniform spacing but some of the geometric details, like the eyes, are lost due to the limited resolution.

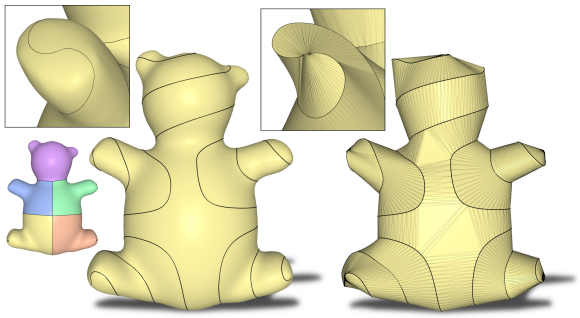


Fig. 24. Our naive meshing algorithm is not guaranteed to produce an optimal approximation of the original surface, especially for a sparse coverage of the target model with the zipper-curve.

ACKNOWLEDGMENTS

We are grateful to Oliver Glauser, Michael Rabinovich and Renana Poranne for invaluable discussions and help with the video. We would like to thank Isabelle von Salis, the tailors from *jaelsigner.ch* and *atelier-renaissance.ch* for their help with the fabrication of the prototypes. The work was supported in part by the European Research Council under Grant No.: StG-2012-306877 (ERC Starting Grant iModel).

REFERENCES

N. Aigerman and Y. Lipman. 2015. Orbifold Tutte Embeddings. *ACM Trans. Graph.* 34, 6, Article 190 (Oct. 2015), 12 pages. https://doi.org/10.1145/2816795.2818099

D. Barnette and E. Jucovič. 1970. Hamiltonian circuits on 3-polytopes. *Journal of Combinatorial Theory* 9, 1 (1970), 54 – 59. https://doi.org/10.1016/S0021-9800(70)80054-0

B. Bickel, M. Bächer, M. A. Otaduy, H. R. Lee, H. Pfister, M. Gross, and W. Matusik. 2010. Design and Fabrication of Materials with Desired Deformation Behavior. *ACM Trans. Graph.* 29, 4, Article 63 (July 2010), 10 pages. https://doi.org/10.1145/1778765.1778800

D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, and D. Zorin. 2013. Quad-Mesh Generation and Processing: A Survey. *Comput. Graph. Forum* 32, 6 (2013), 51–76. https://doi.org/10.1111/cgf.12014

S. Chandra, A. Körner, A. Koronaki, R. Spiteri, R. Amin, S. Kowli, and M. Weinstock. 2015. Computing curved-folded tessellations through straight-folding approximation. In *Proc. Symposium on Simulation for Architecture & Urban Design.* http://dl.acm.org/citation.cfm?id=2873042

X. Chen, C. Zheng, W. Xu, and K. Zhou. 2014. An Asymptotic Numerical Method for Inverse Elastic Shape Design. *ACM Trans. Graph.* 33, 4 (2014).

D. Eppstein and M. Gopi. 2004. Single-strip triangulation of manifolds with arbitrary topology. In *Proc. Symp. Computational Geometry.* 455–456. https://doi.org/10.1145/997817.997888

A. Fondevilla, A. Bousseau, D. Rohmer, S. Hahmann, and M. Cani. 2017. Patterns from photograph: Reverse-engineering developable products. *Computers & Graphics* 66 (2017), 4–13. https://doi.org/10.1016/j.cag.2017.05.017
K. Hormann, B. Lévy, and A. Sheffer. 2007. Mesh parameterization: theory and practice. In *ACM SIGGRAPH Courses*. https://doi.org/10.1145/1281500.1281510
Y. Igarashi and T. Igarashi. 2008. Pillow: Interactive Flattening of a 3D Model for Plush Toy Design. In *Proc. Smart Graphics*. https://doi.org/10.1007/978-3-540-85412-8_1
Y. Igarashi, T. Igarashi, and H. Suzuki. 2009. Interactive Cover Design Considering Physical Constraints. *Comput. Graph. Forum* 28, 7 (2009), 1965–1973. https://doi.org/10.1111/j.1467-8659.2009.01575.x
D. Julius, V. Kraevoy, and A. Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. *Comput. Graph. Forum* 24, 3 (2005), 581–590.
A. Jung, S. Hahmann, D. Rohmer, A. Bégault, L. Boissieux, and M. Cani. 2015. Sketching Folds: Developable Surfaces from Non-Planar Silhouettes. *ACM Trans. Graph.* 34, 5 (2015), 155:1–155:12. https://doi.org/10.1145/2749458
F. Kälberer, M. Nieser, and K. Polthier. 2011. Stripe parameterization of tubular surfaces. In *Topological Methods in Data Analysis and Visualization*. Springer, 13–26.
M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann. 2008. Curved folding. *ACM Trans. Graph.* 27, 3 (2008). https://doi.org/10.1145/1360612.1360674
F. Knöppel, K. Crane, U. Pinkall, and P. Schröder. 2015. Stripe patterns on surfaces. *ACM Trans. Graph.* 34, 4 (2015). https://doi.org/10.1145/2767000
S. Kolmanič and N. Guid. 2002. From Geometric Modeling to Shape Modeling. In *Proc. Workshop on Geometric Modeling: Fundamentals and Applications*. 35–46. http://dl.acm.org/citation.cfm?id=512296.512302
S. Z. Kovalsky, M. Galun, and Y. Lipman. 2016. Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.* 35, 4 (2016), 134. https://doi.org/10.1145/2897824.2925920
Y. Liu, Y. Lai, and S. M. Hu. 2009. Stripification of Free-Form Surfaces With Global Error Bounds for Developable Approximation. *IEEE Trans. Automation Science and Engineering* 6, 4 (2009), 700–709. https://doi.org/10.1109/TASE.2008.2009926
M. Livesu, M. Attene, G. Patanè, and M. Spagnuolo. 2017. Explicit cylindrical maps for general tubular shapes. *Computer-Aided Design* (2017). https://doi.org/10.1016/j.cad.2017.05.002 Accepted for publication.
A. Lubiw, E. D. Demaine, M. L. Demaine, A. Shallit, and J. Shallit. 2010. Zipper unfoldings of polyhedral complexes. In *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada, August 9-11, 2010*. 219–222. http://cccg.ca/proceedings/2010/paper58.pdf
A. Mahdavi-Amiri, P. Whittingham, and F. Samavati. 2015. Cover-it: an interactive system for covering 3D prints. In *Proc. Graphics Interface*. 73–80. https://doi.org/10.20380/GI2015.10
T. Martin, E. Cohen, and M. Kirby. 2008. Volumetric parameterization and trivariate b-spline fitting using harmonic functions. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling, Stony Brook, New York, USA, June 2-4, 2008*. 269–280. https://doi.org/10.1145/1364901.1364938
F. Massarwi, C. Gotsman, and G. Elber. 2007. Papercraft Models using Generalized Cylinders. In *Proc. Pacific Graphics*. https://doi.org/10.1109/PG.2007.16
J. Mitani and H. Suzuki. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph.* 23, 3 (2004), 259–263. https://doi.org/10.1145/1015706.1015711
Y. Mori and T. Igarashi. 2007. Plushie: an interactive design system for plush toys. *ACM Trans. Graph.* 26, 3 (2007). https://doi.org/10.1145/1276377.1276433
A. Myles and D. Zorin. 2012. Global parametrization by incremental flattening. *ACM Trans. Graph.* 31, 4 (2012), 109:1–109:11. https://doi.org/10.1145/2185520.2185605
J. O'Rourke. 2015. Spiral Unfoldings of Convex Polyhedra. *arXiv preprint arXiv:1509.00321* (2015).
H. Pedersen. 2011. Methods for Creating Developable Surfaces. (July 14 2011). https://www.google.ch/patents/US20110169828 US Patent App. 13/005,384.
H. Pedersen and K. Singh. 2006. Organic Labyrinths and Mazes. In *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering (NPAR '06)*. ACM, New York, NY, USA, 79–86. https://doi.org/10.1145/1124728.1124742
M. Rabinovich, R. Poranne, D. Panozzo, and O. Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2 (2017), 16:1–16:16.
N. Ray, W. Li, B. Lévy, A. Sheffer, and P. Alliez. 2006. Periodic global parameterization. *ACM Trans. Graph.* 25, 4 (2006). https://doi.org/10.1145/1183287.1183297
K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert. 2007. Developable Surfaces from Arbitrary Sketched Boundaries. In *Proc. Symp. Geom. Processing*. 163–172.
J. Rossignac. 1999. Edgebreaker: Connectivity Compression for Triangle Meshes. *IEEE Trans. Vis. Comput. Graph.* 5, 1 (1999), 47–61. https://doi.org/10.1109/2945.764870
I. Shatz, A. Tal, and G. Leifman. 2006. Paper craft models from meshes. *The Visual Computer* 22, 9-11 (2006), 825–834. https://doi.org/10.1007/s00371-006-0067-6
J. R. Shewchuk. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*. Lecture Notes in Computer Science, Vol. 1148. 203–222.
A. Shtengel, R. Poranne, O. Sorkine-Hornung, S. Z. Kovalsky, and Y. Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Trans. Graph.* 36, 4, Article 38 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073618
M. Skouras, B. Thomaszewski, B. Bickel, and M. H. Gross. 2012. Computational Design of Rubber Balloons. *Comput. Graph. Forum* 31, 2 (2012), 835–844. https://doi.org/10.1111/j.1467-8659.2012.03064.x
M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, and M. Gross. 2013. Computational design of actuated deformable characters. *ACM Trans. Graph.* 32, 4 (2013), 82:1–82:10.
M. Skouras, B. Thomaszewski, P. Kaufmann, A. Garg, B. Bickel, E. Grinspun, and M. H. Gross. 2014. Designing inflatable structures. *ACM Trans. Graph.* 33, 4 (2014). https://doi.org/10.1145/2601097.2601166
J. Smith and S. Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (2015), 9 pages. https://doi.org/10.1145/2766947
R. Straub and H. Prautzsch. 2011. *Creating optimized cut-out sheets for paper models from meshes*. Technical Report. Karlsruhe Institute of Technology.
S. Takahashi, H. Wu, S. H. Saw, C. Lin, and H. Yen. 2011. Optimized Topological Surgery for Unfolding 3D Meshes. *Comput. Graph. Forum* 30, 7 (2011), 2077–2086. https://doi.org/10.1111/j.1467-8659.2011.02053.x
C. Tang, P. Bo, J. Wallner, and H. Pottmann. 2016. Interactive Design of Developable Surfaces. *ACM Trans. Graph.* 35, 2, Article 12 (2016), 12 pages. https://doi.org/10.1145/2832906
M. Tarini. 2012. Cylindrical and Toroidal Parameterizations Without Vertex Seams. *J. Graphics Tools* 16, 3 (2012), 144–150. https://doi.org/10.1080/2151237X.2012.654054
J. Thiery, B. Buchholz, J. Tierny, and T. Boubekeur. 2012. Analytic Curve Skeletons for 3D Surface Modeling and Processing. *Comput. Graph. Forum* 31, 7-2 (2012), 2223–2232. https://doi.org/10.1111/j.1467-8659.2012.03215.x
C. C. L. Wang. 2010. From Designing Products to Fabricating Them from Planar Materials. *IEEE Computer Graphics and Applications* 30, 6 (2010), 74–85. https://doi.org/10.1109/MCG.2009.155
L. Zeng, Y. Liu, M. Chen, and M. M. Yuen. 2012. Least squares quasi-developable mesh approximation. *Computer Aided Geometric Design* 29, 7 (2012), 565–578. https://doi.org/10.1016/j.cagd.2012.03.009
H. Zhao, F. Gu, Q. Huang, J. A. G. Galicia, Y. Chen, C. Tu, B. Benes, H. Zhang, D. Cohen-Or, and B. Chen. 2016. Connected Fermat spirals for layered fabrication. *ACM Trans. Graph.* 35, 4 (2016), 100:1–100:10. https://doi.org/10.1145/2897824.2925958
Y. Zhou, K. Yin, H. Huang, H. Zhang, M. Gong, and D. Cohen-Or. 2015. Generalized Cylinder Decomposition. *ACM Trans. Graph.* 34, 6 (2015), Article 171.
zipit. 2017. zipit store online. http://www.zipitstore.com/. (2017). Accessed: 2017-02-01.

A OFFSETTING

We show that offsetting a curve in the direction of its binormal and the negative of that direction by the same amount results in two curves of the same length.

Proposition A.1. *Let $\gamma(s)$ be an arc-length parameterized curve, and $\tau(s)$ and $B(s)$ its torsion and binormal at s. Assume that $\tau(s) \neq 0$. Define $\gamma_\pm(s) := \gamma(s) \pm w\, B(s)$, where w is the offset amount. Then $\forall s,\ \|\gamma'_+(s)\| = \|\gamma'_-(s)\|$.*

Proof. By applying the Frenet-Serret formula we obtain

$$\gamma'_\pm(s) = T(s) \mp w\, \tau(s) N(s),$$

where $T(s)$, $N(s)$ are the tangent and the normal of γ at s. The result immediately follows by using the polarization identity:

$$\|T(s) + w\, \tau(s) N(s)\|^2 - \|T(s) - w\, \tau(s) N(s)\|^2 =$$
$$= 4 \left\langle T(s),\ w\, \tau(s) N(s) \right\rangle = 0,$$

where the last equality is due to $T(s) \perp N(s)$. □

The result above also leads to a bound on the local change of length, i.e., speed, of the offset curve. Indeed, using the triangle inequality,

$$\|\gamma_\pm(s)\| \leq \|T(s)\| + \|w\, \tau(s) N(s)\| = 1 + w\, \tau(s),$$

which also means that the speed is determined by the torsion $\tau(s)$. Since the zipper-curve we design usually have a helical shape, we believe that $\tau(s)$ is kept relatively small compared to a random path on the surface.