Designing Structurally-Sound Ornamental Curve Networks

Jonas Zehnder[1] Stelian Coros[2] Bernhard Thomaszewski[1]

[1]Disney Research [2]Carnegie Mellon University

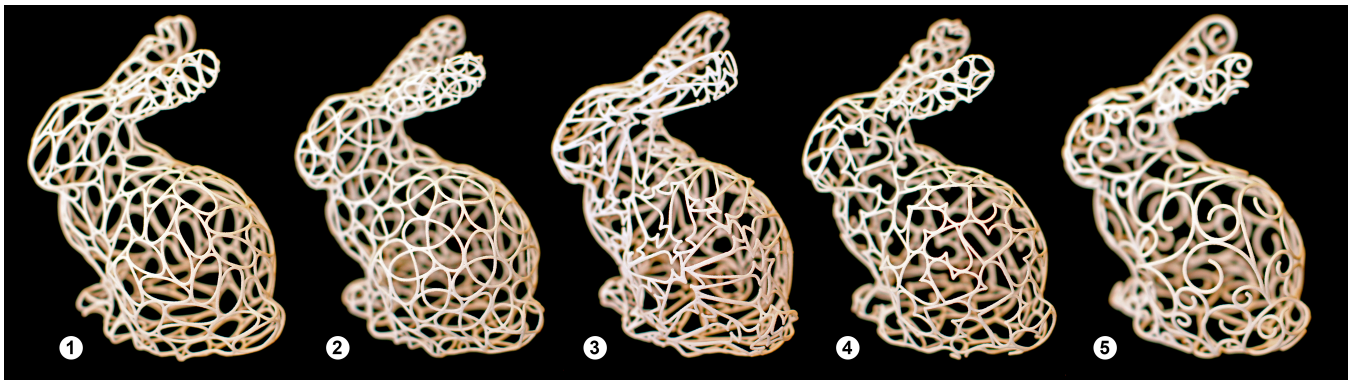


Figure 1: *Using our design tool, users can create structurally-sound curve networks over complex surfaces with a broad variety of aesthetic appeals.* (1-4) *Deformable packings created in a semi-automatic way.* (5) *A manually-designed branching structure.*

Abstract

We present a computational tool for designing ornamental curve networks—structurally-sound physical surfaces with user-controlled aesthetics. In contrast to approaches that leverage texture synthesis for creating decorative surface patterns, our method relies on user-defined spline curves as central design primitives. More specifically, we build on the physically-inspired metaphor of an embedded elastic curve that can move on a smooth surface, deform, and connect with other curves. We formalize this idea as a globally coupled energy-minimization problem, discretized with piece-wise linear curves that are optimized in the parametric space of a smooth surface. Building on this technical core, we propose a set of interactive design and editing tools that we demonstrate on manually-created layouts and semi-automated deformable packings. In order to prevent excessive compliance, we furthermore propose a structural analysis tool that uses eigenanalysis to identify potentially large deformations between geodesically-close curves and guide the user in strengthening the corresponding regions. We used our approach to create a variety of designs in simulation, validated with a set of 3D-printed physical prototypes.

Keywords: Curve Networks, Computational Design, 3D Printing

Concepts: •**Computer graphics** → **Computational geometry and object modeling;** *Physically based modeling;*

SIGGRAPH '16 Technical Paper,, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925888

1 Introduction

Ornamental patterns can transform a common object into a work of art, beautiful and unique. The decorative arts abound with examples of such artifacts, including wood and bone carvings, wire wrappings, intricate jewelery pieces and many other types of valuables. Traditionally, creating designs of such complexity required the steady hand of a skilled and experienced craftsman. Nowadays, with the advent of digital fabrication, the general public has the means to create their own exquisitely decorated objects. However, while manufacturing processes are ever more accessible, designing ornate artifacts that are aesthetically-pleasing and strong enough for fabrication, shipment and handling is a challenging task. Indeed, many fine pieces of craftsmanship exhibit an intimate relation between structure and ornament. As the natural product of this coalescence, we focus on *ornamental curve networks*—aesthetically-pleasing surface representations that convey a sense of delicacy and filigree.

To facilitate the creation of ornamental curve networks that depict arbitrary surfaces, we seek an assisstive tool that provides artistic freedom while automating tedious tasks as much as possible. Taking wrought-iron works as inspiration, our design tool allows the user to decorate input shapes with networks of interconnected curves. In order to lay down an aesthetic language for the design, curves are drawn from a user-defined space and instantiated on the input surface one-by-one or through higher-level layout tools. At any time, existing curves can be repositioned, reoriented or rescaled by the user. Such editing operations are propagated through the entire structure, as curves aim to retain their original shape as much as possible while conforming to the embedding surface and interacting with other curves in physically-meaningful ways.

In addition to promoting creativity, our computational tool must help to ensure that the resulting curve network is stable. To uphold the aesthetic style of the designs, we assume structural stability can be controlled solely through the way in which curves connect to each other, as opposed to, for example, thickening different parts of the network. To guide the user in strengthening structurally-weak regions of their design, our system identifies and visualizes geodesically-close curves that undergo large relative deformations

under worst-case loading scenarios. Based on this information, the user can then improve the stability of the design through simple interactions.

Our approach aims to enable a simple design experience, but implementing the above concepts requires a significant level of technical sophistication. In order to map planar curves to arbitrary surfaces, prevent unwanted intersections, handle contacts, and support connectivity-preserving editing operations, we cast curve network design as a globally-coupled energy minimization problem. To solve this problem efficiently, we turn to the smooth setting of subdivision surfaces, where continuous surface derivatives can be leveraged for fast Newton-type minimization. In particular, we achieve low-distortion curve embeddings by minimizing a shape-preserving energy, inspired by Euler elastica, in the parametric space of the underlying smooth surface. Furthermore, we describe an efficient way to determine the worst-case stability of a curve network through eigenanalysis. We demonstrate our method on a set of digital objects and physical prototypes that exhibit a broad range of ornamental styles, each with its distinct aesthetic appeal.

2 Related Work

Texture Synthesis Creating decorative patterns on digital surfaces has a long history in graphics. A general class of approaches are example-based texture synthesis methods that produce seamless output from a small input example; see Wei et al. [2009] for an overview of the large body of literature on this subject. Dumas et al. [2015] recently extended this idea to the context of digital fabrication as a way of stylizing 3D-printed objects. Our method creates sparse networks from spline-based example curves and could thus be considered a vector-based analogon of the rasterization method by Dumas and colleagues. More importantly, however, our approach puts the user at the heart of the design loop, offering a much higher degree of artistic control.

Tilings and Distributions Another way of creating decorative patterns is through tiling or sampling. For example, the method by Kaplan and Salesin [2000] deforms a single complex shape to seamlessly tile the plane. Kim and Pellacini [2002] use sets of arbitrarily-shaped tiles taken from photographs in order to create jigsaw mosaics in complex-shaped 2D domains. While their method allows gaps between tiles, the approach presented by Peng et al. [2014] achieves gap-less layouts by deforming template tiles. These methods aim to cover the output domain in a seamless way, but other works have focused on less tight object distributions that exhibit specific local [Ma et al. 2011] or global [Reinert et al. 2013] arrangements. As one way of creating curve networks with our tool, we draw inspiration from object distribution methods in order to create deformable curve packings with artistic control.

Curve Network Synthesis A number of works have investigated the problem of creating weaving or otherwise interleaving patterns on surfaces. Kaplan and Cohen [2003] describe a procedural approach for creating Celtic knot patterns over arbitrary surfaces. In a more general setting, the method by Akleman et al. [2009] allows users to create cyclic plain-weaving patterns. Zhou et al. [2006] propose an example-based method to directly synthesize 3D geometry such as basket-like weaves around a surface mesh.

Another line of work has considered the automatic synthesis of (2D) branching curve networks. Merrell and Manocha [2010] described a method for generating branching structures by progressively inserting sub-parts of an input curve. Xu and Mould [2009] create specific types of branching networks by tracing particles through artificial force fields. Runions et al. [2005] propose a biologically-inspired method for creating leaf venation patterns. While each of the above methods creates a particular type of curve network, we seek a more general approach that provides explicit user control over the shape of the curves and the connectivity of the network in order to accommodate a large variety of styles.

Curves on Surfaces While spline curves in Euclidean space have been studied for a long time, the problem of modeling smooth curves on manifolds has only recently received attention from the graphics community. Hofer and Pottman [2004] compute energy-minimizing spline curves on manifolds using constrained minimization with a gradient projection approach. Wallner et al. [2005] subsequently extended this approach to spline networks. Starting directly in the discrete setting, Lee and Lee [2002] propose geometric snakes, an active contour model for feature detection on triangle meshes. Geometric snakes are computed by minimizing feature and shape preservation energies. The minimization is performed in 2D by virtue of a (constantly updated) local parameterization of the corresponding surface region. As another application, Campen et al. [2014] used embedded elastic curves for interactive quad meshing. Our approach for modeling discrete embedded elastica is similar, but the application to 3D-printable curve networks that satisfy aesthetic and structural constraints leads to a very different set of challenges.

In this context, we also mention the method by Stam [2003] for simulating fluid flows on smooth manifolds represented by Loop subdivision surfaces. While our approach relies on the same discrete representation for smooth surfaces, we consider the problem of simulating embedded elastic curves for the problem domain of fabrication-oriented design.

Fabrication-Oriented Design The graphics community has seen a steady increase in design tools for various forms of physical surfaces, including plush toys [Mori and Igarashi 2007], inflatables [Skouras et al. 2012; Skouras et al. 2014], bead-work models [Igarashi et al. 2012], wire-mesh sculptures [Garg et al. 2014], and others. From an aesthetic point of view, our work is also related to the design tool for wire-wrap jewelry by Iarussi et al. [2015].

In the context of 3D-printing, several methods that leverage physics-based simulation in order to analyze and improve the stability of 3D-printed artifacts [Stava et al. 2012; Umetani and Schmidt 2013] while minimizing material use [Lu et al. 2014] have been proposed. Most closely related to our approach, Zhou et al. [2013] determine worst-case force distributions using eigenanalysis and highlight areas of high stress on input 3D models. Our method also relies on eigenanalysis, but instead of visualizing stress concentrations, we identify undesired deformations between geodesically-close curves and guide the user in strengthening the structure.

The types of structures that we target in this work can be modeled as networks of discrete elastic rods [Bertails et al. 2006; Bergou et al. 2008; Spillmann and Teschner 2009]. Our approach for predicting deformations within curve networks is most closely related to the one described by Prez et al. [2015]. However, while they optimize existing curve structures with fixed topology, our method facilitates the design and creation of arbitrary curve networks.

Another line of research has started to explore the connection between aesthetics and structural stability: Dumas et al. [2015] create structurally-sound patterns for 3D-printable surfaces using an example-based texture synthesis approach; Martinez et al. [2015] simultaneously optimize for stability, material use, and appearance using topology optimization; Zhou et al. [2014] optimize the layout of segments in order to create well-connected decorative curves from vector patterns. We share the goal of creating physical sur-

face representations that are aesthetically-pleasing and structurally-sound. However, by combining a curve-centric approach with a user-in-the-loop paradigm, our approach enables very different types of designs.

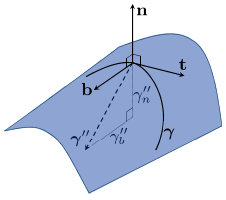
Concurrent to our work, Chen et al. [2016] present a method for creating physical surface representations using decorative patterns. Their approach is to pack an input surface with user-provided base elements that are optimized to overlap in a way that preserves aesthetic appearance while providing sufficient structural strength. While the method by Chen et al. [2016] offers a fully-automated design process that is particularly well-suited for regular structures, our system promotes user interaction and enables heterogeneous designs.

3 Elastic Curves on Surfaces

The underpinnings for our design tool are formed by a computational method for simulating the motion, deformation, and interaction of elastic curves embedded in smooth surfaces.

3.1 Variational Curve Mapping

As a central component of curve network design, we must define a way to map user-defined planar curves to arbitrary 3D surfaces. In seeking such a mapping, our primary goals are to induce minimal distortion and to ensure continuity in geometry as curves move across the surface. In order to achieve low distortion while, at the same time, permitting deformation as needed, we take a variational approach and model curves on surfaces as embedded elastica. These elastic curves can bend and stretch in order to comply with editing operations or contacts with other curves, but aim to preserve their original shape as much as possible. However, we can only expect shape-preservation to the extent to which the underlying surface allows it.



To explain this point further, we turn to the continuous setting and let $\boldsymbol{\gamma}(t)$ denote an arc-length parameterized curve on the surface, whose orthonormal frame $[\mathbf{t}, \mathbf{n}, \mathbf{b}]$ is defined by its tangent $\mathbf{t} = \boldsymbol{\gamma}'$, the normal of the surface $\mathbf{n}$, and the binormal $\mathbf{b} = \mathbf{t} \times \mathbf{n}$. At any point along the curve, the total curvature is given as $\kappa = |\boldsymbol{\gamma}''(t)|$. Noting that $|\boldsymbol{\gamma}'| = 1$ implies $\mathbf{t} \cdot \boldsymbol{\gamma}'' = 0$, the second derivative of the curve can be decomposed into components along the normal $\gamma_n'' = \mathbf{n} \cdot \boldsymbol{\gamma}''$ and binormal $\gamma_b'' = \mathbf{b} \cdot \boldsymbol{\gamma}''$, giving rise to the normal curvature $\kappa_n = |\gamma_n''|$ and geodesic curvature $\kappa_g = |\gamma_b''|$, respectively. As an intuitive interpretation, any embedded curve *must* assume the normal curvature imposed by the underlying surface, whereas its geodesic curvature is independent of the embedding. For an (initially straight) elastic curve embedded on a surface, this observation suggests an energy formulation of the form $\int \kappa_g(t)^2 dt$, as it only considers deviations in geodesic curvature and is *invariant* under normal curvature (see, e.g., [Langer and Singer 1996]). We note that penalizing the normal curvature term in the energy function would result in the embedded curve drifting towards regions of the surface where the total curvature is minimum, which is undesirable for our application.

3.2 Discretization

For numerical treatment, we need discrete representations for elastic curves as well as the surfaces to which they are confined. In order to enable interactive design and editing of curve networks, we would like to take advantage of efficient numerical methods that leverage derivative information. Since the smoothness of a curve's motion in Euclidean space is determined by the underlying surface, we combine higher-order surface representations with piece-wise linear curves.

Discrete Surface Representation Its applicability to arbitrary triangle meshes make the Loop subdivision scheme an appealing candidate for our setting. The Loop scheme defines smooth surfaces as the limit of repeated refinement and smoothing of an initial control mesh $\mathcal{M}$. Let $\mathbf{X}_i$ denote the vertices of the control mesh. The limit position of a point with barycentric (natural) parameters (u, v) within a given triangle $\mathcal{T}_k$ is computed as

$$\mathbf{x}^k(u, v) = \sum_{i=1}^{N+6} \varphi_i^k(u, v) \mathbf{X}_i^k \,, \tag{1}$$

where φ_i^k is the basis function of vertex i and $N + 6$ is the number of nodes in the one-ring neighborhood of $\mathcal{T}_k$. For regular triangles having only vertices of valence six (i.e., $N = 6$), the corresponding part of the limit surface is a quartic bezier patch whose twelve basis functions can be evaluated analytically. For triangles with irregular vertices ($N \neq 6$), we use the eigenbasis of the subdivision operator as proposed by Stam [1998] for efficient evaluation of the basis functions and their derivatives at arbitrary parameter values.

Discrete Embedded Elastica We represent curves on the limit surface through a set of vertices defined on the control mesh $\mathcal{M}$. In particular, for every vertex j of a given curve, we store the index k_j of the triangle it lies on as well as its two barycentric coordinates $0 \leq u_j, v_j \leq 1$ with $u_j + v_j \leq 1$. Curves can therefore be compactly represented as vectors $\mathbf{c}^i = (k_1, u_1, v_1, \ldots, k_n, u_n, v_n)^T$, where the superscript i represents the index of the curve in the network and is omitted for simplicity when referring to a generic curve. As the vertices of a curve glide along the underlying surface, we update both their barycentric coordinates and their triangle index as described in Sec. 3.5. The limit surface position of a vertex j on curve i is denoted by $\mathbf{c}_j^i = \mathbf{x}^{k_j}(u_j, v_j)$, while $\mathbf{c}^i(t)$ denotes the position of an arc-length parameterized point along the curve, determined through linear interpolation of its vertices.

In order to define energies for discrete embedded elastica, we adapt the formulation by Bergou et al. [2008; 2010]. Following this model, the integrated total curvature at a given vertex is captured by the curvature vector

$$\boldsymbol{\kappa}_i = \frac{2\mathbf{e}_{i-1} \times \mathbf{e}_i}{|\mathbf{e}_{i-1}||\mathbf{e}_i| + \mathbf{e}_{i-1} \cdot \mathbf{e}_i}, \tag{2}$$

where $\mathbf{e}_i = \mathbf{c}_{i+1} - \mathbf{c}_i$ are the edge vectors of the curve. By projecting the curvature vector onto the surface normal, we obtain an approximation of discrete geodesic curvature and define a corresponding energy term as

$$E_{\text{bend}} = \mu_{\text{bend}} \sum_i \frac{\left(\boldsymbol{\kappa}_i \cdot \mathbf{n}_i - \bar{\kappa}_i\right)^2}{\bar{l}_i} \,, \tag{3}$$

where $\bar{l}_i = \frac{1}{2}(|\bar{\mathbf{e}}_{i-1}| + |\bar{\mathbf{e}}_i|)$, $\bar{\kappa}_i$ is the curvature at vertex i in the 2D rest state. Here and henceforth, μ_* denotes stiffness parameters whose values are provided in Sec. 6. We complement the curvature energy by a term that penalizes stretching as

$$E_{\text{stretch}} = \mu_{\text{stretch}} \sum_i \frac{1}{2} \left(\frac{|\mathbf{e}_i| - |\bar{\mathbf{e}}_i| s}{|\bar{\mathbf{e}}_i| s} \right)^2 |\bar{\mathbf{e}}_i| \tag{4}$$

where s isotropically scales the curve relative to its undeformed state. The scaling parameter s is a free variable that is computed along with the curve's vertex positions during optimization. One interesting question in this context is whether, instead of using the Euclidean edge vectors $\mathbf{e}_i$, one should consider the corresponding geodesics on the limit surface. Computing these geodesics and their derivatives with respect to the curve points, however, would mean a considerable increase in computational complexity. Nevertheless, both alternatives converge to the same expressions under refinement and since we observed good behavior using Euclidean distances and angles, we opt for this simpler variant.

In summary, we can now compute the mapped shape of a planar curve on a given surface by minimizing its discrete elastic energy, subject to sufficient boundary conditions for position and orientation. Before we describe how to handle intersections and contacts between different curves, we briefly discuss the impact of Gaussian curvature.

Gaussian Curvature The underlying surface determines the normal curvature at every point on the curve, but it also affects its geodesic curvature: according to the Gauss-Bonet theorem, the integrated geodesic curvature along a closed curve is proportional to the integral of Gaussian curvature of the surface within the curve. Consequently, an embedded elastic curve will necessarily have non-zero bending energy in regions of non-zero Gaussian curvature since, unlike in its flat rest state, the sum of its exterior angles does not equal 2π. This residual energy will make closed curves drift towards developable regions where their bending energy can vanish exactly. Fortunately, we found that, for our application, small amounts of regularization can reduce this effect to a level that is unnoticeable in practice (see Sec. 3.5).

3.3 Intersections & Contact

Our focus is on networks of curves that can form contact but, by default, are not allowed to intersect. We therefore automatically prevent intersections during the design process and provide tools for editing contact relations between curves.

Intersections Similar to common practice in computer animation, our strategy is to maintain the invariant that the network is free of intersections at the end of each optimization step. Given a search direction from the Newton solver, we geometrically check for intersections between pairs of edges and reduce the size of the step until the first one is found. Once such a pair is found, we insert a uni-lateral penalty potential preventing further approach between the corresponding vertex-edge pairs. For the penalty potential, we use a truncated log-barrier function

$$E_i(x) = \begin{cases} \infty & \text{if } x \leq 0 \\ \mu_i \left(-\ln(x) - \frac{1}{2}x^2 + 2x - \frac{3}{2} \right) & \text{if } 0 < x < 1 \\ 0 & \text{otherwise} , \end{cases} \quad (5)$$

where $x = \frac{d - d_{\min}}{d_{\max} - d_{\min}}$ and $d_{\min}$ is the minimum distance between two curves, which we set to half their radius. Furthermore, $d_{\max}$ is the distance beyond which the penalty is deactivated. The distance function $d = d(\mathbf{c}^i, \mathbf{c}^j)$ itself is defined in a piece-wise manner to reflect the world-space distance between two given edges i and j, resulting in either a point-to-point, point-to-edge, or edge-to-edge distance. Although this function is only C^1-continuous with respect to the involved vertices, we did not notice any adverse effects during optimization.

Contact During network design, curves will generally form intermittent contacts. By default, these contacts are uni-lateral and will be resolved as soon as the contacting curves want to separate. In order to enforce persistent contact between a pair of curves, we provide an editing tool that allows the user to specify a pair of points, one on each curve, that should be connected. In order to implement bi-lateral contact, we use a penalty energy in the form of a zero-length spring

$$E_{\text{contact}}(t_i, t_j) = \mu_c \tfrac{1}{2} |\mathbf{c}^i(t_i) - \mathbf{c}^j(t_j)|^2 , \quad (6)$$

where $\mathbf{c}^i(t_i)$ and $\mathbf{c}^j(t_j)$ are world-space contact points, corresponding to curve parameters t_i and t_j on curves $\mathbf{c}^i$ and $\mathbf{c}^j$, respectively. Once two curves are connected, the user can still change the curve parameters in order to slide the contact points along the curves. The user can also select the desired angle α_t between the two tangents at the point of contact, which we model as

$$E_{\text{tangent}}(t_i, t_j) = \mu_t \left(\measuredangle(\mathbf{t}^i(t_i), \mathbf{t}^j(t_j)) - \alpha_t \right)^2 , \quad (7)$$

where $\mathbf{t}(t)$ are the tangents at the contact point, computed as the normalized edge vectors.

3.4 Editing Objectives

As a central component of our design system, we propose a number of tools for creating and editing curve networks. We defer a more detailed discussion to Sec. 4, but briefly describe the energies that form the basis of these tools.

In order to move curves across the surface, the user selects a given point $\mathbf{c}^i(t)$ on the curve and drags it to a target location $\mathbf{x}_t$. The curve is then pulled toward the target by means of a penalty energy in the form of

$$E_{\text{pull}} = \mu_p \tfrac{1}{2} |\mathbf{c}^i(t) - \mathbf{x}_t|^2 . \quad (8)$$

Each curve is endowed with a scaling parameter s that enters into the stretching energy (4) and is a degree of freedom in the optimization. In order to change the size of a curve to a desired value s_t, we add a penalty potential of the form

$$E_{\text{scale}} = \mu_{\text{scale}} \tfrac{1}{2} (s - s_t)^2 . \quad (9)$$

It should be noted that the bending energy (3) is scale invariant by design.

3.5 Numerical Solution

We cast curve network design as a minimization problem where the objective function is the summation of energies for shape-preservation, intersection-prevention and contact handling, as well as additional terms relating to design objectives. All terms are scaled by individual stiffness coefficients μ, whose values are given in Sec. 6.

Surface Derivatives All terms of the objective function depend explicitly on world-space positions, but the degrees of freedom of the curve live in 2D parametric space. When computing first and second derivatives required for Newton-type minimization, we therefore have to take into account the derivatives of the surface with respect to the parametric coordinates. The gradients of these energies are computed as

$$\frac{\partial E}{\partial \mathbf{p}} = \frac{\partial \mathbf{x}}{\partial \mathbf{p}}^T \frac{\partial E}{\partial \mathbf{x}} = -\mathbf{J}^T \mathbf{f} , \quad (10)$$

where $\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \mathbf{p}}$ is the surface Jacobian obtained by differentiating (1) and $\mathbf{f} \in \mathbb{R}^3$ is the penalty force obtained as the negative gradient of the energy with respect to Euclidean positions.

Newton Updates In order to minimize the objective function, we use Newton's method with line search and adaptive viscous regularization to ensure convergence in the presence of non-linearities. In each iteration, the solver provides a search direction $\Delta\mathbf{p}$ for updating the parametric positions of all curve vertices as well as their scale parameters. If the linear solver signals indefiniteness, we add a viscous regularizer of the form $E_{\text{reg}} = \frac{1}{2}\mu_{\text{reg}}^n ||\Delta\mathbf{x}(\Delta\mathbf{p})||^2$, where n is incremented until the system is positive definite. It should be noted that it is crucial to define the regularizer on Euclidean displacement $\Delta\mathbf{x}$ in order to prevent distortion artifacts due to the non-isometric nature of the intrinsic per-triangle parameterization.

Surface Coordinate Updates For each curve vertex, we use the natural piece-wise parameterization of Loop subdivision surfaces, corresponding to barycentric coordinates of a host triangle. When updating these coordinates according to a given search direction from the Newton step, curve vertices may migrate from one triangle to another. Whenever such a transition occurs, we must smoothly update both the parameter values and the search direction in order to reflect the change in parameterization. Let $\Delta\mathbf{p}_k \in \mathbb{R}^2$ denote the update direction for a given vertex $\mathbf{c}_i$ currently located in triangle $\mathcal{T}_k$ and suppose that moving $\mathbf{c}_i$ along $\Delta\mathbf{p}_k$ would make it transition into its neighboring triangle $\mathcal{T}_l$. To perform this transition, we trace $\mathbf{c}_i$ up to the boundary of $\mathcal{T}_k$ and then update its search direction by requiring that the corresponding Euclidean vectors coincide at the boundary between the two triangles, i.e., $\mathbf{J}_k\Delta\mathbf{p}_k = \mathbf{J}_l\Delta\mathbf{p}_l$. Although this system is over-determined ($\mathbf{J} \in \mathbb{R}^{3\times 2}$) the C^1-continuity of the surface ensures that, away from irregular vertices, it always has a unique solution, which we compute by applying QR-decomposition to the normal equations.

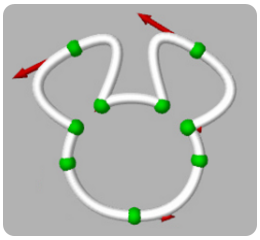
With the computational framework laid out, we can now proceed to the actual design of curve networks.

4 Curve Network Design

Our interface offers a simple work flow, aimed at making the design of ornamental curve networks simple, efficient, and enjoyable. The user starts by creating a set of planar example curves that reflect a desired aesthetic style. These curve primitives are then instantiated on the surface, where they can be moved, deformed, and connected to other curves in order to form networks. We consider two basic approaches for curve network design: a *manual mode*, in which the user, supported by our system, instantiates curves one-by-one; and a *semi-automatic mode* in which, guided by the user, curves are seeded and grown in an automated fashion. Both modes build on a set of editing operations that allow the designer to guide and adapt the resulting networks.

4.1 Curve Library

The user defines curve samples in 2D-space using a simple spline tool, illustrated in the inset figure. Example curves can be open or closed, and they can also self-intersect. To convert the spline curves into embedded elastica, we evaluate them at a given number of equally-spaced samples to obtain a set of curve points that define the rest state of the piece-wise linear elastic curve. In order to instantiate elastica, we first project a downscaled version onto the surface and then grow the curve to the desired size using the scaling energy (9). For initial projection, we simply use the normal at the location selected by the user.



4.2 Manual Layout

The manual mode implements a forward-design approach, in which curves are instantiated one-by-one. The user chooses a sample curve from the library and instantiates it on the surface: selecting a point on the surface determines the origin of the curve, dragging in a given direction changes its size and orientation. Curves can also be instantiated by selecting a point on an existing curve in order to *extrude* a new curve. The connection between the two curves is modeled using penalty energies for contact (6) and tangent orientation (7) as described in Sec. 3. This way of curve instantiation is particularly convenient when modeling, e.g., branching structures (see Fig. 1, 5).

Editing After instantiation, the user can reposition, reorient, or rescale the curves. By default, curves can move freely across the surface, but they can also be constrained by creating connections between curves, by pinning individual curve points to the surface, or by locking curves entirely. In the presence of motion constraints or contact, curves will deform in order to comply with editing objectives issued by the user; the deformed state follows as the one that minimizes the sum of elastic energy and all other energies due to contact or editing. In addition to purely aesthetic purposes, these editing operations are also used to improve the stability of the design (see Sec. 5).

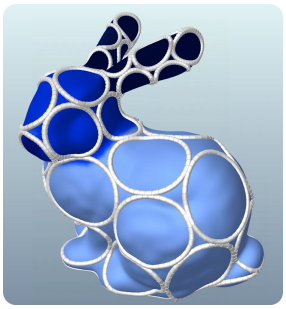
During manual design, we sometimes found it convenient to automatically lock curves in order to avoid unwanted alterations to the existing network. To allow fast and easy editing, curves will unlock immediately upon selection, and relock after editing. As the network design proceeds, curves become increasingly connected. However, since manual network design is an inherently explorative and iterative process, curves may be revisited and altered several times during the design process. Thanks to its flexible contact model, our formulation allows connected curves to be edited while preserving connectivity. In order to propagate these edits, selecting a connected curve will automatically unlock a local neighborhood of curves, allowing them to deform in order to comply with the requested edit. By default, this neighborhood includes all curves that are either directly connected to the curve being edited or within a given distance from it. However, the size can also be manually adjusted by the user.

4.3 Automated Layout

The manual design mode poses virtually no limits on creativity, allowing users to create, e.g., intricate branching networks that follow a specific aesthetic vision on both local and global scales. Other types of networks appeal *because* of their less structured or even random layouts, which require less deliberation and are thus more amenable to automation. Targeting a specific class of such layouts, we propose a semi-automatic method for creating deformable packings—quasi-random arrangements of deformable curves that evolve according to an elastic growth metaphor.

To create deformable packings, we first sample the surface according to an underlying sizing field. We mimic on-surface blue noise distributions using a dart throwing algorithm based on approximate geodesic distances. After samples are generated, we instantiate the curves at a small scale, ensuring that no overlap occurs. Curves

then *grow* by virtue of penalty energy (9) with their scale parameter determined by the underlying sizing field.



In order to create sizing fields, the users draws on the surface with the help of a brush tool. The inset figure shows an example of a sizing field, darker shade indicating smaller target scale. The brush sets the target scale $\bar{s}_i$ for the affected vertices $\mathbf{X}_i$ of the control mesh to the selected value. The target scale $\bar{s}^j$ for a given curve $\mathbf{c}^j$ is computed from the weighted average at its vertices,

$$\bar{s}^j = \frac{\sum \bar{s}(\mathbf{c}_i^j)\bar{l}_i}{\sum \bar{l}_i} , \qquad (11)$$

where $\bar{s}(\mathbf{c}_i^j)$ is determined from per-vertex sizing values using the interpolation functions of the subdivision surface. It is worth noting that the target scale for a curve changes as a function of its configuration. Curves can therefore adapt their position, size, and shape in order to better comply with the underlying sizing field.

After the curves started to grow, the user can pause the process at any time in order to guide the evolution of the network. Curves can be deleted, inserted, or edited using the tools described above. Once the user is satisfied with the network, its structural stability can be analyzed and, if necessary, improved as explained next.

5 Structural Stability

The class of artifacts that we target with our design systems are not meant to bear significant loads, but they have to be strong enough to warrant safe fabrication, shipment, and handling. Furthermore, while flexible curve networks can exhibit interesting deformation behaviors, we found that strongly localized deformations are rather disturbing. Our goal is therefore to analyze the structural stability of curve networks, identify weak spots, and guide the user in making improvements. Since loads that occur during manipulation cannot be predicted, we preclude optimization based on specific load cases. Instead, we leverage eigenanalysis to identify deformation modes that can be induced with minimal applied force.

5.1 Structural Eigenanalysis

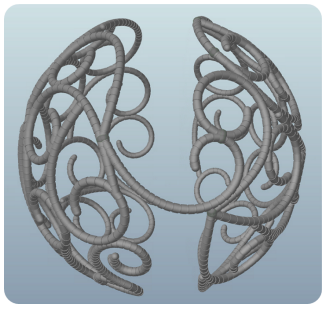
In order to predict the deformations caused by applying loads to the curve network, we use a simulation method based on discrete elastic rods [Bergou et al. 2008]. Our extension to rod networks largely follows the one described by Pérez et al. [2015], but instead of inferring connection states using best-fit transformations to incident rods, we model connections as rigid bodies with explicit degrees of freedom. In order to compute eigenmodes for a given input design, we first down-sample the curve network in a connectivity-preserving way and compute reference frames via parallel transport. We then evaluate the Hessian $\mathbf{H}$ of the elastic energy at the undeformed configuration and use Spectra [Qiu 2015] in shift-and-invert mode to compute the n eigenvectors corresponding to the smallest eigenvalues (we use $n = 20$). Note that, for a given eigenvector $\mathbf{u}^i$, we have

$$\mathbf{H}\mathbf{u}^i = \sigma_i \mathbf{u}^i = -\mathbf{f}^i , \qquad (12)$$

implying that eigendisplacements $\mathbf{u}^i$ and corresponding eigenloads $\mathbf{f}^i$ are collinear—the smaller the eigenvalue σ_i, the less force is required to create a unit displacement along the eigenvector.

5.2 Detecting Structural Weakness

In order to improve the stability of a given structure, a central question is how to detect and quantify structural weakness. Following common practice in engineering, recent work from the graphics community has considered the maximum deformation or stress induced by a given load [Stava et al. 2012; Zhou et al. 2013; Lu et al. 2014]. But while these measures can provide information on where a structure is likely to fail, these locations are not necessarily the ones where it should be improved. For example, when connecting two hemispheres with a single curve as shown in the inset figure, eigenanalysis will indicate relative rotation and twist between the two halves as the low-energy modes. While the connecting curve experiences the highest deformation for these modes, the best location for improving the structure is not at the curve, but on the opposite side of the sphere. In a broader perspective, a single missing link within a curve network can disrupt load propagation and lead to critical deformations in far away regions. But without assistance, the location of the missing link is generally not obvious. In order to directly identify locations that should be strengthened, we detect deformations in the network's dual structure—i.e., between existing curves, not within existing curves. To this end, we set up *virtual edges* between nodes from the existing network in order to measure relative displacement of curves within a given spatial radius. The underlying reasoning is that high deformations in virtual edges indicate structurally-weak spots that, once strengthened, will reduce the relative displacement in the network.



Virtual Edges In order to detect structurally-weak spots, we measure the deformation induced in the virtual edges when applying eigenloads $\mathbf{f}^i = \sigma_i \mathbf{u}^i$ corresponding to low-energy modes of the structure. Let $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ denote the positions of the two end points of a given virtual edge in the undeformed configuration. Furthermore, let $\mathbf{x}_1(t) = \bar{\mathbf{x}}_1 + t\mathbf{u}_1^i$ and $\mathbf{x}_2(t) = \bar{\mathbf{x}}_2 + t\mathbf{u}_2^i$ for $0 < t \leq 1$ be the positions obtained by displacing the original points along the corresponding components of the eigenvector $\mathbf{u}^i$, which, for small deformations, is equivalent to applying the corresponding eigenload. For a given value of t, we define the strain of the edge as

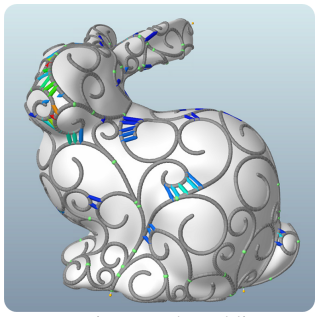
$$\varepsilon(t) = \frac{|\mathbf{x}_2(t) - \mathbf{x}_1(t)| - |\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1|}{l_{gd}} , \qquad (13)$$

where l_{gd} measures the geodesic distance between the two points in the undeformed configuration. We use geodesic instead of Euclidean distance for two reasons: first, it puts less emphasis on pairs of points that are close in Euclidean space but far apart on the surface; second, it better reflects the amount of change that would be required to connect the corresponding points. Noting that $\varepsilon(0) = 0$, we do not consider the absolute deformation for a given value of t but the rate at which strain increases when deforming the structure along the eigenvector. We therefore evaluate the rate of strain at the undeformed configuration as

$$\varepsilon'(0) = \frac{(\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1)^T(\mathbf{u}_2^i - \mathbf{u}_1^i)}{|\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1| l_{gd}} . \qquad (14)$$

It should be pointed out that, while Dumas et al. [2015] also consider the ratio of Euclidean and geodesic distance as a stability indicator, they do not take into account the deformation induced in this dual structure.

Improving Stability Once the user has converged on a candidate design, it can be checked for structural weaknesses and, if necessary, corrected. Virtual edges are set up according to a user-provided search radius and their deformations are computed simultaneously for all eigenmodes. The maximum values are then visualized by color-coding (14) relative to a user-provided interval (see inset figure). Since automatically changing the network at this stage of the design would run the risk of conflicting with the user's intent, we only provide suggestions on where to change the existing structure. The user can then make an informed decision on where and how to change the design in order to improve stability. Corrections are applied sequentially by editing existing curves (using the operations described in Sec. 4) to form new or stronger connections, or by adding new structure. The user can update the eigenmodes of the structure after each correction, which typically does not take more than a few seconds of computation.



5.3 Finishing & Fabrication

Once a design has been checked for stability and possibly corrected, we have to convert the curve network into a surface mesh for fabrication. In order to obtain smooth geometry, we first fit Catmull-Rom splines to the piece-wise linear curves and then create high-resolution triangle meshes. We fabricate the resulting meshes in PA12, a flexible thermoplastic polymer, using Selective Laser Sintering.

6 Results

In order to evaluate our design tool, we created ornamental curve networks for a variety of surface shapes and experimented with different aesthetics. Before reporting on the findings and experiences made during the design process, we start with an analysis of the performance and robustness of our method.

6.1 System Analysis

We designed a set of simulation experiments in order to assess the performance, scalability, and robustness of our method with regard to various factors. Unless indicated, we use the following values for the stiffness coefficients of the individual potentials described in Sec. 3: $\mu_i = 1$, $\mu_c = 10^7$, $\mu_p = 10^3$, $\mu_t = 100$. While the remaining parameters can be set by the user, we typically used $\mu_{\text{bend}} = 5$, $\mu_{\text{stretch}} = 10^3$, and $\mu_{\text{scale}} = 100$.

Performance In order to provide a representative sample of average run time costs for our method, we consider a simple circle packing experiment. We evenly distribute varying numbers of samples on a sphere and set the target scale of the circles such that, in the flat setting, the sum of their areas would be 1.5 times the area of the sphere. The curve network is then grown by performing 500 solver iterations, which yields visually stable results for all cases. Table 1 lists statistics obtained on a standard desktop machine with an Intel Core i7-3770 CPU and an nVidia GeForce GTX 680 graphics card. It can be seen that, when displaying the result after each solver iteration, our method provides interactive frame rates for up to around 100 curves.

When editing curves in an existing network, the performance of the system depends on the complexity of the curves and the size of the neighborhood, i.e., the number of curves that are activated during the edit. Using a similar setup with circles, each having 79 degrees of freedom, placed on a sphere, our method was fast enough to enable editing of around 20 curves at interactive rates.

Material Parameters Apart from the design objectives defined by the user, the resulting networks also depend on physical parameters, i.e., the stretching and bending stiffness of the curves, and the stiffness of the scaling energy. Due to the direct coupling between stretching and scaling, we only investigate the ratio between scaling and bending stiffness. To this end, we again run a circle packing experiment with progressively decreasing bending stiffness. In addition to intriguing structures that emerge for low bending stiffness, the results shown in Fig. 2 indicate that this ratio can be adjusted to trade-off shape preservation against packing density. It can also be noted that, even for tight packings with extreme deformations, the system remains responsive and robustly prevents intersection between the curves (we do not prevent self-intersections).

6.2 Design Examples

We performed design experiments using both the computer-assisted manual mode and the semi-automatic packing mode. In the following, we discuss some representative examples of the various curve networks that we created using our method.

Forward Network Design Aiming for a maximum degree of artistic freedom, the manual design mode of our system helps the user in creating, adapting and connecting curves, but the user is responsible for creating the layout. This mode is particularly well-suited for complex, heterogeneous networks that exhibit specific structures on both global and local scales. An example of such a network can be seen in Fig. 1 (5). Reminiscent of wrought-iron railings, this branching network of Euler spirals conveys the impression of an organically-grown structure whose carefully-placed curves delineate and emphasize different features of the underlying surface. Another example is shown in Fig. 4, where the user created a sparse surface covering with composite shapes made from individual curve primitives.

Deformable Packings Using the semi-automatic layout tool, we created deformable packings with different shapes and varying degrees of user control. Fig. 1 (1-4) shows examples of homogeneous packings that were created with minimal user input: apart from slightly smaller scaling targets for the ears of the bunny, the user only made a few changes to improve the aesthetics and the stability of the converged layouts. Another example can be seen in Fig. 3, where the designer specified two circular curves to delineate the eyes of the character, while all other curves where seeded and grown automatically. By painting sizing fields on the target surface,

# Curves	# DOFs	$t_{avg}[s]$	$t_{tot}[s]$	fps
13	1027	0.0165	22.0	22.7
21	1659	0.0247	28.6	17.5
37	2923	0.045	42.0	11.9
83	6557	0.101	76.9	6.5
304	24016	0.479	296.2	1.7

Table 1: *Statistics for circle packing test. Average time per iteration (t_{avg}), total time (t_{tot}) including rendering, and average frames per second (*fps*) for 500 iterations.*

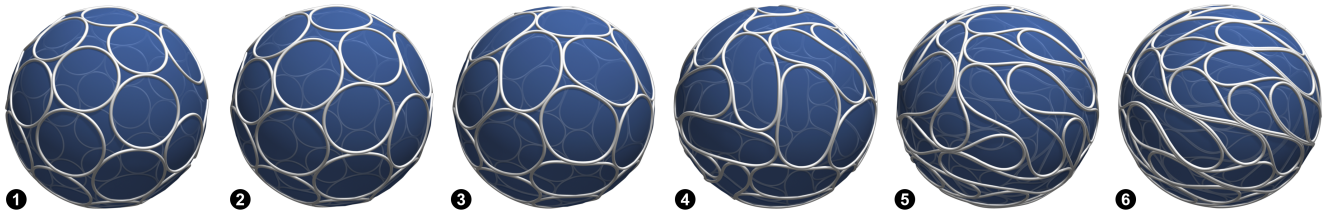


Figure 2: *Effect of bending and scaling stiffness demonstrated on a circle packing example.* (1) *Starting from a high ratio between bending and scaling stiffness that yields circles with virtually no bending deformation, we gradually decrease the bending stiffness.* (2) *circles form more contact without noticeable deformation,* (3) *circles begin to deform, densifying the packing,* (4-6) *ellipsoids progressively collapse into lobes.*

users can control the size of the curves for artistic purposes or to better reflect the underlying geometry in regions of high curvature or for small features.

Depending on the number of curves, packings can take up to a few minutes to converge to their final states. However, we found that the aesthetic impression can typically be assessed reasonably well after only a few seconds, thus enabling iterative tuning. It is worth noting that the user can intervene at any point in order to guide the evolution of the layout and to edit or fine-tune the converged networks in order to achieve specific artistic effects; see also the accompanying video. On average, creating the deformable packings that we present here took on the order of 15 minutes, including packing simulation, editing, and structural improvements.

In order to obtain further indications as to the usability of our design tool, we asked five users who had not used our tool before to create a curve network of their liking for a given input surface. The aesthetic variability among the resulting designs (see Fig. 5) confirms the consistent impression among the users that our tool provides ample artistic freedom. The design times were between 20 (Fig. 5, 1) and 40 minutes (Fig. 5, 5).

6.3 Structural Stability

We used our structural analysis tool for improving the stability of all examples that we generated. Most of the indicated weak spots could be mended by editing existing curves in order to form new or strengthen existing connections. For some cases, the user introduced new curves to add more stability to the structure. For the manually designed spiral bunny (Fig. 1, 5), the analyzer signaled a number of structurally-weak spots. While the most significant ones could be resolved without noticeable visual impact on the design, some were deliberately left unchanged in order to maintain the aesthetic intent of the designer. These decisions being governed by subjective considerations, this example illustrates the importance and advantage of a user-guided approach to structural optimization.

We performed an additional simulation experiment in order to further analyze the effectiveness of the structural improvements suggested by our system. To this end, we probe the curve network shown in Fig. 1 (5) by applying concentrated loads of 0.1N in the normal direction at 1000 different locations on the surface and compute the resulting deformed state by minimizing the (nonlinear) energy of the system. In order to quantify the improvement in stability, we measure the resulting relative displacements between near-by curves according to (13) and compare the maximum values before and after user-applied corrections. We find that the average peak and peak deformations are reduced by a factor of 1.93 (from 0.05 to 0.02) and 2.24 (from 0.42 to 0.18), respectively. These observations indicate that our method is indeed effective at reducing excessive relative deformations within the curve networks.

7 Conclusions

We presented a computational design tool for ornamental curve networks—delicate, yet structurally-sound surface representations with user-controlled aesthetics. Building on elastic curves embedded in smooth subdivision surfaces, we developed an intuitive set of user-driven design and editing tools for curve networks. Our computational system assists users in realizing their creative vision while helping to ensure that the resulting designs are structurally stable. The stability of the networks is predicted through an eigenanalysis that efficiently identifies geodesically-close curves undergoing large relative displacements under low-energy deformation modes. As evidenced by the simulated and fabricated designs created with our system, curve networks are well-suited to portraying a rich variety of aesthetic styles.

7.1 Limitations & Future Work

The energy model we employ to generate low-distortion curve embeddings relies on the normal of the input limit surface, as well as its derivatives. Unfortunately, irregular vertices introduce discontinuities in the derivatives of the surface normals. As a side effect, the convergence rates of our numerical solver are somewhat affected when curves glide over irregular vertices. In practice, we have observed that the artifacts are not significant enough to interfere with the design process. Nevertheless, as an avenue for future work, we would like to experiment with the improved subdivision scheme of Loop and Schaefer [2008], which yields surfaces that are at least G^2-continuous everywhere.

For performance reasons, we model elastica as piece-wise linear curves whose vertices live on the limit surface. The energy term that measures stretch is therefore based on Euclidean distances, while the bending energy only approximately captures geodesic curvature, as edges incident to a vertex are not perfectly tangent to the surface. While this approximate model converges to the exact one under refinement, it would be interesting to investigate ways of directly accounting for geodesic distances in a computationally-tractable way.

From a design and interaction point of view, there are several useful extensions that we plan to explore. For example, scalar fields are currently used to control the size of the curves depending on their position on the surface. It would be useful to have similar levels of control over the orientation of individual curves. Higher-level brushes could also be used to more precisely specify the distribution and packing of curves, while procedural brushes based on L-systems could add further automation to the design of intricate branching structures.

Acknowledgements

We would like to thank Eitan Grinspun for valuable discussions and insights. Furthermore, we are grateful to Alessia Marra, Loic Ciccone, Antoine Milliez, and Christian Schumacher for helping us to create and present results.

References

AKLEMAN, E., CHEN, J., XING, Q., AND GROSS, J. L. 2009. Cyclic plain-weaving on polygonal mesh surfaces with graph rotation systems. In *Proc. of ACM SIGGRAPH '09*.

BERGOU, M., WARDETZKY, M., ROBINSON, S., AUDOLY, B., AND GRINSPUN, E. 2008. Discrete elastic rods. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3.

BERGOU, M., AUDOLY, B., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2010. Discrete viscous threads. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4.

BERTAILS, F., AUDOLY, B., CANI, M.-P., QUERLEUX, B., LEROY, F., AND LÉVÊQUE, J.-L. 2006. Super-helices for predicting the dynamics of natural hair. In *Proc. of ACM SIGGRAPH '06*.

CAMPEN, M., AND KOBBELT, L. 2014. Dual strip weaving: Interactive design of quad layouts using elastica strips. *ACM Trans. Graph. 33*, 6.

CHEN, W., ZHANG, X., XIN, S., XIA, Y., LEFEBVRE, S., AND WANG, W. 2016. Synthesis of filigrees for digital fabrication. *ACM Trans. Graph. (Proc. SIGGRAPH) 35*, 4.

DUMAS, J., LU, A., LEFEBVRE, S., WU, J., AND DICK, C. 2015. By-example synthesis of structurally sound patterns. *ACM Trans. Graph. (Proc. SIGGRAPH) 34*, 4.

GARG, A., SAGEMAN-FURNAS, A. O., DENG, B., YUE, Y., GRINSPUN, E., PAULY, M., AND WARDETZKY, M. 2014. Wire mesh design. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4.

HOFER, M., AND POTTMANN, H. 2004. Energy-minimizing splines in manifolds. *ACM Trans. Graph. 23*, 3 (Aug.), 284–293.

IARUSSI, E., LI, W., AND BOUSSEAU, A. 2015. Wrapit: Computer-assisted crafting of wire wrapped jewelry. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 34*, 6.

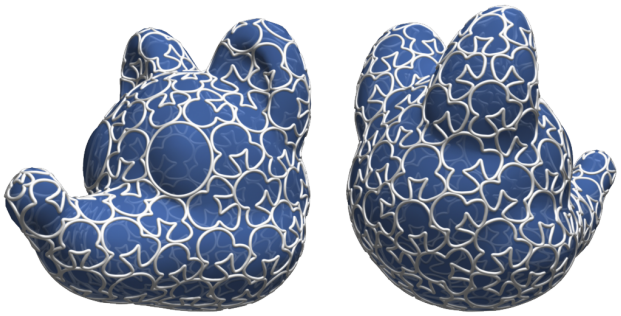


Figure 3: *For this fox model, the user specified two circular curves to delineate the eyes. Apart from a few edits for aesthetic and stability purposes, the structure was computed automatically using our deformable packing tool.*

Figure 4: *A doe decorated with a manually-designed flower covering next to its fawn, represented by an automatically-generated deformable packing of milk bottle curves.*

IGARASHI, Y., IGARASHI, T., AND MITANI, J. 2012. Beady: Interactive beadwork design and construction. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4.

KAPLAN, M., AND COHEN, E. 2003. Computer generated celtic design. In *Proceedings of the 14th Eurographics Workshop on Rendering*, EGRW '03.

KAPLAN, C. S., AND SALESIN, D. H. 2000. Escherization. In *Proc. of ACM SIGGRAPH '00*.

KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. *ACM Trans. Graph. 21*, 3.

LANGER, J., AND SINGER, D. A. 1996. Lagrangian aspects of the kirchhoff elastic rod. *SIAM Rev. 38*, 4, 605–618.

LEE, Y., AND LEE, S. 2002. Geometric snakes for triangular meshes. *Computer Graphics Forum 21*, 3, 229–238.

LOOP, C., AND SCHAEFER, S. 2008. G2 tensor product splines over extraordinary vertices. *Computer Graphics Forum 27*, 5.

LU, L., SHARF, A., ZHAO, H., WEI, Y., FAN, Q., CHEN, X., SAVOYE, Y., TU, C., COHEN-OR, D., AND CHEN, B. 2014. Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4.

MA, C., WEI, L.-Y., AND TONG, X. 2011. Discrete element textures. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4.

MARTÍNEZ, J., DUMAS, J., LEFEBVRE, S., AND WEI, L.-Y. 2015. Structure and appearance optimization for controllable shape design.

MERRELL, P., AND MANOCHA, D. 2010. Example-based curve synthesis. *Comput. Graph. 34*, 4, 304–311.

MORI, Y., AND IGARASHI, T. 2007. Plushie: An interactive design system for plush toys. *ACM Trans. Graph. (Proc. SIGGRAPH)*.

PENG, C.-H., YANG, Y.-L., AND WONKA, P. 2014. Computing layouts with deformable templates. *ACM Trans. Graph. 33*, 4.

PÉREZ, J., THOMASZEWSKI, B., COROS, S., BICKEL, B., CANABAL, J. A., SUMNER, R., AND OTADUY, M. A. 2015. Design and fabrication of flexible rod meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 34*, 4.

QIU, Y., 2015. Spectra. http://yixuan.cos.name/spectra/index.html. Accessed: 2016-01-15.

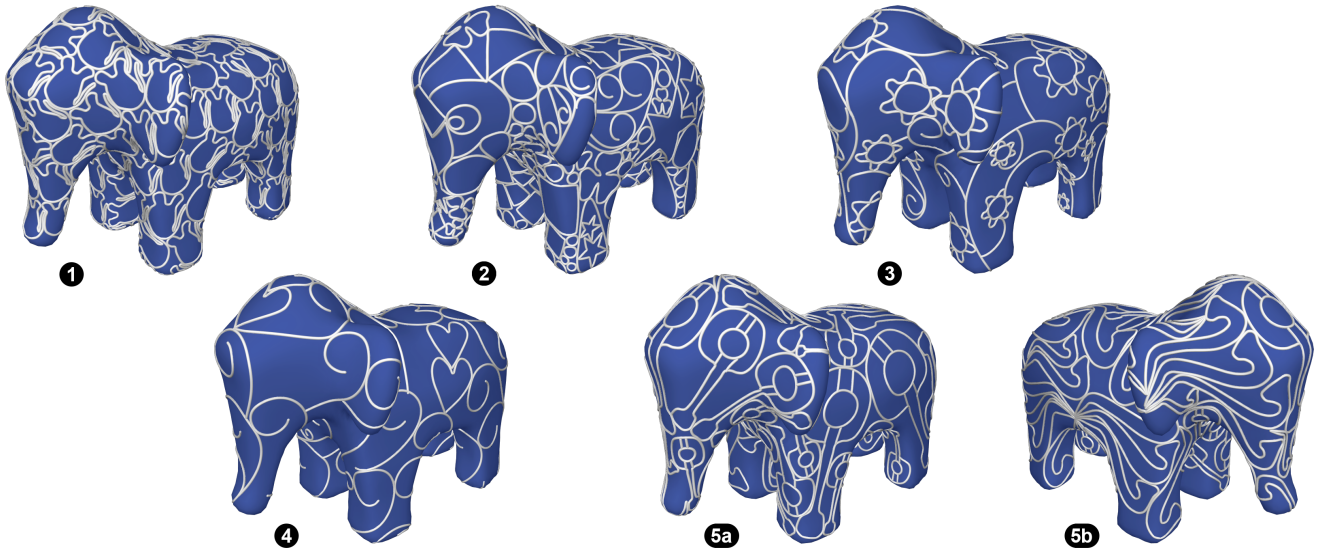


Figure 5: *Curve networks created by different users who had not used the tool before.* (1) *was created using the deformable packing tool, while* (2-5) *were created using the manual design mode.* (5a) *and* (5b) *show views of a single network with significant design variations between its two sides.*

REINERT, B., RITSCHEL, T., AND SEIDEL, H.-P. 2013. Interactive by-example design of artistic packing layouts. *ACM Trans. Graph. 32*, 6.

RUNIONS, A., FUHRER, M., LANE, B., FEDERL, P., ROLLAND-LAGAN, A.-G., AND PRUSINKIEWICZ, P. 2005. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph. 24*, 3.

SKOURAS, M., THOMASZEWSKI, B., BICKEL, B., AND GROSS, M. 2012. Computational design of rubber balloons. *Comput. Graphics Forum (Proc. Eurographics) 31*, 2.

SKOURAS, M., THOMASZEWSKI, B., KAUFMANN, P., GARG, A., BICKEL, B., GRINSPUN, E., AND GROSS, M. 2014. Designing inflatable structures. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4.

SPILLMANN, J., AND TESCHNER, M. 2009. Cosserat nets. *Visualization and Computer Graphics, IEEE Transactions on 15*, 2.

STAM, J. 1998. Evaluation of loop subdivision surfaces. In *Proc. of ACM SIGGRAPH'98 (Suppl. Material)*.

STAM, J. 2003. Flows on surfaces of arbitrary topology. In *Proc. of ACM SIGGRAPH '03*.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4.

UMETANI, N., AND SCHMIDT, R. 2013. Cross-sectional structural analysis for 3d printing optimization. In *SIGGRAPH Asia 2013 Technical Briefs*.

WALLNER, J., POTTMANN, H., AND HOFER, M. 2005. Fair curve networks in nonlinear geometries. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05.

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. In *Eurographics (STARs)*, 93–117.

XU, L., AND MOULD, D. 2009. Magnetic curves: Curvature-controlled aesthetic curves using magnetic fields. In *Proceedings of the Fifth Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics'09.

ZHOU, K., HUANG, X., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H.-Y. 2006. Mesh quilting for geometric texture synthesis. *ACM Trans. Graph. 25*, 3.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case structural analysis. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4.

ZHOU, S., JIANG, C., AND LEFEBVRE, S. 2014. Topology-constrained synthesis of vector patterns. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 33*, 6.