# Singular Foliations for Knit Graph Design

Rahul Mitra
rahulm@bu.edu
Boston University
Boston, MA, USA

Erick Jimenez Berumen
erickjb@bu.edu
Boston University
Boston, MA, USA

Megan Hofmann
m.hofmann@northeastern.edu
Northeastern University
Boston, MA, USA

Edward Chien
edchien@bu.edu
Boston University
Boston, MA, USA

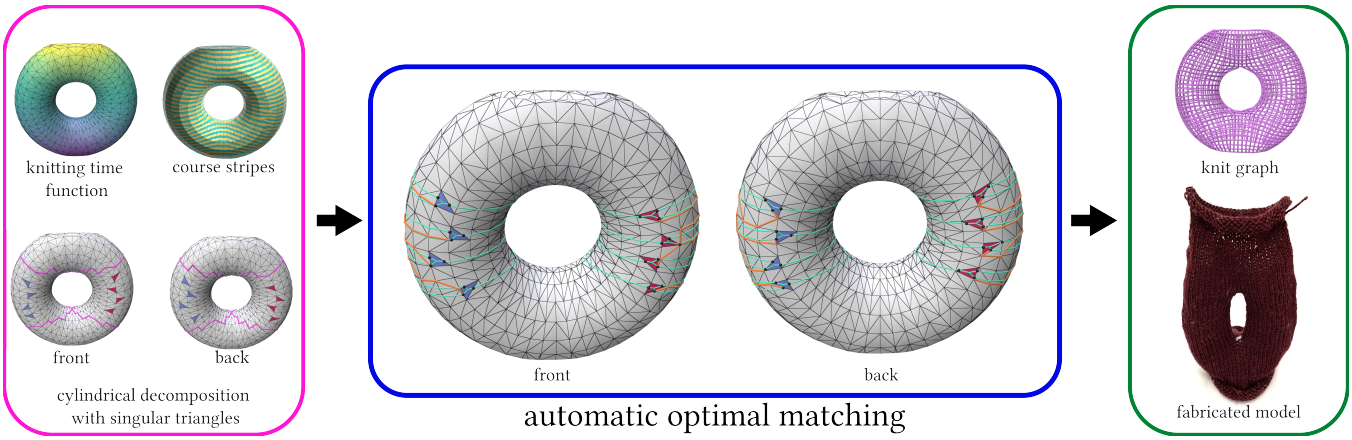



**Figure 1: Via a singular foliations perspective on the course stripe pattern, we are able to automatically match singular triangles (blue/red) and separatrices (cyan) to ensure that *all* integral curves (candidate course rows) do not helix. Our improved workflow for [Mitra et al. 2023] extends it to models with non-zero genus, decomposing the input mesh $M$ into cylindrical components along critical level sets of the knitting time function. We then solve an optimal assignment LP to obtain appropriate linear level set constraints (orange) .**

## ABSTRACT

We build upon the stripes-based knit planning framework of [Mitra et al. 2023], and view the resultant stripe pattern through the lens of *singular foliations*. This perspective views the stripes, and thus the candidate course rows or wale columns, as integral curves of a vector field specified by the spinning form of [Knöppel et al. 2015]. We show how to tightly control the topological structure of this vector field with linear level set constraints, preventing helicing of *any* integral curve. Practically speaking, this obviates the stripe placement constraints of [Mitra et al. 2023] and allows for shifting and variation of the stripe frequency without introducing additional helices. En route, we make the first explicit algebraic characterization of spinning form level set structure within singular triangles, and replace the standard interpolant with an "effective" one that improves the robustness of knit graph generation. We also extend the model of [Mitra et al. 2023] to surfaces with genus, via a Morse-based cylindrical decomposition, and implement automatic singularity pairing on the resulting components.

## CCS CONCEPTS

• **Computing methodologies** → **Shape analysis**; • **Applied computing** → **Computer-aided manufacturing**.

## KEYWORDS

computational knitting, foliations

**ACM Reference Format:**
Rahul Mitra, Erick Jimenez Berumen, Megan Hofmann, and Edward Chien. 2024. Singular Foliations for Knit Graph Design. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657487



*SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA*

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657487

## 1 INTRODUCTION

There has been much recent interest in designing algorithms for stitch structure planning in computational knitting. In the setting of

AutoKnit [Narayanan et al. 2018], these algorithms abstract stitch patterns with the notion of a knit graph, which must satisfy numerous important properties to achieve machine-knittability. The most challenging property to maintain is the *helix-free* condition: that the course rows should not form spirals. Narayanan et al. [2018] use iterative, local geodesic distance estimates to produce their knit graphs, which while helix-free, are not globally smooth and provide little to no user-control in stitch irregularity placement.

Nader et al. [2021] and Mitra et al. [2023] both leverage the more global stripe generation framework of [Knöppel et al. 2015] to generate knit graphs of an input surface $M$, although these techniques may introduce arbitrary helicing. Nader et al. [2021] use the operators presented in [Bommes et al. 2011] to address helicing, but this method is not guaranteed to remove all helices. Mitra et al. [2023] optimize for stripes directly in the space of differential 1-forms and present numerous linear constraints which may be used individually or together to generate helix-free knit graphs.

In this work, we extend and provide further insights on the 1-form-based framework of [Mitra et al. 2023]. We view the resulting *spinning form* and its stripes as a vector field flow over $M$. The integral curves make up the leaves in a *singular foliation* of $M$, decomposing it into 1-dimensional curves that serve as candidate course rows or wale columns. Singularities of the foliation correspond to vector field singularities, and represent stitch irregularities (short rows, increases/decreases).

The singular foliation splits $M$ into *cells* of equivalent flow behavior, along *separatrices* emanating from foliation singularities. We prove that matching these separatrices appropriately with linear *level set constraints* guarantees that *none* of the integral curves will form helices. Practically, this allows one to "shift" the course rows arbitrarily and change their frequency without introducing additional helices. Local variation of frequency, and thus knit density, is also used by practitioners to achieve different aesthetic/physical properties, e.g., joining of knit textures of different resolution, or knitting of ribbed cuffs. Our matching level set constraints also obviate the use of *stripe placements* in [Mitra et al. 2023], needed to prevent *specific* integral curves from helicing.

Precise topological control of foliation structure also presents an opportunity for "single-thread-level modelling". The integral curves of our foliations may be used to explicitly trace the helical yarn path in a machine knit. This constrasts with the popular knit graph of AutoKnit [Narayanan et al. 2018], where each graph node corresponds to two stacked stitches, and uses a *tracing* procedure to produce the knitting order on these stitches. As a result, this classic knit graph cannot represent certain elementary knit structures that singular foliations should be able to model.

We leverage the above understanding into an improved workflow for [Mitra et al. 2023]. This includes automation of singularity matching, extension to models with genus, and more robust knit graph generation. We list our main contributions below.

- A novel topological understanding of stripe patterns as singular foliations (§3), leading to a theoretical guarantee on no helicing from *any* integral curve (§3.2 & Prop. 1).
  - Eliminates need for stripe placement constraints of [Mitra et al. 2023].
  - Allows for shifting and frequency adjustment of stripe pattern without introduction of helicing (see Fig. 9a).
- A first algebraic characterization of the spinning form level set structure in singular triangles (§3.1.3), and a novel "effective interpolant" that improves robustness of stripe tracing for knit graph generation (§4.4.1).
- An improved pipeline for the form-based framework of [Mitra et al. 2023] (§4):
  - An automatic process for optimal singularity matching and construction of level set constraints (§4.3).
  - Extension to models of genus $g > 0$, using a Morse-theoretic cylindrical decomposition (§4.1).
- First steps toward "single-thread-level modelling" with stripe patterns following the actual (helical) yarn path of a machine knit extending the space of representable knit structures beyond that of the popular knit graph of [Narayanan et al. 2018] (§5.1).

## 2 RELATED WORK

Our work has been influenced by several knitting frameworks from the graphics and computational fabrication communities. A primer on knitting terminology is presented in Supp. §1. Narayanan et al. [2018] present an end-to-end pipeline for machine-knitting arbitrary input geometries and the works of [Jones et al. 2022; Kaspar et al. 2019, 2021; Narayanan et al. 2019] demonstrate interfaces for varying stitch layout, coloring and texturing. Albaugh et al. [2023]; Hofmann et al. [2019, 2023, 2020] construct tools and a domain-specific language for generating machine knitting instructions, with special care taken for handling knit textures. Other works [Wu et al. 2018; Yuksel et al. 2012] generate *stitch-meshes* for rendering purposes while the works of [Igarashi et al. 2008; Wu et al. 2019] are aimed at producing hand-knittable output. Lastly, the works of [Mitra et al. 2023] and [Nader et al. 2021] leverage stripe pattern tracing to produce knit graphs that are machine-knittable.

### 2.1 Stripe Patterns for Fabrication

Methods of stripe generation on surfaces have received much recent attention in the realm of digital fabrication. The ability to specify evenly-spaced stripes with directional guidance has found several modeling uses. Nader et al. [2021] were the first to use stripe patterns for knit graph generation, tracing the stripes of [Knöppel et al. 2015]. Noma et al. [2022] presented a spinning-form-based framework for editing and connecting stripe singularities in such patterns, applying their tools to 3D wireframe structures and "zippables" [Schüller et al. 2018]. Mitra et al. [2023] extend and apply a similar form-based framework, developing novel constraints for eliminating the helicing inherent in stripe patterns generated with [Knöppel et al. 2015]. The work of [Tricard et al. 2020] generates 2D stripe patterns with phasor noise methods and extrude them to obtain microstructures with tailored deformation properties. Montes Maestre et al. [2023] develop a differentiable version of the [Knöppel et al. 2015] pipeline and use it for inverse design of stripe-shaped bi-material distributions. Lastly, [Jourdan et al. 2023] utilize [Knöppel et al. 2015] stripe patterns to generate layer toolpaths for 3D printing of self-shaping shells. Like many works above, we use the spinning form interpolant of [Knöppel et al. 2015], but we are

the first to explicitly characterize the level set structure in singular triangles (§3.1.3).

With regard to [Mitra et al. 2023]: we improve their pipeline by preventing helicing of *any* level set in the stripe pattern, as opposed to only specific level sets. This allows for shifting and frequency change of the stripe pattern without introducing additional helices, removing their need for stripe placement constraints. We also implement an automatic optimal-assignment-based strategy for matching singular triangles (§4.3), and appropriately extend their framework to models with genus (§4.1). Lastly, we achieve more robust stripe tracing for knit graph generation via our "effective interpolant" (§4.4.1).

## 2.2 Foliations in Geometry Processing

There have been several works in graphics and geometry processing that have used and referenced foliations in various applied contexts.

Vekhter et al. [2019] produce approximately geodesic foliations for fabricating triaxial weaves of input mesh surfaces. In our setting, we do not require our foliations to be approximately geodesic, but we do require more precise global topological control on the foliation structure due to the manufacturing constraints of knitting.

Campen et al. [2016b] produce bijective maps between simply-connected 2D and 3D domains via globally trivial simplicial foliations, specified via face- or tet-wise constant vector fields. Our setting requires foliations with singularities to produce geometric shaping and is formulated on surface domains with more complex topology (at least two boundaries, and potentially some genus).

Foliations are implicit in many works on vector field design or tracing, as any vector field (in the smooth setting) gives rise to a foliation via its integral curves. We highlight specific works that are particularly relevant, and refer the reader to [Vaxman et al. 2017] for a broader survey. Explicit care is taken in [Zhang et al. 2006], where they describe the dynamics of the vector field flow, but do not attempt to specify global control over the *orbit complex* (see §3.2), instead focusing on the ability to combine and move singularities. Two relevant field tracing works are [Bhatia et al. 2011; Ray and Sokolov 2014] which aim to provide vector field representations and robust tracing algorithms that guarantee a well-defined global vector field flow. Our analysis in §3.1.3 provides just such a tracing for the spinning form considered as a vector field representation.

A related line of works are quad- and hex-meshing works, with these meshes arising from integral curves of orthogonal vector fields (perhaps on a branched cover). Again, we cite particularly relevant works and refer the reader to surveys [Bommes et al. 2013; Pietroni et al. 2022] for more information. One work that explicitly uses the language of foliations is [Liu and Bommes 2023]. They consider the matter of locally hexable volumetric frame fields, and use the singularity theory of foliations to characterize hexability. Within the quad meshing setting, there are also several works that aim to control the analogous helicing behavior [Bommes et al. 2011; Campen et al. 2016a]. The latter is especially relevant, with the use of "cycle" constraints (akin to our level set constraints) on the parameterization (§4.3) used. The knit graph generated by our method may also be viewed as the edges of a quad-dominant mesh, with singular triangles arising as *position field singularities*, as described in [Jakob et al. 2015]. Unlike this more general setting, our manufacturing constraints do not allow *orientation singularities* and require more precise topological control than their framework affords. Finally, we note a recent work [Mitropoulou et al. 2024] that considers a subset of quad meshes, *strip-decomposable*, that are aimed at adjacent fabrication tasks. While they provide manual topological editing tools for controlling separatrix behavior, our singularity matching is automatic. Furthermore, their setting is also different, with pure quad meshes and only orientation singularities.

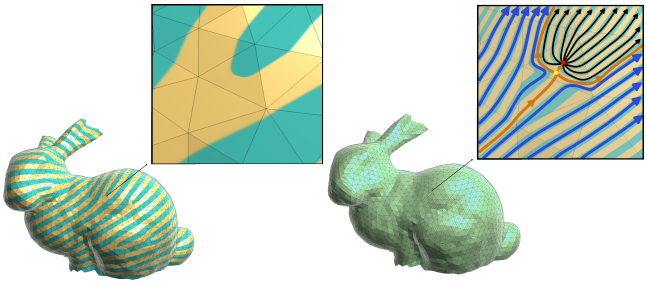


**Figure 2: Stripe patterns as oriented foliations. *Oriented foliations* are a collection of integral curves of a vector field. A contiguous set of integral curves form the stripes in a stripe pattern specified via spinning form. We visualize these integral curves on the right by increasing the stripe frequency and highlighting specific curves. The surface is partitioned into 0- and 1-dimensional integral curves: *leaves* and *singularities* (flow fixed points), respectively. The red point denotes a *source*, the yellow point denotes a *saddle*. The curves are colored black, orange, and blue to denote curves born at the source, *separatrices*, and curves born elsewhere respectively.**

# 3 FOLIATIONS LENS

In this section, we set down a basis for structural understanding of stripe patterns as *oriented foliations*[1], or equivalently as the collection of integral curves of a vector field. Informally, a foliation of a surface $M$ is a partition of the surface into 1-dimensional curves, called *leaves*, and 0-dimensional points, called *singularities*. Away from singularities, the partition looks locally like the partition of the $xy$-plane into constant $x$ or $y$ lines. As can be seen in Fig. 2, the stripes in a pattern are sets of "contiguous" leaves, starting and ending at singularities. We will see that control over the foliation topology translates into the strict helix-free manufacturing constraints required of machine knitting.

Throughout the exposition below, we will refer to [Aranson et al. 1996] as a detailed source on surface foliations. The manuscript discusses many foliations more exotic than those arising from stripe patterns, so we summarize and extract the sections relevant to our setting. Implicit above is a key fact from the study of foliations: Whitney's Theorem (Theorem 2.3 in [Aranson et al. 1996]), which states that every orientable foliation is induced by the *flow* of a vector field. That is to say, the leaves and singularities are formed

[1]Unoriented foliations contain singularities of half-integer index, like those specified via line field (rather than vector field) in [Knöppel et al. 2015]. These singularities are undesirable as they result in stitch structures that are not machine-knittable.

by the integral curves of a vector field $V$. A curve $\gamma : I \to S$ is *integral* if $\gamma'(t) = V(\gamma(t))$ for all times $t$.

## 3.1 Spinning-form-based stripes

We use the setting of form-based stripe patterns, first introduced in [Knöppel et al. 2015], and built upon for knitting applications in [Mitra et al. 2023]. We provide a review of the framework, and present a novel analysis of the stripe patterns on singular triangles from a foliations perspective. For background on discrete differential forms and exterior derivatives we refer readers to the works above and [Crane et al. 2013] for an in-depth introduction.

The method of [Knöppel et al. 2015] is based on a smooth setting where a complex function $\psi : M \to \mathbb{C}$ over a surface $M$ is considered, with $\arg\psi$ used as a stripe texturing coordinate. The input to their method is a vector field $Z$ that is used to roughly specify $\nabla(\arg\psi)$, designating the direction and frequency of pattern variation. Intuitively, stripes are "born" or "die" at zeros of $\psi$, where $\arg\psi$ has no continuous local definition.

In their discretization, $Z$ and the resulting optimized $\psi$ are specified per-vertex, and a discrete *spinning form* is used to interpolate $\arg\psi$ into the interior of mesh triangles. The spinning form is a discrete 1-form, specified by a scalar per mesh edge: $\sigma : E \to \mathbb{R}$, where $\sigma_e = \sigma_{ij}$ represents the change in $\arg\psi$ over edge $e_{ij}$. $\sigma_e$ is optimized to best match the desired frequency specified by $Z$, over the mesh as a whole. The result is a 1-form for which $(d_1\sigma)_m = 2\pi K_m,\ K \in \mathbb{Z}^{|\mathbb{F}|}$ on each face $m$ of the mesh. In general, $2\pi$ may be replaced by a user-specified period $P$, resulting in stripes of width $P/2$.

On most triangles within the mesh, one should have $K_m = 0$. This denotes zero curl of $\nabla(\arg\psi)$ over face $m$, leading to a well-defined $\arg\psi$, and a simple linear stripe pattern (Fig. 3, left). On other triangles, $K_m \neq 0$, indicating non-zero curl, and these correspond to zeros of $\psi$. These are termed *singular triangles* and $K_m$ is the *singular index* of face $m$. On such triangles, $K_m$ stripes are created (Fig. 3, right), and the stripe pattern is governed by a discontinuous *texture function interpolant* $\varphi$, presented in Supp. §2 and analyzed in §3.1.3.

The work of [Mitra et al. 2023] optimized for such a spinning form directly, in a framework that allowed for the use of linear *level set* and *stripe placement* constraints to prevent helicing of *certain* level sets. In particular, stripe placement constraints ensured that stripes were centered on non-helicing level sets (see Fig. 6 of [Mitra et al. 2023]). In their framework, a harmonic knitting time function $h$ (e.g., Fig. 1, top left) and its gradient $\nabla h$ roughly specify the course row alignment and direction of knitting via an optimization objective (Eq. (2a) and §3.1 of [Mitra et al. 2023]). Several linear structural constraints were applied to achieve singularity placement and stripe alignment amongst other aims (§3.2 of [Mitra et al. 2023]). Most relevant here are the level set constraints, which set the path integral $\int_\gamma \sigma = 0$, where $\gamma$ is a path of mesh halfedges (e.g., visualized in Fig. 1 in orange). These ensure that no level set (or stripe) crosses $\gamma$ without crossing back, and are used to match separatrices starting or ending at singular triangles.

*3.1.1 Forms as discretized vector fields.* One-forms are common "edge" discretizations of continuous vector fields $W$ on triangle meshes $M$ [de Goes et al. 2016]. Typically, one takes the field $W$ and sets the 1-form value on edges to be equal to the path integral $\sigma_e = \int_e W$. This perspective results in simple discretized differential operators like the discrete curl given by the exterior derivative $d_1$. To go in the other direction, from a discretized 1-form to a vector field $W$ on the entire mesh $M$, one makes a choice of field interpolant, like the common Whitney basis [Whitney 1957].

In our setting, we may consider the texture interpolant $\varphi$, (Eq. 3, supp.) as a particular choice of field interpolant, via its gradient $W = \nabla\varphi$. The level sets of $\varphi$ are then perpendicular to $W$ and are traced out by the integral curves of $V = W^\perp$. This is illustrated in Fig. 3. Ultimately, we characterize the flow and integral curves of $V$ via an algebraic analysis of the level sets of $\varphi$.

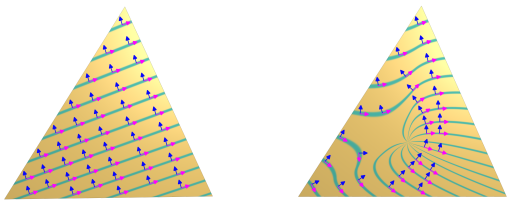


**Figure 3: Non-singular triangle left, singular triangle right, with integral curves (cyan) of $V = W^\perp$ (pink) and $W = \nabla\varphi$ (blue) illustrated. The convention to rotate $W$ clockwise is adopted so that a positive/negative singular index implies that the barycenter is a *source/sink* fixed point.**

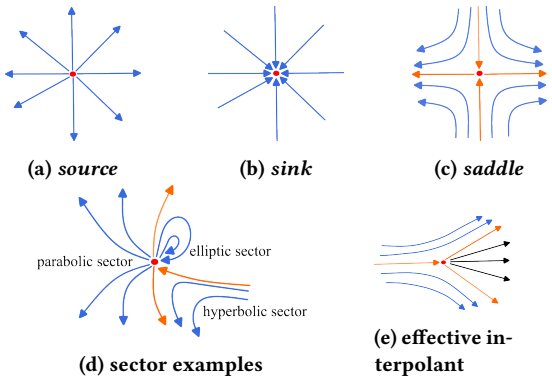



**Figure 4: Flow structure near singularities: (a) a *source*, (b) a *sink*, (c) a *saddle*, (d) sector examples (inspired by Fig. 1.44 from [Aranson et al. 1996]), (e) a schematic representation of our "effective interpolant" with two hyperbolic sectors and one parabolic sector (see §4.4.1).**

*3.1.2 Basic terminology for 2-dimensional flows near singularities.* We recall some basic definitions for 2-dimensional flows near *singularities*. Further information may be found in [Aranson et al. 1996], or in [Günther and Baeza Rojo 2020]. Singularities are fixed points of the flow: points $p$ for which $V(p) = 0$. The integral curve containing $p$ is simply stationary: $\gamma(t) = p$ for all times $t$.

In our setting[2], at a given singularity $p$, the flow is split locally into *sectors* bounded by *separatrices*. Separatrices are integral curves that approach or leave $p$ as $t \to \infty$ or $t \to -\infty$, and that have nearby integral curves not converging to $p$ (see §2.3.3 of [Aranson et al. 1996] for an epsilon-delta definition). These sectors may be of three types: *hyperbolic*, *elliptic*, or *parabolic*, and are illustrated in Fig. 4d. Arising from $\varphi$, there are three kinds of singularities we encounter: *sinks*, *sources*, and *saddles*; all illustrated in Fig. 3.1.1.

*3.1.3 Flow structure in singular triangles.* We now present an algebraic characterization of the integral curves of $V$, via an analysis of the level sets of $\varphi$, the texture interpolant of [Knöppel et al. 2015]. In non-singular triangles the level sets are linear and easily inferred, but on singular triangles the level sets are quadratic in barycentric coordinates, and more challenging to detail.

For the sake of brevity, we present just the case of $nP > 0$, illustrated in Fig. 5. When $nP < 0$, the characterization merely differs by a few sign changes. The expression for $\varphi$ and full arguments for this characterization are deferred to Supp. §2, as we are mostly interested in the topological structure of the flow.

Subcases are formed by considering the set of signs of $\sigma_{ij}, \sigma_{jk}, \sigma_{ki}$. As their sum is positive, at least one must be positive, and we denote the cases $+++, ++-, +--$. Note that ordering does not matter, merely how many positive and negative values there are.

- $+++$: A single source singularity is present at the barycenter, with all integral curves exiting at the boundary (Fig. 5, left).
- $++-$: There is a source at the barycenter, and a saddle in the barycentric region bounded by the negative sign edge (Fig. 5, middle). A separatrix from the saddle exits the source, and the rest of the integral curves exiting the source start on the edges with positive $\sigma$.
- $+--$: There is a source at the barycenter, and a saddle in the barycentric region bounded by the more negative edge (Fig. 5, right). A separatrix from the saddle exits the source, and the rest of the integral curves exiting the source start on the single positive edge.

Our optimization overwhelmingly produces the $++-$ case, as the spinning form optimization of [Mitra et al. 2023] motivates the $\sigma_{ij}$ to agree with $\nabla h \cdot e_{ij}$.

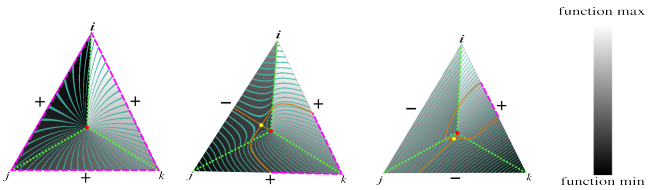
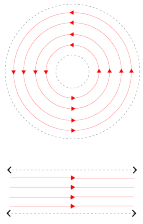



**Figure 5: Behavior of integral curves in singular triangles when $nP > 0$. Signs of $\sigma_{ij}, \sigma_{jk}, \sigma_{ki}$ are indicated with ±'s. Purple indicates the *birth interval* where integral curves exiting the source leave the triangle. Color-coded are separatrices (orange), sources (red), and saddles (yellow).**

For precise control of global foliation structure, we obtain exact expressions for the separatrix level sets in both the $++-$ and $+--$ cases. Two of these level sets bound the *birth/death interval* of integral curves that tend to the barycenter as $t \to -\infty/+\infty$ when $nP > 0$ / $nP < 0$, respectively. In Supp. §2 we derive these expressions and describe how to obtain their intersection with the triangle boundary.

## 3.2 Global structure of flows and foliations

The flows induced by spinning forms are of a well-behaved class: flows *without nontrivial recurrent trajectories* (see §2.2.1 of [Aranson et al. 1996]). In particular, the only trajectories that are part of their limit sets are singularities (fixed points) and closed cycles ($\{\gamma(t)\}_{t\in\mathbb{R}}$ forms a closed loop). A point $p$ is in the *limit set* of a trajectory $\gamma$ if there is a sequence $\{t_i\}_{i=1}^{\infty}$ with $t_i \to \infty$ such that $\gamma(t_i) \to p$.



For this set of flows, the global topological structure is captured by an object called the *orbit complex*. Roughly speaking (see §6.5 of [Aranson et al. 1996] for detail), this describes a decomposition of $M$ into *cells* that are open disks or annuli topologically. These cells are the complement of the singularities and the separatrices. Within these cells, the flow is like that of a parallel flow on an open strip or a parallel flow on an open annulus (illustrated in the inset).

In Fig. 6, one can see a simple example of this cell structure on a cylinder. When no level set constraint is used (left), there are two disc-shaped cells with different flow behaviors with respect to the candidate wale columns (purple): cell A helices, while cell B does not. If we join separatrices with a level set constraint (right, orange), birth/death intervals line up and only one non-helicing disc-shaped cell results. The practical implication of this behavior is that if one were to sample the integral curves at a higher rate (by increasing stripe frequency) or shift the integral curves traced, then no new helices may be introduced. An example of frequency doubling with no new helices is demonstrated with the sock model in §5.

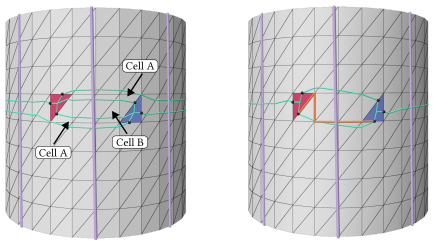



**Figure 6: Left: With no separatrix (cyan) matching, two cells with different behavior arise: those in cell A helix, while those in cell B do not. Right: By matching with a level set constraint (orange), we ensure a single disc-shaped cell, with all integral curves not helicing with respect to the central candidate wale column (purple).**

In Supp. §3, we prove the following proposition, which shows that control of the boundary separatrices via level set constraints can be used to guarantee that none of the curves in a cell helix with respect to a *transverse* wale foliation. In short, *if the level set constraint does not helix, then the integral curves in the neighboring cell won't either.*

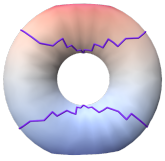
[2]Arising from spinning forms, $V$ has *isolated* singularities (as there are only a finite number). Furthermore, these singularities are not surrounded by closed cycles.

Proposition 1. *Consider two transverse course and wale foliations specified by $\sigma_c, \sigma_w$, and our effective interpolant. In any disc-shaped cell $R$ in the orbit complex of $\sigma_c$, if either boundary of $R$ is helix-free with respect to $\sigma_w$, then all integral curves of $R$ are helix-free.*

## 4 METHOD

In this section, we outline our procedure for generating a stripe pattern that avoids helicing over all leaves. First, we semi-automatically generate singular triangle positions and spinning form values on said triangles, informed by the geometry of the model. Next, we automatically match singular triangles with level set constraints that align their birth/death intervals and ensure no helicing. Lastly, we use our effective interpolant to more robustly trace and produce a knit graph suitable for machine knitting.

### 4.1 Models with non-zero genus



We first describe the additional structures needed to handle more complex topologies for $M$ (those with $> 2$ boundaries and/or nonzero genus). The first is a decomposition of $M$ into cylinders along edge cycles that roughly follow critical level sets (in the Morse sense) of the guiding time function $h$, as illustrated in the inset. These critical level sets are the values of $h$ at saddle vertices of $M$. In §4.2, we check that each cylindrical component has singular indices that sum to 0, and in §4.3, we only match singular triangles within these cylindrical components. Details on our method for obtaining the necessary edge cycles are in Supp. §4.

Secondly, in genus $g > 0$ cases, a spinning form $\sigma$ must satisfy integer integrability constraints along homology generators to result in a well-defined function $M \to \mathbb{S}^1$. If these do not hold, the relative value between two vertices would be path-dependent. To impose these constraints, the well-known tree-cotree algorithm [Dłotko 2012] is applied to find a set of $2g + n - 1$ edge cycle generators. These are gathered into a matrix $\mathbf{H} \in \{+1, -1\}^{(2g+n-1)\times|E|}$ that performs path integration along the generators, and $\mathbf{k}^{\text{hg}}$ gathers the corresponding integer variables in Eq. (2d).

### 4.2 Singular triangle generation

To generate singular triangle positions, indices, and spinning form values on these triangles, we use [Knöppel et al. 2015] to make an initial guess. The method generates geometrically-informed singular triangle positions and indices, and the resulting form values on these triangles is roughly aligned with the gradient.

Oftentimes, this method alone is suitable for determining reasonable singular triangle positions. They may then be paired with the procedure of §4.3 to fix helicing. However, there are two instances that may trigger manual fixing. First, the singular indices on each cylindrical component may not sum to 0. This constraint ensures that short rows do not wander from component to component and encourages satisfaction of "simple splits and merges" (Property 5 of [Narayanan et al. 2018]). The second related instance is when pairs of matched singular triangles differ greatly in terms of their harmonic time function $h$ values. This leads to stripes that are quite "slanted" with respect to the level sets of $h$ as can be seen in the left of Fig. 7. The optimization of [Knöppel et al. 2015] does not avoid this, due to the lack of constraints preventing helicing. The manual fix in both instances is placement of additional singular triangles on approximately the same isoline of $h$. We leave to future work the development of a fully automatic method that solves this problem.

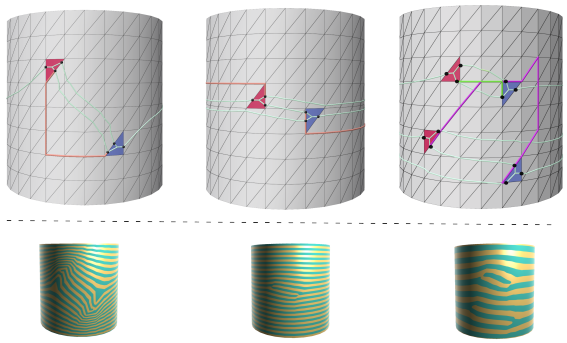


**Figure 7: Left: Paired singularities highly offset with respect to $h$ leads to a highly slanted stripe pattern with much width variation. Middle: The level set constraint (orange) opposes $V$ when going from index +1 (red) to index -1 (blue). This leads to helicing of the integral curves in the resulting cell. Right: Two level set constraints (purple and green) intersect, leading to helicing integral curves in the bottom cell.**

### 4.3 Birth/death interval matching

In our next step, we automatically match singular triangles, and align their birth/death intervals with suitable level set constraints. For simplicity, we assume that singular indices are all $\pm 1$, and thus by the global index theorem (Thm. 3.1 of [Noma et al. 2022]), there should be an equal number $s$ of $+1$ and $-1$ singularities to match. This aligns with the output of [Knöppel et al. 2015] in nearly all cases, as spacing dislocations in the stripe pattern lowers the objective energy.

For the singularity matching and level set constraints, three desiderata inform our automatic pairing strategy (motivating examples illustrated in Fig. 7):

(1) The value of the time function $h$ on paired singularities should not differ too much, and level set constraints should roughly follow the isolines of $h$.
(2) Level set constraints should roughly follow the direction of $V$ when going from index $+1$ to index $-1$ singular triangles.
(3) Level set constraints should not cross each other, as this causes separatrices to wander wildly to avoid crossing.

Thus, we solve for the matching as an optimal assignment problem, calculating a cost matrix $\mathbf{C} \in (\mathbb{R}^+)^{s\times s}$ for matching each $+1$ singularity with each $-1$ singularity. The cost is obtained via a Djikstra's shortest path search on a custom-weighted graph of mesh halfedges. The weight for a halfedge $e_{ij}$ is given by the absolute difference between $\vec{e}_{ij} \cdot (V/\|V\|)$ and the maximal such dot product over all mesh halfedges. The dot product measures how well each halfedge aligns with $V$, and encourages our shortest path algorithm to find paths satisfying (1) and (2) above. For $+1$ singularity $i$ and $-1$ singularity $j$, the length of the shortest path is stored in entry $\mathbf{C}_{ij}$.

The optimal assignment will be stored in a permutation matrix $\mathbf{T} \in \{0,1\}^{s\times s}$, where $\mathbf{T}_{ij} = 1$ denotes a matching of +1 singularity $i$ with −1 singularity $j$. We solve the following LP relaxation which has a binary minimizer:

$$\min_{\mathbf{T}\in[0,1]^{s\times s}} \quad \langle \mathbf{C}, \mathbf{T}\rangle \tag{1a}$$

$$\text{such that} \quad \mathbf{T}\mathbb{1} = \mathbb{1} \tag{1b}$$

$$\mathbb{1}^T\mathbf{T} = \mathbb{1}^T \tag{1c}$$

Eq. (1a) denotes a sum of the entrywise products, and Eqs. (1b), (1c) ensure that any binary minimizer $\mathbf{T}$ is a permutation matrix.

Finally, with the matching in hand, we need to find the exact path of the level set constraints aligning birth/death intervals. We may not use the paths used to calculate $\mathbf{C}$ as these may violate (3) above. We recall here as well that the choice of path is only relevant up to homological class in the mesh $M$ minus the singular triangles (see Lemma 1 of [Mitra et al. 2023]), so deviating from the "best paths" used to calculate $\mathbf{C}$ is not damaging.

To find these, we simply take each matched pair in turn and calculate the best path available currently. After each exact path is calculated, the halfedges on the path have their weight set to $+\infty$ to prevent subsequent paths from crossing. Lastly, we utilize the separatrix intersections described in Supp. §2.3 to complete the level set constraint. Our matching *automates* the manual matching of [Mitra et al. 2023], and produces good pairings reliably.

## 4.4 Form optimization and knit graph generation

With constraints gathered, we solve the following optimization problem for the course 1-form $\sigma_c$. We provide a brief review of our notation here, borrowed from [Mitra et al. 2023], and refer the reader there for a more in-depth description, if desired. The gradient of the time function, $\nabla h$ should be approximately orthogonal to the course rows and parallel to the wale columns. Thus, we define our comparison 1-form $\omega_c$ as the average of the normalized adjacent face gradients for each edge [Mitra et al. 2023]. $d_1$, $P$, $W$, and $\mathbf{H}$ are the discrete exterior derivative operator, stripe period, diagonal cotangent weight matrix, and set of homology generators, respectively. $\gamma_j^{ls}$ is used to denote a particular level set constraint, where the total number of such constraints are given by $N^{ls}$. Finally, $\mathbf{k}, \mathbf{k}^{hg}$ are integer variables that represent singularity index at a face and the number of striping level sets modulo $P$ that pass through a homology generator, respectively.

$$\min_{\sigma_c, \mathbf{k}^{\text{hg}}} \quad ||W(\sigma_c - \omega_c)||^2 \tag{2a}$$

$$\text{subject to} \quad \sigma_c\big|_{\partial M} = 0, \tag{2b}$$

$$d_1\sigma_c = P\mathbf{k} \tag{2c}$$

$$\mathbf{H}\sigma_c = P\mathbf{k}^{\text{hg}} \tag{2d}$$

$$\int_{\gamma_j^{\text{ls}}} \sigma_c = 0, \qquad 1 \le j \le N^{\text{ls}} \tag{2e}$$

This problem only has $2g+n-1$ integer variables, and is significantly quicker to run than the mixed-integer problems considered in S2 of [Mitra et al. 2023] ($O(|F|)$ integer variables).

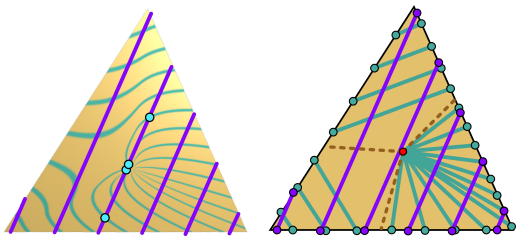


**Figure 8: Left: The spinning form interpolant $\varphi$ produces course curves (cyan) that intersect with a single wale column (purple) multiple times if they pass close enough to the barycenter. Right: Our effective interpolant ensures that this problem does not occur, only modifying the course foliation locally.**

For the wale $\sigma_w$, we can usually simply call [Knöppel et al. 2015]. In the rare instances when we have collisions of wale and course singular triangles, we are forced to run a larger mixed-integer problem, with integer variables $\mathbf{k} \in \mathbb{Z}^{|F|}$. This problem drops the level set constraints, Eq. (2e), as in [Mitra et al. 2023] (see last paragraph of §4.1 in that work). Finally, we note that it is possible to include the stripe alignment constraints of [Mitra et al. 2023], but care would need to be taken to avoid infeasibility, so we do not include them here.

*4.4.1 Effective interpolant.* Our final step is to generate the knit graph by tracing the integral curves of $V$. However, as our analysis in the singular triangles shows, this is likely to cause failure in certain cases (Fig. 8). Thus, we develop an *effective interpolant*, that achieves the spinning form values on the edges of a singular triangle (also shown in Fig. 8) and does not suffer these robustness issues.

Schematically, we replace the interior integral curve structure with a single source/sink singularity at the barycenter with two hyperbolic sectors and a parabolic sector. Our procedure for tracing this effective interpolant is described in Supp. §5.

# 5 RESULTS

We apply our improved workflow and automatic separatrix matching to numerous models illustrated throughout the text and in Fig. 9. We highlight the "torus" and "holey pants" as models with genus, and the two versions of the "sock" as showing frequency doubling without introduction of additional helices. With regard to manual intervention for singularity placement: a pair of singularities is induced for explanatory purposes in the cylinder model (Fig. 9d); the automatic singularity positions from [Knöppel et al. 2015] are used directly on the sock model (Fig. 9a); for the remaining models, initial singularity positions from [Knöppel et al. 2015] are modified minimally to ensure appropriate time function alignment. Edge length error histograms for our knit graphs are in Fig. 10.

After knit graph generation, we use Autoknit [Narayanan et al. 2018] for machine scheduling. All samples were knitted on a Shima Seiki SWG91N2 15-gauge v-bed knitting machine using 2/28NM rayon yarn at a 35-stitch size at a rate of 0.8 m/s.

To solve the optimization problems of §4.4, we use the vanilla Gurobi solver [Gurobi Optimization, LLC 2022] on a 2.3GHz Intel

Core i5 Macbook Pro with 8GB of RAM. As with S1 of [Mitra et al. 2023], with few integer variables, our runtimes are interactive and under a second. The applications of [Knöppel et al. 2015] are even faster, and follow the behavior noted in their paper.

## 5.1 Single-thread-level modelling

In Fig. 11, we show that our representation provides the opportunity for "single-thread-level modelling" of knit patterns and is capable of representing knit patterns that are not accessible to the widely-used knit graph representation of AutoKnit [Narayanan et al. 2018]. On the left, a tracing of the bent cylinder model is implied by the foliation structure. On the right, a similar tracing is implied, and the short row wraps around the cylinder model multiple times, something not possible with the knit graph representation of [Narayanan et al. 2018]. These are generated with our novel topological understanding and separatrix matching level set constraints, which allow us to precisely specify the "amount" of helicing. We are excited at the prospect of a "tracing-free" pipeline for generating machine instructions, but leave this to future work.

# 6 CONCLUSION

In this work, we outline and demonstrate the benefits of a greater topological understanding of stripe patterns. We view them as singular foliations, and match separatrices appropriately to effect precise control on integral curves, and thus candidate course rows and wale columns. This prevents helicing in a more robust fashion, and presents an opportunity for development of a form-based tracing-free pipeline for generation of machine knitting instructions. We also improve greatly on the workflow of [Mitra et al. 2023] as outlined in §1, and demonstrate it on a selection of models in §5.

## 6.1 Limitations and Future Work

Our workflow still sometimes requires manual intervention in the generation of singular triangle positions. Going forward, we hope to develop a method that will automatically generate pairs of singularities along isolines of $h$. Alternately, the development of a tracing-free pipeline would allow for matched pairs to differ greatly in $h$ values (Fig. 11, right). The development of such a pipeline for generating machine knit instructions is interesting for independent reasons. Foremost among these is that it allows for a vastly expanded range of representable knit structures.

# ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation (NSF) under Grant No. 2327137. We would like to thank James McCann for assistance with the AutoKnit scheduler, and Samuel Silverman for generation of the "holey pants" model.

# REFERENCES

Lea Albaugh, Scott E Hudson, and Lining Yao. 2023. Physically Situated Tools for Exploring a Grain Space in Computational Machine Knitting. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (<conf-loc>, <city>Hamburg</city>, <country>Germany</country>, </conf-loc>) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 736, 14 pages.

S. Kh. Aranson, G.R. Belitsky, and E.V. Zhuzhoma. 1996. *Introduction to the Qualitative Theory of Dynamical Systems on Surfaces*. Translations of Mathematical Monographs, Vol. 153. American Mathematical Society.

Harsh Bhatia, Shreeraj Jadhav, Peer-Timo Bremer, Guoning Chen, Joshua A. Levine, Luis Gustavo Nonato, and Valerio Pascucci. 2011. Edge maps: Representing flow with bounded error. In *Proceedings of the 2011 IEEE Pacific Visualization Symposium (PACIFICVIS '11)*. IEEE Computer Society, USA, 75–82.

David Bommes, Timm Lempfer, and Leif Kobbelt. 2011. Global Structure Optimization of Quadrilateral Meshes. *Computer Graphics Forum* 30, 2 (2011), 375–384.

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. 2013. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum* 32, 6 (2013), 51–76.

Marcel Campen, Moritz Ibing, Hans-Christian Ebke, Denis Zorin, and Leif Kobbelt. 2016a. Scale-invariant directional alignment of surface parametrizations. In *Proceedings of the Symposium on Geometry Processing* (Berlin, Germany) *(SGP '16)*. Eurographics Association, Goslar, DEU, 1–10.

Marcel Campen, Cláudio T. Silva, and Denis Zorin. 2016b. Bijective Maps from Simplicial Foliations. *ACM Trans. Graph.* 35, 4, Article 74 (jul 2016), 15 pages.

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 courses* (Anaheim, California) *(SIGGRAPH '13)*. ACM, New York, NY, USA, 126 pages.

Fernando de Goes, Mathieu Desbrun, and Yiying Tong. 2016. Vector Field Processing on Triangle Meshes. In *ACM SIGGRAPH 2016 Courses* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 27, 49 pages.

Paweł Dłotko. 2012. A fast algorithm to compute cohomology group generators of orientable 2-manifolds. *Pattern Recognition Letters* 33, 11 (2012), 1468–1476. Computational Topology in Image Context.

Tobias Günther and Irene Baeza Rojo. 2020. Introduction to Vector Field Topology. In *Topological Methods in Data Analysis and Visualization VI*. Springer.

Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com

Megan Hofmann, Lea Albaugh, Ticha Sethapakadi, Jessica Hodgins, Scott E. Hudson, James McCann, and Jennifer Mankoff. 2019. KnitPicking Textures: Programming and Modifying Complex Knitted Textures for Machine and Hand Knitting. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 5–16.

Megan Hofmann, Lea Albaugh, Tongyan Wang, Jennifer Mankoff, and Scott E Hudson. 2023. KnitScript: A Domain-Specific Scripting Language for Advanced Machine Knitting. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (<conf-loc>, <city>San Francisco</city>, <state>CA</state>, <country>USA</country>, </conf-loc>) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 21, 21 pages.

Megan Hofmann, Jennifer Mankoff, and Scott E. Hudson. 2020. KnitGIST: A Programming Synthesis Toolkit for Generating Functional Machine-Knitting Textures. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 1234–1247.

Yuki Igarashi, Takeo Igarashi, and Hiromasa Suzuki. 2008. Knitting a 3D Model. *Computer Graphics Forum* 27, 7 (2008), 1737–1743.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant field-aligned meshes. *ACM Trans. Graph.* 34, 6, Article 189 (nov 2015), 15 pages.

Benjamin Jones, Yuxuan Mei, Haisen Zhao, Taylor Gotfrid, Jennifer Mankoff, and Adriana Schulz. 2022. Computational Design of Knit Templates. *ACM Transactions on Graphics* 41, 2 (April 2022), 1–16.

David Jourdan, Pierre-Alexandre Hugron, Camille Schreck, Jonàs Martínez, and Sylvain Lefebvre. 2023. Shrink & Morph: 3D-Printed Self-Shaping Shells Actuated by a Shape Memory Effect. *ACM Trans. Graph.* 42, 6, Article 187 (dec 2023), 13 pages.

Alexandre Kaspar, Liane Makatura, and Wojciech Matusik. 2019. Knitting Skeletons: Computer-Aided Design Tool for Shaping and Patterning of Knitted Garments. *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*.

Alexandre Kaspar, Kui Wu, Yiyue Luo, Liane Makatura, and Wojciech Matusik. 2021. Knit Sketching: from Cut & Sew Patterns to Machine-Knit Garments. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 40, 4 (2021).

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe patterns on surfaces. *ACM Transactions on Graphics* 34, 4 (July 2015), 39:1–39:11.

Heng Liu and David Bommes. 2023. Locally Meshable Frame Fields. *ACM Transactions on Graphics* 42, 4 (2023).

Rahul Mitra, Liane Makatura, Emily Whiting, and Edward Chien. 2023. Helix-Free Stripes for Knit Graph Design. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–9.

Ioanna Mitropoulou, Amir Vaxman, Olga Diamanti, and Benjamin Dillenburger. 2024. Fabrication-aware strip-decomposable quadrilateral meshes. *Computer-Aided Design* 168 (2024), 103666.

Juan Sebastian Montes Maestre, Yinwei Du, Ronan Hinchet, Stelian Coros, and Bernhard Thomaszewski. 2023. Differentiable Stripe Patterns for Inverse Design of Structured Surfaces. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–14.

Georges Nader, Quek-Yu Han, Chia-Pei Zhi, Oliver Weeger, and Sai-Kit Yeung. 2021. KnitKit. *ACM Transactions on Graphics (TOG)* (July 2021).

Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James Mccann. 2018. Automatic Machine Knitting of 3D Meshes. *ACM Transactions on Graphics* 37, 3 (Aug. 2018), 35:1–35:15.

Vidya Narayanan, Kui Wu, Cem Yuksel, and James McCann. 2019. Visual knitting machine programming. *ACM Transactions on Graphics* 38, 4 (July 2019), 63:1–63:13.

Yuta Noma, Nobuyuki Umetani, and Yoshihiro Kawahara. 2022. Fast Editing of Singularities in Field-Aligned Stripe Patterns. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 37, 8 pages.

Nico Pietroni, Marcel Campen, Alla Sheffer, Gianmarco Cherchi, David Bommes, Xifeng Gao, Riccardo Scateni, Franck Ledoux, Jean Remacle, and Marco Livesu. 2022. Hex-Mesh Generation and Processing: A Survey. *ACM Trans. Graph.* 42, 2, Article 16 (oct 2022), 44 pages.

Nicolas Ray and Dmitry Sokolov. 2014. Robust Polylines Tracing for N-Symmetry Direction Field on Triangulated Surfaces. *ACM Trans. Graph.* 33, 3, Article 30 (jun 2014), 11 pages.

Christian Schüller, Roi Poranne, and Olga Sorkine-Hornung. 2018. Shape Representation by Zippables. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 37, 4 (2018).

Thibault Tricard, Vincent Tavernier, Cédric Zanni, Jonàs Martínez, Pierre-Alexandre Hugron, Fabrice Neyret, and Sylvain Lefebvre. 2020. Freely orientable microstructures for designing deformable 3D prints. *ACM Trans. Graph.* 39, 6 (2020), 211–1.

Amir Vaxman, Marcel Campen, Olga Diamanti, David Bommes, Klaus Hildebrandt, Mirela Ben-Chen Technion, and Daniele Panozzo. 2017. Directional field synthesis, design, and processing. In *ACM SIGGRAPH 2017 Courses* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 12, 30 pages.

Josh Vekhter, Jiacheng Zhuo, Luisa F Gil Fandino, Qixing Huang, and Etienne Vouga. 2019. Weaving Geodesic Foliations. *ACM Trans. Graph.* 38, 4, Article 34 (jul 2019), 22 pages.

H. Whitney. 1957. *Geometric Integration Theory.* Princeton University Press.

Kui Wu, Xifeng Gao, Zachary Ferguson, Daniele Panozzo, and Cem Yuksel. 2018. Stitch meshing. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–14.

Kui Wu, Hannah Swan, and Cem Yuksel. 2019. Knittable Stitch Meshes. *ACM Transactions on Graphics* 38, 1 (Feb. 2019), 1–13.

Cem Yuksel, Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2012. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Transactions on Graphics (TOG)* (July 2012).

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2006. Vector Field Design on Surfaces. *ACM Trans. Graph.* 25, 4 (oct 2006), 1294–1326.

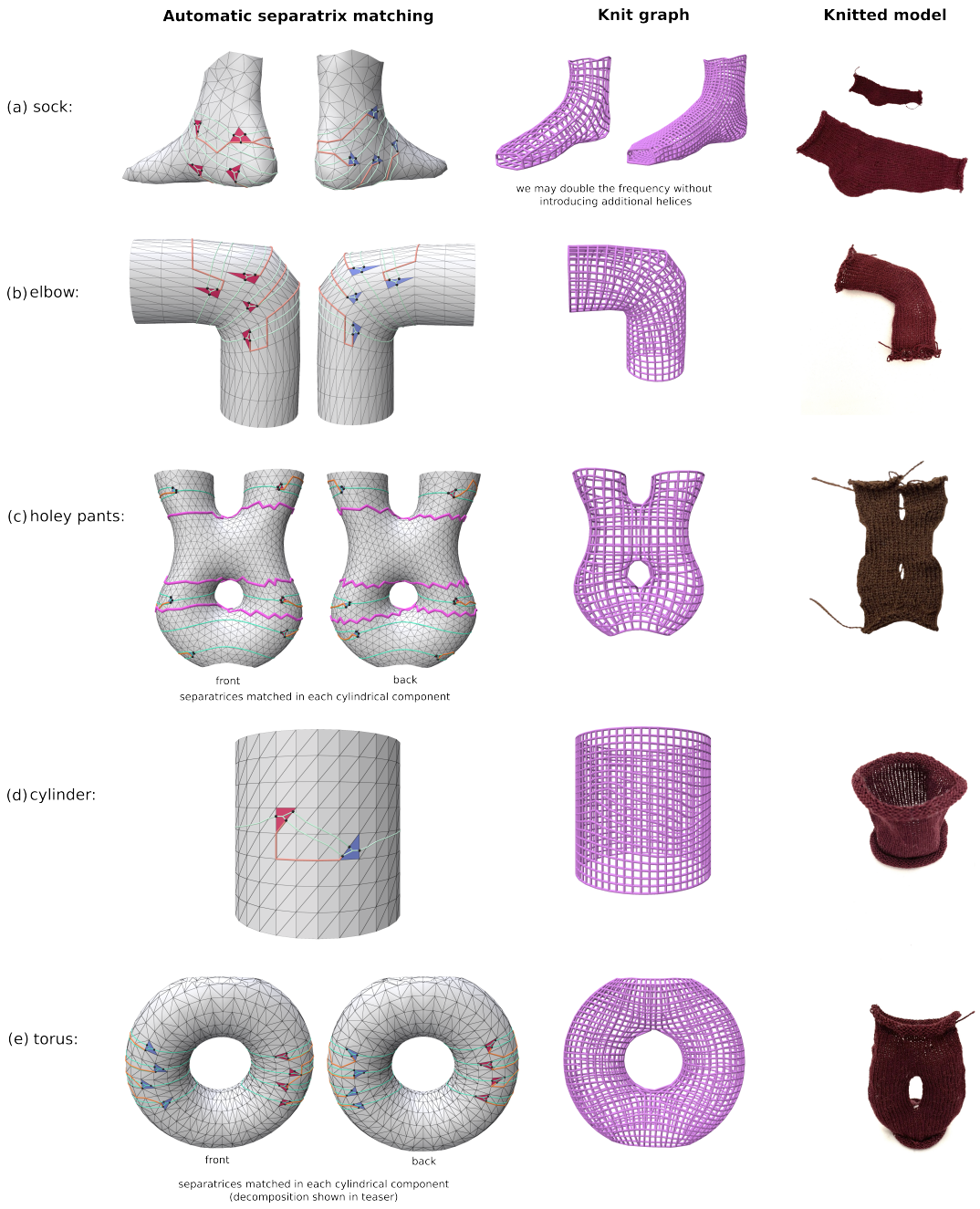



**Figure 9: Results. (a) Singularities generated by [Knöppel et al. 2015] are matched and the corresponding level set constraints are shown in orange. We double the frequency of the stripe pattern and still achieve a valid knit graph of higher resolution without additional helicing. (b) Edited singularities from [Knöppel et al. 2015] are matched to generate a knit graph. (c) Our method is able to handle models with non-zero genus via a cylindrical decomposition shown in pink. Using our optimal matching scheme, singularities are matched in each cylindrical component. (d) A pair of singularities are matched using a level set constraint. This induces a forced short row in the knit graph. (e) Edited singularities from [Knöppel et al. 2015] are matched in each cylindrical component of the torus model (decomposition shown in teaser).**

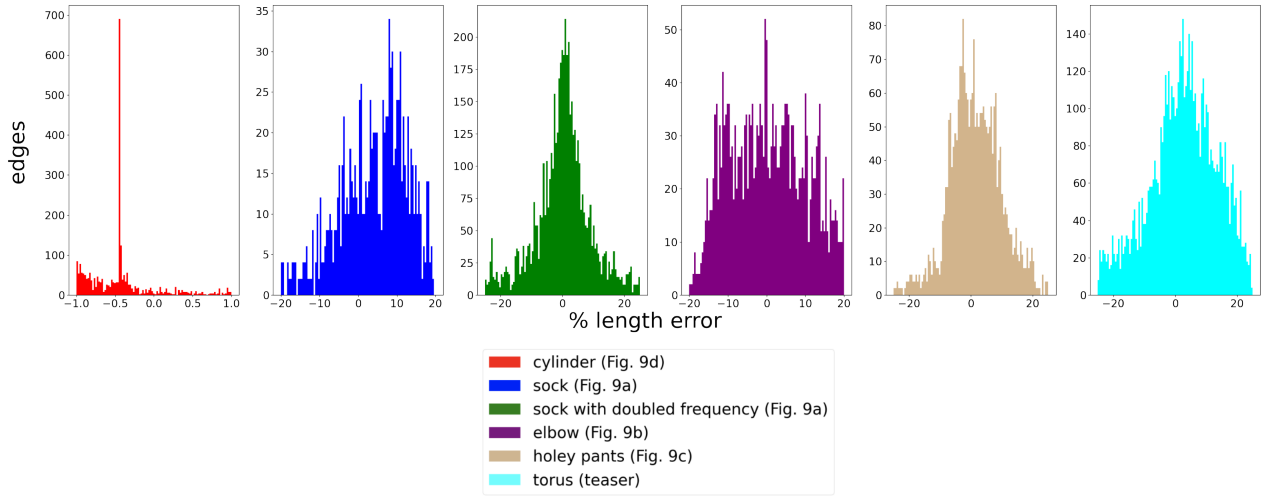



**Figure 10: Edge length error histograms for the fabricated models. While quantifying geometric fidelity is challenging due to the ability of knits to stretch and be malleable, we note that our histograms are similar to those of [Mitra et al. 2023]. Befitting the use of our $L^2$ objective, our histograms aren't as strongly concentrated near 0 as those of [Narayanan et al. 2018] but do present significantly fewer outliers. As is the case with [Narayanan et al. 2018], most of our edge length errors are $< 10\%$.**

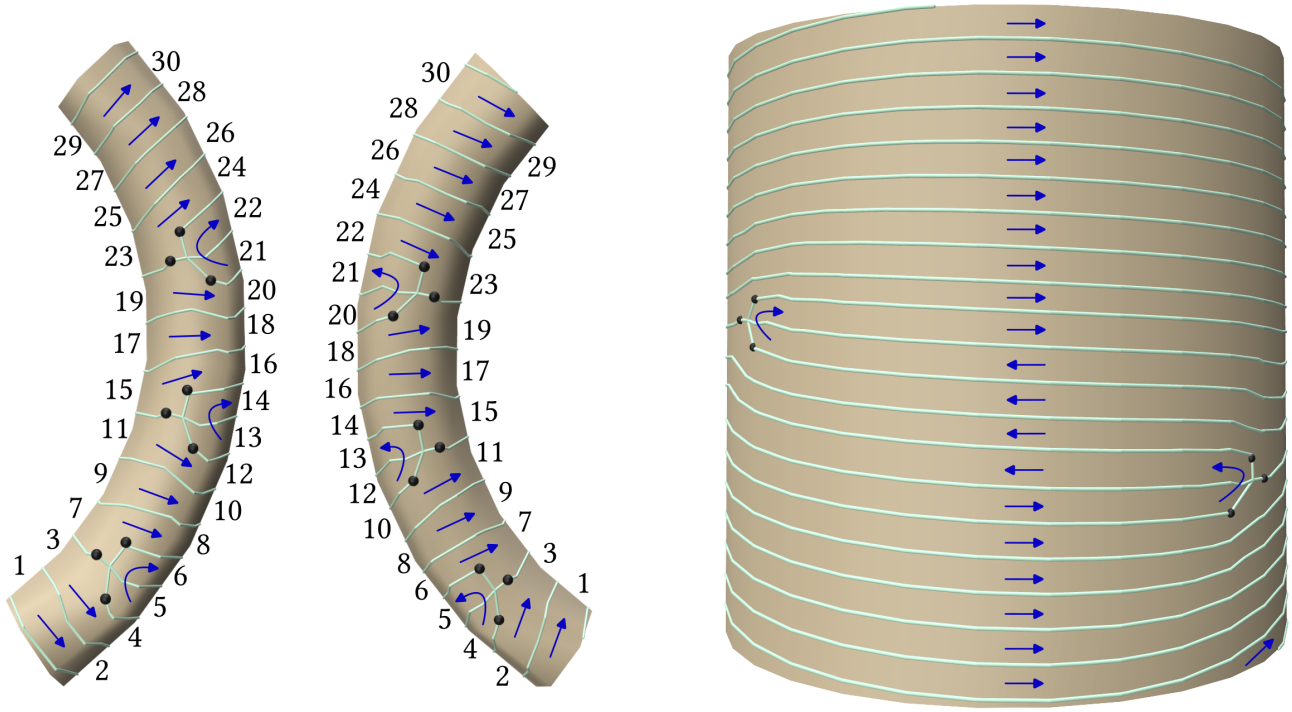


**Figure 11: Implied yarn paths following the blue arrows. Left: the implied path winds up the bent cylinder (both sides of the model shown), visiting each number in turn, and tracing out the short rows. This example is generated by doubling the singular triangle indices, and allowing the boundary integral constraints to equal $+P$ and $-P$ on the top and bottom, respectively. Right: the implied path winds up the cylinder, and reverses direction between the short row ends, helicing fully around the cylinder in the opposite direction several times. This was generated with the same $\pm P$ boundary integral constraints and a single level set constraint $\int_\gamma \sigma_c = \tilde{k}P$ joining separatrices, where $-\tilde{k}$ denotes the number of opposite direction helices desired.**