Curl Quantization for Automatic Placement of Knit Singularities

RAHUL MITRA*, Boston University, USA and LightSpeed Studios, USA
MATTÉO COUPLET, Boston University, USA
TONGTONG WANG, LightSpeed Studios, China
MEGAN HOFFMAN, Northeastern University, USA
KUI WU, LightSpeed Studios, USA
EDWARD CHIEN, Boston University, USA

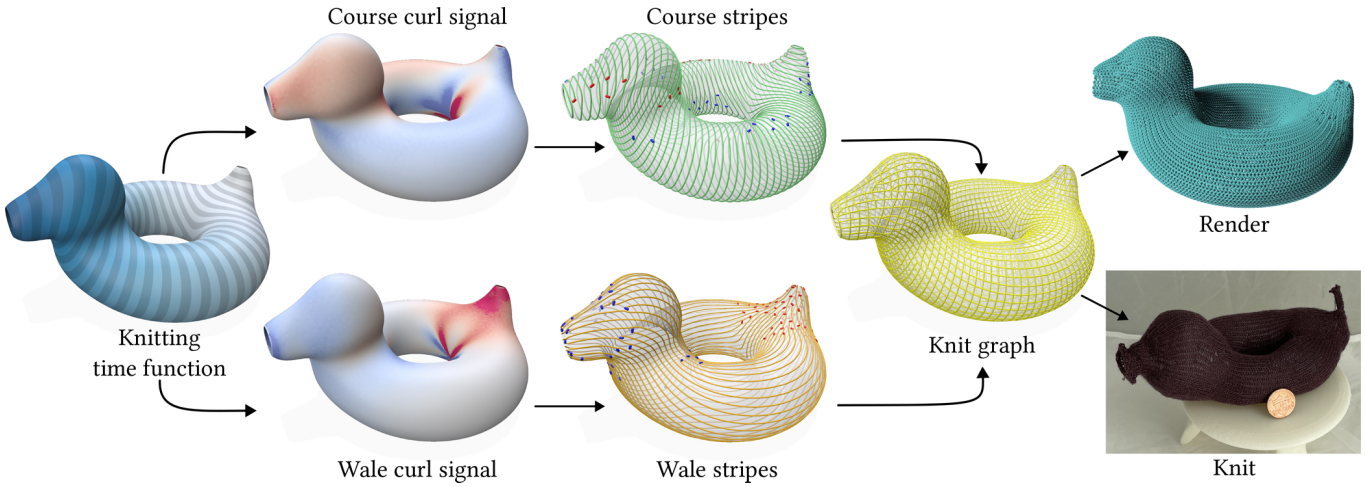


Fig. 1. *A knitting time function is computed over an input model, and the curl signal of its gradient is measured in the two orthogonal knitting directions, course and wale. Our method automatically places singularities in regions of high curl while satisfying all structural manufacturing constraints. The orthogonal stripe patterns are intersected to generate a smooth knit graph suitable for both artifact-free yarn-level rendering and machine-knitting.*

We develop a method for automatic placement of knit singularities based on curl quantization, extending the knit-planning frameworks of Mitra et al. [2024, 2023]. Stripe patterns are generated that closely follow the isolines of an underlying knitting time function, and has course and wale singularities in regions of high curl for the normalized time function gradient and its 90° rotated field, respectively. Singularities are placed in an iterative fashion, and we show that this strategy allows us to easily maintain the structural constraints necessary for machine-knitting, e.g., the helix-free constraint, and to satisfy user constraints such as stripe alignment and singularity placement. Our more performant approach obviates the need for a mixed-integer solve [Mitra et al. 2023], manual fixing of singularity positions, or the running of a singularity matching procedure in post-processing [Mitra et al. 2024]. Our global optimization also produces smooth knit graphs that provide quick simulation-free previews of rendered knits without the surface artifacts of competing methods. Furthermore, we extend our method to the popular cut-and-sew garment design paradigm. We validate our method by machine-knitting and rendering yarn-based visualizations of prototypical models in the 3D and cut-and-sew settings.

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

Additional Key Words and Phrases: Computational Knitting, Vector Fields

ACM Reference Format:
Rahul Mitra, Mattéo Couplet, Tongtong Wang, Megan Hoffman, Kui Wu, and Edward Chien. 2025. Curl Quantization for Automatic Placement of Knit Singularities. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3721238.3730715

1 Introduction

Machine knitting is an additive manufacturing technique that creates garments by interlocking yarn loops through a CNC knitting machine (Computer Numerical Control knitting machine). It has

*Part of this work was done when Rahul Mitra was an intern at LIGHTSPEED.

Authors' Contact Information: Rahul Mitra, Boston University, Boston, USA and LightSpeed Studios, Los Angeles, USA, rahulm@bu.edu; Mattéo Couplet, Boston University, Boston, USA, mcouplet@bu.edu; Tongtong Wang, LightSpeed Studios, Shen Zhen, China, wangtong923@gmail.com; Megan Hoffman, Northeastern University, Boston, USA, m.hofmann@northeastern.edu; Kui Wu, LightSpeed Studios, Los Angeles, USA, walker.kui.wu@gmail.com; Edward Chien, Boston University, Boston, USA, edchien@bu.edu.

SIGGRAPH Conference Papers '25, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/25/08
https://doi.org/10.1145/3721238.3730715

been widely used in everyday applications, including clothing, home decor, and personal accessories. With advances in functional knitting, e.g., actuation [Albaugh et al. 2019; Luo et al. 2022], tactile [Albaugh et al. 2021], and sensing [Luo et al. 2021], there is an increasing demand for a system capable of converting 3D shapes into knittable structures, serving as carriers for various functionalities in architecture [Sharmin and Ahlquist 2016], robotic coverings [Zlokapa et al. 2022], and conformal sportswear [Liu et al. 2021]. However, it is a labor-intensive process to design knitted garments, requiring experienced knitting engineers to manually translate artists' designs into machine instructions. It is an even more challenging task to automatically transform general 3D shapes into machine-knittable structures due to the specific geometric and structural constraints required by the nature of the interlocking yarns and machine constraints during fabrication.

Previous works have attempted both local greedy algorithms and global parametrization approaches to tackle this problem. Unfortunately, these methods do not incorporate user constraints [Narayanan et al. 2018] or fail to guarantee knittability [Wu et al. 2018]. The latter often requires cutting along direction-mismatched edges [Jones et al. 2021; Wu et al. 2022], which must be sewn afterward. This seam requires additional labor, hindering a fully automated fabrication process, and is also unacceptable for animation purposes, as it necessitates specialized knit structures and introduces visible bump artifacts.

We solve this problem by building upon two prior works [Mitra et al. 2024, 2023] that approach stitch structure planning from a stripe-texturing perspective [Knöppel et al. 2015], and focus on maintaining helix-free course stripes and thus knit structures. As shown in Fig. 2, knit singularities – i.e., increases and decreases in the wale direction and short rows in the course direction – are essential for creating structures that conform to the input shape. We introduce a faster, iterative automatic method for singularity placement, which does not need to solve an expensive mixed-integer problem or borrow placements from striping methods that may need manual adjustment if not suitable. Instead, singularity placement is considered as a curl quantization problem, as stripe bifurcations, corresponding to knit irregularities, occur at integer curl localizations. For course stripes/rows, we approximate initial curl density of normalized knitting time function gradient (§4.1) with integer placements, similar to early approaches to quad meshing with vertices of high/low valence (>/< 4) placed in regions of low/high Gaussian curvature [Ben-Chen et al. 2008].

Our greedy iterative approach allows us to maintain structural and user constraints as we place singularities at regions of highest curl. In the course direction, singularities are placed in pairs on isolines of the time function, ensuring helix-free rows. In the wale direction, we have no such constraint, but we ensure that singularities are far enough from each other for construction of a valid knit graph. After each iteration, our 1-form (stripe pattern) evolves and its curl is used as the signal to place the next singularity.

A novel discretization is used in the 1-form setting, with discrete 1-forms on a *lens complex* [Soliman et al. 2021]. Curl singularities are placed on edges of the input mesh, as opposed to faces as in prior works. Crucially, this formulation allows for simpler and more robust stripe tracing to produce the knit graph, and maintains the foliation guarantees of [Mitra et al. 2024] in a simpler fashion.

We demonstrate our method in two related settings: that of arbitrary 3D input meshes and that of intrinsically-represented 2D cut & sew panels for garments [Kaspar et al. 2021; Korosteleva and Sorkine-Hornung 2023]. The latter is a more common setting in industrial garment design and boosts the applicability of our method. We fabricate models in both settings and show that our method achieves comparable uniformity in stitch size to Autoknit [Narayanan et al. 2018], but smoother stitch patterns. Additionally, we demonstrate an application of our method for generating high-quality visual outputs, enabling knit structure previews and facilitating applications in virtual applications, such as VFX and games. Prior methods do not ensure truly plausible knit structures, resulting in visible artifacts on the surface [Wu et al. 2018], or do not produce globally smooth stitch patterns [Narayanan et al. 2018].

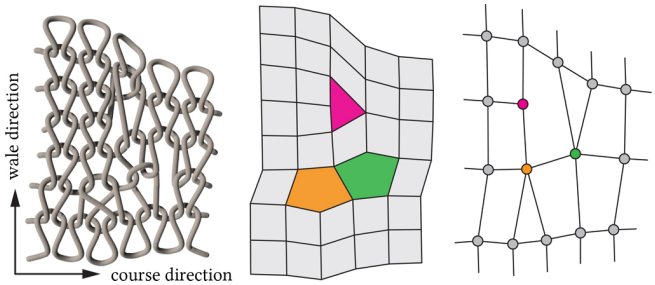


Fig. 2. *A close-up of a knit structure and three stitch irregularities (left), along with the corresponding stitch mesh [Yuksel et al. 2012] (middle) and knit graph (right) representations. A decrease (orange), an increase (green), and a short-row end (pink) are shown. The doubled wale resolution of stitch mesh, as a single knit graph vertex, corresponds to two stacked stitches (in wale direction) [Narayanan et al. 2018].*

To recap, our contributions are:

- A curl quantization formulation for singularity placement.
- An automatic iterative solution strategy that ensures machine-knittability (e.g., helix-free) and user constraints (e.g., singularity avoidance), and achieves comparable stitch size regularity.
- A curl placement on edges for more robust knit graph tracing.
- An extension to patch-based (intrinsic) garment models.

2 Related Work

Knit structures are formed by interlacing yarn into successive loops (Fig. 2), creating rows of *stitches* called *courses*. The direction of knitting rows over time is known as *wale* direction, and columns of stitches are *wales*. While the standard *knit* stitch, produced by pulling a new loop through the back of an existing loop, is the most common, various irregular stitches are used to shape the fabric, including *increase* (adding extra loops to expand), *decrease* (merging loops to narrow), and *short-row* (partial rows to create local shaping), which are used to create complex knit shapes and textures.

Yuksel et al. [2012] first introduced *stitch meshes*, which abstract interlocked stitch structures into quad-dominant meshes, with pentagonal or triangular faces for irregular stitches. Such meshes were

primarily intended for rendering and animation purposes. Later, Wu et al. [2019] extended this framework for hand-knitting. To bridge the gap between 3D modeling and industrial knitting machines, McCann et al. [2016] brought the knit compiler to the graphics community, which can schedule knitting time and needles and generate low-level knitting machine instructions for automated knitting. This work also introduced the *knit graph* representation for a stitch structure, which is roughly dual to the stitch mesh representation (see Fig. 2). Several follow-up works have further refined this compiler, e.g., completeness [Lin et al. 2023], ease-of-use [Lin et al. 2024], and abstractions [Hofmann et al. 2023]. Leveraging the knit compiler and automatic design-to-fabrication pipelines, numerous functionalities have been realized in the form of knitting, including soft actuation [Albaugh et al. 2019], tactile [Albaugh et al. 2021], sensing [Luo et al. 2021; Zlokapa et al. 2022], deformation absorption [Liu et al. 2021], pneumatic actuation [Luo et al. 2022], and illusion [Zhu et al. 2024]. In this work, we use stitch meshes [Yuksel et al. 2012] and knit graphs [Narayanan et al. 2018] to represent knit structures for rendering and machine knitting, respectively.

In addition to advancements in knit compilers and functional knitting, significant efforts have focused on developing more intuitive and automated design pipelines, e.g., shape primitives [Kaspar et al. 2019; McCann et al. 2016], cut & sew patterns [Kaspar et al. 2021], and 2D design interfaces [Twigg-Smith et al. 2024a,b]. Despite these strides, automatically transforming general 3D shapes into machine-knittable designs remains a challenging problem. Wu et al. [2018] introduced an automatic pipeline to convert arbitrary shapes into quad-dominant stitch meshes. However, their method does not guarantee the resulting meshes are machine-knittable. Narayanan et al. [2018] proposed a computational approach that "peels" stitch rows from a 3D mesh, aligning them to a time field. This was later extended by Narayanan et al. [2019], enabling stitch-type editing through an augmented stitch mesh structure. Instead of iterative mesh peeling, Jones et al. [2021] utilized quad meshing with user-defined singularities as input. Any mismatched edges in the resulting mesh were cut and treated as seams. Wu et al. [2022] introduced a physics-based simulation to minimize the cutting-edge length, ensuring the resulting knit object could be physically worn.

Nader et al. [2021] first treated knit graph generation as a global stripe optimization problem using the framework of Knöppel et al. [2015]. However, their method could not guarantee knittability due to the presence of helices in the stripe patterns. A quad-meshing-based fix for these helicing stripes is suggested, but comes without guarantees. Addressing this limitation, Mitra et al. [2023] introduced several linear constraints within the space of differential 1-forms to more robustly eliminate helices, ensuring the generation of helix-free knit graphs. However, in their work, singularity positions originated from either Knöppel et al. [2015], which may have stripes escape out of the boundaries and poorly aligned singularities, or by solving a very expensive mixed-integer problem (Strategy 2) where the number of integer variables scales with the number of faces in the input mesh. Building on the stripes line of work, Mitra et al. [2024] presented a topological understanding of stripe patterns using the concept of singular foliations. While their method allowed for the automatic pairing of optimal singularities, the placement of these singularities remained manual, which is both time-consuming and labor-intensive. This work extends the framework of global stripe pattern optimization [Mitra et al. 2024, 2023] by presenting a method for the automatic placement of knit singularities. Our greedy curl-informed algorithm does not require solving a large mixed-integer problem or manual placement and fixing of singularities and allows for the incorporation of several user constraints.

3 Background

Vector fields, integrability, and curl. We recall a few important definitions here and refer the reader to Crane et al. [2013]; de Goes et al. [2016] for more detail. A *vector field* V on a surface M is a continuous specification of a vector at each point. One common example is the gradient of a differentiable function $f : M \to \mathbb{R}$, denoted ∇f. Most vector fields are not gradients of a function, but if they are, we say that they are *integrable*. On domains that are simply connected (disc-like, i.e., homeomorphic to $\mathbb{D}^2$) a vector field is integrable if and only if its curl vanishes: $\nabla \times V = 0$. Taking a small enough neighborhood around any point on M gives a simply-connected set, so vanishing curl is equivalent to *local integrability*, and if $\nabla \times V = 0$ in such a small neighborhood, one can define a function g locally for which $V = \nabla g$.

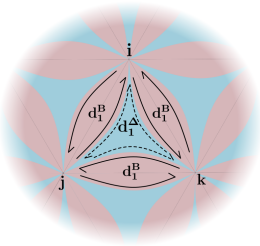
In our discretized setting, the input is a triangular mesh $M = (V, E, F)$, with sets of vertices, edges, and faces, respectively. A vector field may be represented as a discrete 1-form $\sigma : E \to \mathbb{R}$, an assignment of a real number to each oriented edge. The curl of such a vector field is given by the first exterior derivative operator, d_1, an $|F| \times |E|$ matrix that calculates the oriented sum of σ around each triangular face. As in the continuous setting, if $(d_1\sigma)_t = 0$, it is locally integrable and we may define a linear function g over t such that σ is equal to the change in g over the edges of t.

Lastly, in our optimization we use the *Whitney interpolant* to interpret a discrete 1-form as a face-based vector field. This gives a formula for interpolating such a representation to the entire face of a triangle (see Eq. 16 in de Goes et al. [2016]), but we only use the value at the barycenter, and denote this:

$$\delta(\sigma_{ij}, \sigma_{jk}, \sigma_{ki}) := \frac{(\sigma_{ki} - \sigma_{ij})\mathbf{p}_{jk}^{\perp} + (\sigma_{ij} - \sigma_{jk})\mathbf{p}_{ki}^{\perp} + (\sigma_{jk} - \sigma_{ki})\mathbf{p}_{ij}^{\perp}}{6|t_{ijk}|} \tag{1}$$

In the above, $\mathbf{p}_{ij}^{\perp}, \mathbf{p}_{jk}^{\perp}, \mathbf{p}_{ki}^{\perp}$ denote the edge vectors rotated by $\pi/2$ with respect to the face normal, and $|t_{ijk}|$ denotes the triangle area.

Form-based stripes. The texturing function for a stripe pattern may be viewed as a periodic function $\alpha : M \to \mathbb{S}^1$. Following Knöppel et al. [2015], one may specify such a function via its gradient, with a discrete 1-form $\sigma : E \to \mathbb{R}$ that has integer curl singularities on faces $d_1\sigma = 2\pi\mathbf{k}$ with $\mathbf{k} \in \mathbb{Z}^{|F|}$. Note that the periodic nature of the texturing function allows integer multiples of curl, unlike the vanishing curl requirement mentioned in the previous subsection for regular scalar functions. Optimization directly in the space of discrete 1-forms was first considered in Noma et al. [2022] and built upon in Mitra et al. [2024, 2023]. An example of a texturing function on a singular ($k_f \neq 0$) triangle is shown in Fig. 4. Lastly, for the remainder of the paper, we replace 2π with the desired period P, which will reflect the course row height or wale column width.



Lens complex. For placement of singularities, we consider an extension of the standard triangle mesh structure that includes bigons for each of the triangle edges, referred to as a *lens complex* in Soliman et al. [2021] (see inset). This extends the set of faces $\tilde{F} = F \sqcup B$ to include the set of such bigons B, and doubles the set of edges, denoted $\tilde{E}$, turning each original half-edge into its own edge (the vertex set V remains fixed). This construction allows us to define 1-forms that disagree on twin half-edges of the original triangle mesh, and we place singularities on these bigons. This results in a piecewise linear stripe pattern on all triangular faces, allowing for simpler and more robust knit graph tracing (detailed in Supp. §4). The first exterior derivative extends naturally to this context and we use d_1^Δ and d_1^B to denote the blocks corresponding to d_1 on the triangular and bigon faces, respectively.

Structural manufacturing constraints. To use the machine-knitting compiler of Narayanan et al. [2018], one requires several properties of the knit graph, and we mention the most challenging ones here. Property 2 is the hardest, and asks that the knit graph be *helix-free*: two nodes in a course row may not be intersected by the same wale column. Intuitively, if this is violated, the course row is offset from itself as it winds around the shape. We collect Property 3 and 4 (limited node degree and simple short rows), into a practical constraint that we call *singularity separation*. Essentially, they ask that course/wale singularities be sufficiently far apart from each other. Lastly, Property 5: *simple splits/merges*, asks that there not be stitch singularities near critical isolines of the time function.

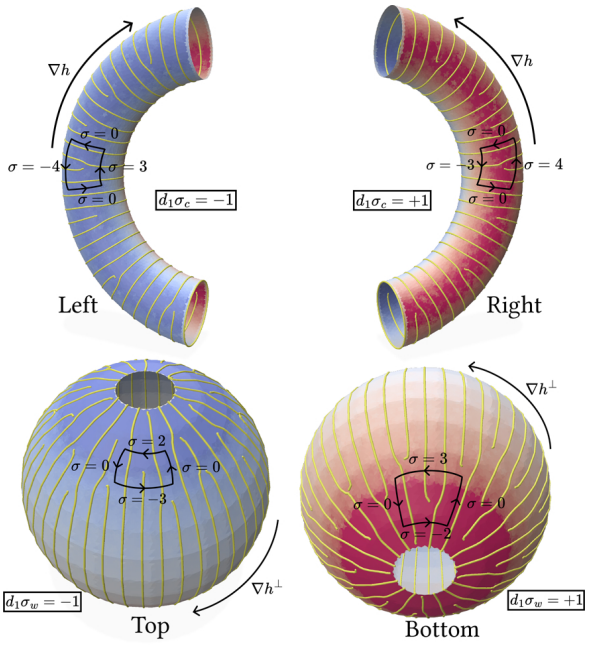


Fig. 3. *Top: $\nabla \times \overline{\nabla h}$ informs the placement of short row ends. Bottom: $\nabla \times \overline{\nabla h}^\perp$ informs the placement of wale increases (positive curl; red) and decreases (negative curl; blue). Integrating the 1-form σ_c or σ_w around a singularity yields its index. Intuitively, the curl is the limit of integrals around infinitesimal loops. Integrals over portions closer to the "spine" of the bent cylinder and to the poles of the sphere make larger (absolute) contributions than those further away.*

4 Method

We now describe our iterative optimization procedure to generate machine-knittable stitch structures. A pair of orthogonal course and wale stripe patterns are generated, with automatic singularity placement informed by the model geometry. These stripes are then traced to form a helix-free knit graph which may be fed to the knitting compiler [Narayanan et al. 2018] and printed on an industrial knitting machine. We present details of the tracing in Supp. §4.

4.1 A discrete curl quantization problem

For automatic singularity placement, we view the problem as one of *curl quantization* of a non-integrable vector field. In particular, for the course rows, this vector field is derived from the *knitting time function* $h : M \to [0, 1]$ which takes values 0 and 1 along starting and ending boundaries of M, the surface to be knit. The isolines of h specify the desired directionality of course rows.

Naively, one might try to use h directly as a texturing function, but the spacing between isolines, proportional to $1/\|\nabla h\|$, may vary widely over the surface. Thus, we use its normalized gradient to inform our course texture function and its rotated (counterclockwise) vector field for the wale texture function:

$$\overline{\nabla h} := \frac{\nabla h}{\|\nabla h\|} \qquad \& \qquad \overline{\nabla h}^\perp := \mathrm{Rot}_{\pi/2}\left(\overline{\nabla h}\right).$$

While ∇h is clearly integrable, $\overline{\nabla h}$ and $\overline{\nabla h}^\perp$ are not. Figure 3 shows that for course rows, positive/negative curl correspond to regions suitable for short row ends. For wale columns, positive/negative curl correspond to potential increase/decrease placements, respectively.

As described in §3, we aim for a valid texturing function by specifying a 1-form with curl quantized on bigon faces of the lens complex. In particular, for course rows we aim to find a nearly integrable 1-form $\sigma : \tilde{E} \to \mathbb{R}$ and a singularity index on bigons $\mathbf{k}_B \in \{-1, 0, +1\}^{|E|}$ in the following quadratic mixed-integer problem:

$$\min_{\sigma,\, \mathbf{k}_B} \quad \mathcal{F}(\sigma) := \sum_{t \in F} A_t \|(\delta\sigma)_t - (\overline{\nabla h})_t\|^2 \tag{2a}$$

$$\text{such that} \quad d_1^\Delta \sigma = 0 \tag{2b}$$

$$d_1^B \sigma = P\mathbf{k}_B \tag{2c}$$

For the wale columns, we replace $\overline{\nabla h}$ with $\overline{\nabla h}^\perp$ in the above problem. Going forward, we will use σ_c and σ_w to denote the 1-forms for the course/wale patterns, respectively. If an equation or statement applies to both patterns σ will be used. We append additional linear constraints that reflect structural and user constraints in our optimization, but we delay details of these to §4.3.

The problem above may be viewed as a discretization of a continuous one where the optimal vector field V has quantized curl equal to the sum of Dirac delta measures placed at singular points

$p_1, \ldots, p_n \in M$, scaled by ± 1 indices $k_1, \ldots, k_n \in \{-1, +1\}$.

$$\min_{V, p_1, \ldots, p_n} \quad \int_M \|V - \overline{\nabla h}\|^2 \tag{3a}$$

$$\text{such that} \quad \nabla \times V = P \sum_{i=1}^{n} k_i \delta(p_i) \tag{3b}$$

Unfortunately, this is ill-posed with objective (3a) diverging near the isolated curl singularities p_i. This is analogous to the discrete connection smoothness energy of Crane et al. [2010], which also diverges under refinement, but still proves useful for application.

4.2 An iterative framework

As mentioned previously, the quadratic mixed-integer problem outlined in Eqs. (2a),(2b),(2c) is not complete, and must be augmented by additional structural and user constraints. One important example is the helix-free knit graph requirement which imposes a loose desideratum that pairs of course singularities should lie on similar isolines of the time function h. This is enforced via linear path constraints described more fully in §4.3.1. Other examples include singularity separation and simple splits/merges, which are incorporated into the singularity selection process (§4.4).

Prior global methods for this and related problems [Jakob et al. 2015; Knöppel et al. 2015] cannot easily handle these essential additional constraints (amongst others). Thus, we pursue an iterative approach where we place singularities at high curl locations in each iteration, while maintaining satisfaction of these. In summary, at each step, we consider adding singularities to the current iterate according to steps roughly outlined below.

(1) Given a set of accumulated singularities, solve for stripes that minimize objective $\mathcal{F}(\sigma)$; terminate if $\mathcal{F}(\sigma)$ is not lowered.
(2) Evaluate the curl of the current iterate, correcting for local curl imparted by prior singularity placement.
(3) For courses, place pairs of singularities along time function isolines with the highest curl range. For wales, place a singularity on the edge with the highest absolute curl.

Lastly, we note that the course pattern is calculated before the wale pattern, as it is slightly more constrained with the helix-free condition, and singularity separation imposes further conditions on the second pattern to be optimized.

4.3 Iterate Optimization & Potential Termination

Let σ_s, $\sigma_{c/w,s}$ with $s = 0, 1, 2, \ldots$ denote iterates of our optimization, and $\mathbf{k}_{B,s}$, $\mathbf{k}_{B,s}^{c/w} \in \{-1, 0, +1\}^{|E|}$ denote the accumulated set of course/wale singularities at step s (by gathering ± 1 singularity indices on edge bigons). We initialize $\mathbf{k}_{B,0} = \mathbf{0}$ to denote no singularities selected. Given $\mathbf{k}_{B,s}$ we solve a constrained quadratic optimization for σ_s. This iterate will have the singularities selected to this point, and will also satisfy structural non-helicing constraints, boundary constraints, and Morse constraints akin to those found in Mitra et al. [2024, 2023]. For this problem, replace σ with σ_s and $\mathbf{k}_B$ with $\mathbf{k}_{B,s}$ in Problem (2), and include the linear constraints:

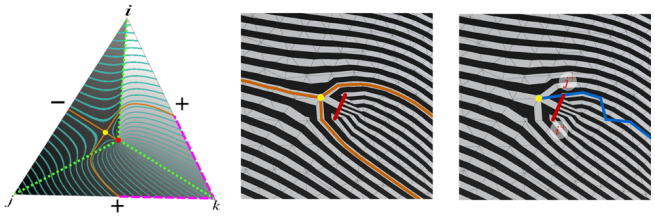


Fig. 4. *Foliation behavior near a +1 singularity, showing a source (red) and saddle (yellow) in close proximity, with level sets emanating to the right. In the left and middle images, separatrices are orange. Left: Singular foliation structure as in Knöppel et al. [2015] (from Mitra et al. [2024]). Middle: We place singularities on edges. Right: Non-helicing path constraint visualized in blue for the singular edge.*

$$\int_{\gamma_i} \sigma_{c,s} = 0 \qquad i = 1, \ldots, n_{\text{pairs}} \tag{4a}$$

$$\int_{\mu_i} \sigma_{c,s} = 0 \qquad i = 1, \ldots, n_{\text{Morse}} \tag{4b}$$

$$\sigma_{c,s}|_{\partial M} = 0 \tag{4c}$$

$$\sigma_s|_{l_j} = 0 \qquad j = 1, \ldots, n_{\text{align}} \tag{4d}$$

Eq. (4c) expresses course boundary alignment and Eq. (4d) expresses any user alignment constraints along edge sequences or cycles l_j where n_{align} is the number of such edge paths. An example of such constraints may be seen in Fig. 9 *left*. Eq. (4b) expresses path integral constraints along critical (in the Morse sense) isolines μ_i of the time function. All of the path integral constraints are discretized as simple oriented sums of 1-form values along edges. Eqs. (4b), (4c), and (4d) are identical to those from Mitra et al. [2024]. We detail Eq. (4a) below.

4.3.1 Non-helicing path constraints. For the course pattern, with each pair of singularities inserted, we introduce a non-helicing path constraint (4a). The integral is over a path γ_i that goes from the positive short row end singularity to the negative short row end singularity. In particular, let $e_{i^+ j^+}, e_{i^- j^-}$ denote the positive/negative singular edges and assume that the half-edge going from $i^{+/-}$ to $j^{+/-}$ is more aligned with ∇h. Then the path will run from the vertex opposite halfedge $e_{j^+ i^+}$ to the vertex opposite $e_{i^- j^-}$. See the rightmost image in Fig. 4 for a schematic. We show in Supp. §5 that these constraints preserve the foliation guarantees of [Mitra et al. 2024] ensuring that all level sets of the course stripes are helix-free.

The routing of the path is important, and we ask that it start by passing through vertex j^+ and end by passing through vertex i^-. The remainder of the path is found through a custom-weighted Dijkstra's shortest path on the halfedges of the mesh.

$$C_{ij} = 1 + \overline{\nabla h}^{\perp} \cdot \frac{\mathbf{p}_{ij}}{\|\mathbf{p}_{ij}\|} \tag{5}$$

These encourage movement from the positive singularity to the negative singularity along their shared isoline, as described in §4.2 of Mitra et al. [2024]. Further commentary on the linear constraints Eqs. (4a) - (4d), and (6) is provided in Supp. §2.

4.3.2 Termination and Integer Refinement. Upon completion of the optimization we compare the objective values and terminate if $\mathcal{F}(\sigma_s) > \mathcal{F}(\sigma_{s-1})$. As we greedily check potential edge-pair candidates and there are only a finite number of edges, termination of our algorithm is guaranteed. Practically, we never get close to exhausting all possibilities and terminate in ~20-50 iterations. For $s = 0$, we do not terminate.

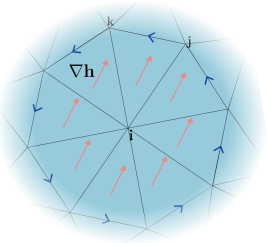
If termination occurs, we must add a few topologically correlated linear constraints and solve one last refinement optimization. In particular, we must add integrability constraints along homology generators to ensure a well-defined stripe pattern is used for knit graph construction.

$$\int_{g_l} \sigma = \mathbf{k}_g \qquad l = 1, \ldots, 2g + n_{\text{bdy}} - 1 \tag{6}$$

In the above, g is the topological genus of M, and n_{bdy} is the number of boundary components, and the g_l denote the homology generators of M. The homology generators are obtained with a standard tree-cotree algorithm, as described in §4.1 of Mitra et al. [2024].

For efficiency, the entries of $\mathbf{k}_g \in \mathbb{Z}^{2g+n_{\text{bdy}}-1}$ are estimated via a simple path integral of σ_c or σ_w over the generators, followed by a rounding to the closest integer. Empirically, we find that this does well and avoids the explicit use of *any* integer variables in our method. Lastly, we note that Eq. (6) barely affects local behavior of σ, so we do not enforce it when optimizing for iterates σ_s.

4.4 Singularity Placement



As described previously, we place singularities at regions of high (absolute) curl. In particular, we use the Whitney interpolant to produce a face-based vector field $\delta\sigma_s$ and calculate the per-vertex curl [de Goes et al. 2016; Wardetzky 2007]:

$$[\nabla \times \delta\sigma_s]_i = \frac{1}{A_{v_i}} \sum_{ijk \in N_1(v_i)} (\delta\sigma_s)_{ijk} \cdot \mathbf{p}_{jk} \tag{7}$$

The above specifies the curl at vertex i and captures the integral around the boundary of neighboring triangles (see inset); $A_{v_i} := \frac{1}{3}\sum_{ijk \in N_1(v_i)} A_{ijk}$ denotes the area of the corresponding 1-ring. As we are specifying singularities on edges, we average these values to edges to give a signal supported on edges: $[\nabla \times \delta\sigma_s]_{ij} = \left([\nabla \times \delta\sigma_s]_i + [\nabla \times \delta\sigma_s]_j\right)/2$. We alter this signal slightly to help satisfy singularity avoidance (denoted $[\nabla \times \delta\sigma_s]^{\sim}$), but delay details to §4.4.1 for pedagogical reasons.

Course rows. For short row end placements, isolines of h are sampled at regular intervals based on the period P of the desired stripe pattern (the course height). In particular we sample $N = 1/P\|\nabla h\|_{\text{avg}}$ isolines, where $\|\nabla h\|_{\text{avg}}$ denotes the area-weighted average of the time function norm. This achieves isolines that are spaced by P on average.

We denote T to be the set of sampled isolines and E_t to be the set of edges that a particular isoline $t \in T$ intersects. Then, a simple approach would be to seek the isoline $t_{\max}$ with max curl range:

$$t_{\max} = \arg\max_{t \in T} \left(\left(\max_{e \in E_t} [\nabla \times \delta\sigma_{c,s}]^{\sim}_e \right) - \left(\min_{e \in E_t} [\nabla \times \delta\sigma_{c,s}]^{\sim}_e \right) \right) \tag{8}$$

The max and min curl singular edges of $t_{\max}$ are incorporated into $\mathbf{k}^c_{B,s}$ with a +1 and −1, respectively. In later iterations, we remove $t_{\max}$ from consideration to help ensure short rows are not too close to each other (part of singularity separation).

In actuality, we found it more effective to try the top n_{cand} candidate isolines and stop when one of them improves the objective during our iterate optimization. For most of the results in this paper, we use $n_{\text{cand}} = 10$. Note that in effect, this subsumes the termination check of §4.3.2, but we present it as such for pedagogical simplicity. Lastly, we actually consider T as the set of connected components of isolines, so that we are not prevented from placing singular pairs on different tubes at the same isoline value.

Wale columns. For increases and decreases, we do not need to restrict to sets of isolines, as there is no helix-free condition to satisfy. Thus, the simplified strategy is to select the edge $e_{\max}$ with maximum absolute curl $\left|[\nabla \times \delta\sigma_{w,s}]^{\sim}\right|$. $e_{\max}$ is removed from consideration for future iterations, again to help satisfy singularity avoidance. As with the course pattern, we take the top n_{cand} candidate edges and stop when one of them improves the objective during our iterate optimization.

4.4.1 Local curl correction. When curl singularities are placed on an edge, there is a high compensatory curl nearby, drowning out the curl signal elsewhere. We correct for this local curl around singularities with two processes. The first is a solve for an *impulse function* of sorts, which approximates the local curl imparted by singularity placement. Specifically, we solve for a 1-form on the lens

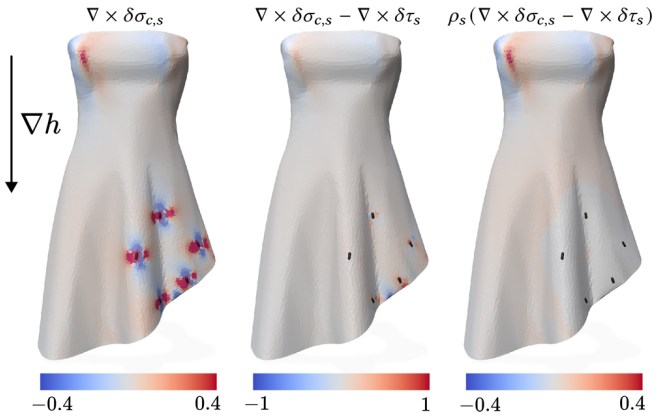


Fig. 5. *Curl signal of the course 1-form after inserting $s = 5$ short rows. Left: unaltered curl signal. Middle: an impulse function was subtracted off, removing most of the compensatory behavior and providing a better signal for the next singularity placement. Right: singularity separation masking was applied, enforcing manufacturing and rendering constraints by keeping short row ends, increases and decreases away from each other.*

complex τ_s with the following constrained quadratic problem.

$$\min_{\tau_s \in \mathbb{R}^{|\hat{E}|}} \quad \sum_{t \in F} A_t \|(\delta\tau_s)_t\|^2. \tag{9a}$$

$$\text{such that} \quad d_1^{\Lambda}\tau_s = 0 \tag{9b}$$

$$d_1^{B}\tau_s = \mathbf{k}_{B,s} \tag{9c}$$

$$\tau_s|_{\partial M} = 0 \tag{9d}$$

The solution captures the curl imparted by the current singularity placement, only specifying boundary alignment and singularity placement constraints. We subtract off $\nabla \times \delta\tau_s$ from $\nabla \times \delta\sigma_s$ when considering the curl for singularity placement. Fig. 5 shows these results in a curl measure that no longer peaks solely in the neighborhood of the singular edges, but is comparable over the mesh.

Additionally, we also want to avoid placements near the critical (saddle) isolines of h (simple splits/merges), and placements near existing singularities (singularity separation). To force satisfaction of these, we zero out the signal based on a geodesic distance estimate calculated via the heat method [Crane et al. 2017]. Let $\rho_s : M \to \mathbb{R}$ denote a function which is 0 if we are within a distance d_{mask} of the critical isolines, the existing singularity placements given by $\mathbf{k}_{B,s}$, and (in the wale case) the course singularities given by $\mathbf{k}_B^c$. Ultimately, our altered curl signal is given by:

$$[\nabla \times \delta\sigma_s]_i^{\sim} = \rho_s(v_i)\,([\nabla \times \delta\sigma_s]_i - [\nabla \times \delta\tau_s]_i) \tag{10}$$

4.5 User Control

Our framework allows for some easily incorporated user control over singularity placement, and we demonstrate some of these capabilities. First, as in Mitra et al. [2024, 2023], it is easy to place individual singularities by initializing them in $\mathbf{k}_{B,0}$. Secondly, alignment of courses and wales to particular paths may be achieved by simple linear constraints and specification of the l_j in Eq. (4d). A clear use case of this is in the cut-and-sew patch setting, where designers often expect wale patterns to align with boundaries of the patches, which we demonstrate in Fig. 9 *left*. Lastly, we demonstrate the novel tool of singularity avoidance (Fig. 9) *right* implemented via modification of Eq. (10). These may be done via hard constraints, by modification of ρ_s, or may be done in a soft fashion by adding another factor ρ_{seam} which is 1 near a user-designated seam and decreases to a non-zero value, e.g. 1/2, over the rest of the mesh. The latter option may be easily done with a shifted and scaled heat diffusion kernel [Crane et al. 2017] and allows for singularity placement away from the designated seam. This approach allows for far more intuitive user-control compared to Mitra et al. [2024, 2023]. By scaling the curl signal, we allow seam specification in a stroke-based manner (Fig. 9, *left*) and region-based singularity avoidance (Fig. 9, *right*). Mitra et al. [2024, 2023] would require tedious face-by-face specification of these singularity positions and optimal choices are often not clear. We infer these positions exactly.

5 Results

We solve the optimization problems using the vanilla Gurobi solver [Gurobi Optimization, LLC 2022] on an M3 MacBook Pro with 32GB of RAM. We validate our techniques through physical fabrication using an industrial knitting machine and visual verification by rendering yarn-level structures. Our highlighted results in Fig. 6 include both intrinsic cut-and-sew patch designs and extrinsic 3D-model-based geometries, and show that our method places singularities in curl-informed regions and results in evenly-sized stitches. Edge length error histograms for selected models showing comparable performance to AutoKnit are in Supp. §3. Auxiliary results beyond those in Fig. 6 and Fig. 7 are in Supp. §1.

Table 1. Runtime statistics and comparisons against Strategy 2 of Mitra et al. [2023]. The Sock, Curved cylinder, and Cactus are models from their work. Their method did not find any integer solution for Glove after 10 minutes, and does not handle the cut-and-sew setting (4-panel skirt, Pants).

Model	#V	#F	Strategy 2	Ours	Speed-up
Sock	279	538	21s	0.395s	~53×
Curved cylinder	54	96	<5s	0.025s	~182×
Cactus	391	736	35s	0.348s	~100×
Glove	37493	74524	N/A	56s	N/A
Dress	8634	16995	6min	11.521s	~31×
4-panel skirt	1094	1936	N/A	0.597s	N/A
Pants	1217	2046	N/A	0.526s	N/A

Fabrication Results. We use Autoknit [Narayanan et al. 2018] for knit scheduling. All samples were knitted on a Shima Seiki SWG91N2 15-gauge v-bed knitting machine using 2/28NM rayon yarn at a 40-stitch size at a rate of 0.6 m/s.

Rendering Results. Besides validating our results via machine knitting, we also render several outputs with yarn-level details. Particularly, we convert the knit graph to a stitch mesh [Yuksel et al. 2012] as its dual representation. As each type of stitch mesh face is embedded with a yarn-level template, we can generate the corresponding yarn structure and render with an offline path tracing render. No yarn-based relaxation is done. We compare to a stitch mesh derived from AutoKnit [Narayanan et al. 2018] and point out surface artifacts that arise from the stitch meshes of Wu et al. [2018] in Fig. 8. Our results are smoother and free of surface artifacts.

Runtimes. The cost of our method is dominated by the constrained quadratic problem solved for each singularity candidate, which amounts to a linear solve that scales with the number of edges. Possible performance improvements include preconditioning the linear solver (since the system remains mostly the same through iterations) and parallelizing the checking of singular edge candidates. Table 1 details the run times and compares against Mitra et al. [2023].

6 conclusion

This works presents an iterative, curl-informed method for automatic placement of knit singularities, improving the pipeline of Mitra et al. [2024, 2023]. This formulation avoids the need for an expensive mixed-integer solve, any manual correction of placements, or singularity matching in post-processing. Our smooth knit graphs are suitable for both machine-knitting and yarn-level rendering. We introduce linear constraints to satisfy all structural manufacturing constraints. Additionally, our method allows a high degree of user control as shown in Fig. 9 which are not offered by Narayanan et al.

[2018]. We demonstrate our method in both the 3D and the 2D cut-and-sew setting on a variety of diverse models.

Limitations & Future Work. Befitting the use of a greedy algorithm, our method does not incorporate underlying symmetry information. Developing a more global, non-iterative approach to singularity placement, while maintaining structural and user constraints, is a natural extension. Additionally, to ease specification of non-helicing path constraints, we scale mesh resolution to have an average edge length less than the desired course and wale periods. A deeper analysis of the local foliation structure near singular edges may allow us to coarsen the mesh significantly and improve performance. As with most knitting works, our time function is currently naively computed from user-specified start and end constraints. This may lead to poor behavior on topologically complex models as seen in Fig. 8. More sophisticated time function optimization is another natural direction to pursue.

Acknowledgments

We acknowledge the Belgian American Educational Foundation, Wallonie-Bruxelles International and the NSF (Grant No. 2341880).

References

Lea Albaugh, Scott Hudson, and Lining Yao. 2019. Digital Fabrication of Soft Actuated Objects by Machine Knitting. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13.

Lea Albaugh, James McCann, Scott E. Hudson, and Lining Yao. 2021. Engineering Multifunctional Spacer Fabrics Through Machine Knitting. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 498, 12 pages.

Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal Flattening by Curvature Prescription and Metric Scaling. *Computer Graphics Forum* 27, 2 (2008), 449–458.

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)*. Association for Computing Machinery, New York, NY, USA, Article 7, 126 pages.

Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum* 29, 5 (2010), 1525–1533.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2017. The Heat Method for Distance Computation. *Commun. ACM* 60, 11 (Oct. 2017), 90–99.

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.

Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual.

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* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 21, 21 pages.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-Aligned Meshes. *ACM Trans. Graph.* 34, 6 (Nov. 2015).

Benjamin Jones, Yuxuan Mei, Haisen Zhao, Taylor Gotfrid, Jennifer Mankoff, and Adriana Schulz. 2021. Computational Design of Knit Templates. *ACM Trans. Graph.* 41, 2, Article 16 (Dec. 2021), 16 pages.

Alexandre Kaspar, Liane Makatura, and Wojciech Matusik. 2019. Knitting Skeletons: A Computer-Aided Design Tool for Shaping and Patterning of Knitted Garments. 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, 53–65.

Alexandre Kaspar, Kui Wu, Yiyue Luo, Liane Makatura, and Wojciech Matusik. 2021. Knit sketching: from cut & sew patterns to machine-knit garments. *ACM Trans. Graph.* 40, 4, Article 63 (July 2021), 15 pages.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe patterns on surfaces. *ACM Trans. Graph.* 34, 4, Article 39 (July 2015), 11 pages.

Maria Korosteleva and Olga Sorkine-Hornung. 2023. GarmentCode: Programming Parametric Sewing Patterns. *ACM Trans. Graph.* 42, 6, Article 199 (2023), 15 pages.

Jenny Lin, Vidya Narayanan, Yuka Ikarashi, Jonathan Ragan-Kelley, Gilbert Bernstein, and James Mccann. 2023. Semantics and Scheduling for Machine Knitting Compilers. *ACM Trans. Graph.* 42, 4, Article 143 (July 2023), 26 pages.

Jenny Han Lin, Yuka Ikarashi, Gilbert Louis Bernstein, and James McCann. 2024. UFO Instruction Graphs Are Machine Knittable. *ACM Trans. Graph.* 43, 6, Article 206 (Nov. 2024), 22 pages.

Zishun Liu, Xingjian Han, Yuchen Zhang, Xiangjia Chen, Yu-Kun Lai, Eugeni L. Doubrovski, Emily Whiting, and Charlie C. L. Wang. 2021. Knitting 4D garments with elasticity controlled for body motion. *ACM Trans. Graph.* 40, 4, Article 62 (July 2021), 16 pages.

Yiyue Luo, Kui Wu, Tomás Palacios, and Wojciech Matusik. 2021. KnitUI: Fabricating Interactive and Sensing Textiles with Machine Knitting. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 668, 12 pages.

Yiyue Luo, Kui Wu, Andrew Spielberg, Michael Foshey, Daniela Rus, Tomás Palacios, and Wojciech Matusik. 2022. Digital Fabrication of Pneumatic Actuators with Integrated Sensing by Machine Knitting. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 175, 13 pages.

James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica Hodgins. 2016. A compiler for 3D machine knitting. *ACM Trans. Graph.* 35, 4, Article 49 (July 2016), 11 pages.

Rahul Mitra, Erick Jimenez Berumen, Megan Hofmann, and Edward Chien. 2024. Singular Foliations for Knit Graph Design. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 38, 11 pages.

Rahul Mitra, Liane Makatura, Emily Whiting, and Edward Chien. 2023. Helix-Free Stripes for Knit Graph Design. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 75, 9 pages.

Georges Nader, Yu Han Quek, Pei Zhi Chia, Oliver Weeger, and Sai-Kit Yeung. 2021. KnitKit: a flexible system for machine knitting of customizable textiles. *ACM Trans. Graph.* 40, 4, Article 64 (July 2021), 16 pages.

Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James Mccann. 2018. Automatic Machine Knitting of 3D Meshes. *ACM Trans. Graph.* 37, 3, Article 35 (Aug. 2018), 15 pages.

Vidya Narayanan, Kui Wu, Cem Yuksel, and James McCann. 2019. Visual knitting machine programming. *ACM Trans. Graph.* 38, 4, Article 63 (July 2019), 13 pages.

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.

Shahida Sharmin and Sean Ahlquist. 2016. Knit Architecture: Exploration of Hybrid Textile Composites Through the Activation of Integrated Material Behavior.

Yousuf Soliman, Albert Chern, Olga Diamanti, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2021. Constrained willmore surfaces. *ACM Trans. Graph.* 40, 4 (2021), 1–17.

Hannah Twigg-Smith, Yuecheng Peng, Emily Whiting, and Nadya Peek. 2024a. What's in a cable? Abstracting Knitting Design Elements with Blended Raster/Vector Primitives. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 62, 20 pages.

Hannah Twigg-Smith, Emily Whiting, and Nadya Peek. 2024b. KnitScape: Computational Design and Yarn-Level Simulation of Slip and Tuck Colorwork Knitting Patterns. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 860, 20 pages.

Max Wardetzky. 2007. *Discrete Differential Operators on Polyhedral Surfaces - Convergence and Approximation.* Dissertation.

Kui Wu, Xifeng Gao, Zachary Ferguson, Daniele Panozzo, and Cem Yuksel. 2018. Stitch meshing. *ACM Trans. Graph.* 37, 4, Article 130 (July 2018), 14 pages.

Kui Wu, Hannah Swan, and Cem Yuksel. 2019. Knittable stitch meshes. *ACM Trans. Graph.* 38, 1 (2019), 1–13.

Kui Wu, Marco Tarini, Cem Yuksel, James McCann, and Xifeng Gao. 2022. Wearable 3D Machine Knitting: Automatic Generation of Shaped Knit Sheets to Cover Real-World Objects. *IEEE Transactions on Visualization and Computer Graphics* 28, 9 (2022).

Cem Yuksel, Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2012. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Trans. Graph.* 31, 4, Article 37 (July 2012), 12 pages.

Amy Zhu, Yuxuan Mei, Benjamin Jones, Zachary Tatlock, and Adriana Schulz. 2024. Computational Illusion Knitting. *ACM Trans. Graph.* 43, 4, Article 152 (July 2024), 13 pages.

Lara Zlokapa, Yiyue Luo, Jie Xu, Michael Foshey, Kui Wu, Pulkit Agrawal, and Wojciech Matusik. 2022. An Integrated Design Pipeline for Tactile Sensing Robotic Manipulators. In *2022 International Conference on Robotics and Automation (ICRA)* (Philadelphia, PA, USA). IEEE Press, New York, NY, USA, 3136–3142.

Fig. 6. *Highlighted results, showing time functions, curl signals, knit graphs showing singularity placements, renders, and stuffed fabricated results (with 0.75 inch coin or dressed on 8 inch tall mannequin). 1. Sock. 2. Dress. 3. Cactus. 4. Pipes. 5. Pants. 6. Two-panel skirt. 7. Glove. Note that the leftward bend is a common tension artifact rather than a topological effect. Sock, skirt and pants fabricated at two sizes.*

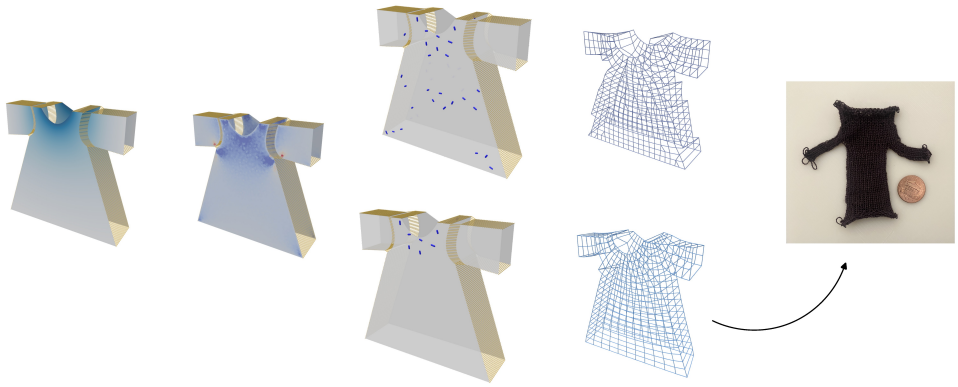


Fig. 7. *Cut & Sew T-shirt model with six-panels. From left to right: Time function. Wale curl signal. Middle top: Decrease placements of our method. We schedule the associated graph using a modified version of Autoknit's [Narayanan et al. 2018] scheduler, resulting in unsafe machine operations (original scheduler was thrashing on this intput). Due to the unsafe operations, we did not proceed with machine-knitting (dat file presented in supplementary). Middle bottom: We cap the number of inserted wale decreases. The associated graph and knitted result presents much less shaping but completes on the scheduler.*

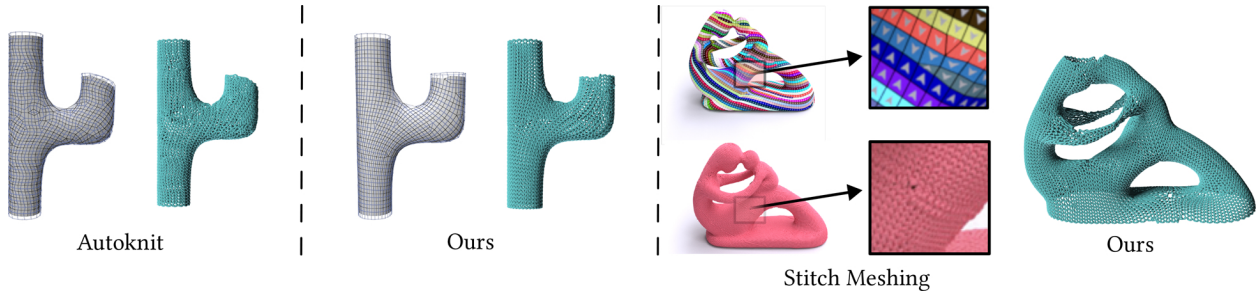


Fig. 8. *Rendering results. The stitch mesh is computed as the dual of the knit graph for template-based yarn-level rendering. Left: Stitch mesh obtained from Autoknit [Narayanan et al. 2018] has irregular columns and rows which are clearly visible in the render. Middle: Our stitch mesh presents far smoother rows and columns making it suitable for yarn-level rendering. Right: Stitch meshes from Wu et al. [2018] (with permission from author) present surface artifacts due to inconsistently oriented stitch faces and local remeshing. Our stitch meshes are guaranteed to be consistently oriented due to the machine-knittability of the produced knit graphs. Non-uniform stitches on the arms of the model arises due to poor time function choice on this model. Other knitting works also suffer from this problem, making the fabrication of complex topological models challenging. We leave time function optimization to future work.*

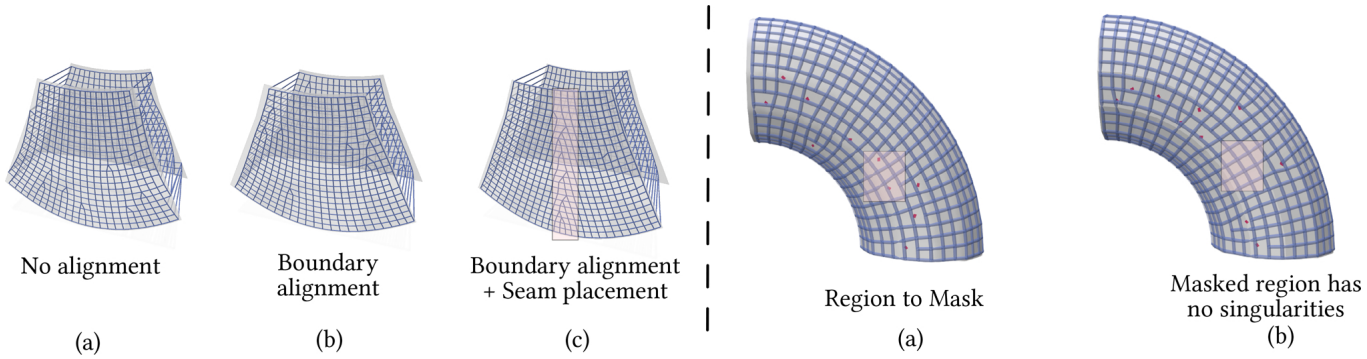


Fig. 9. *Our approach allows the incorporation of intuitive user constraints. Left: (a) Without any edge alignment constraints, the wale columns of the knit graph flow arbitrarily between panels in the cut & sew setting. (b) Our method allows for the alignment of wale columns to patch boundaries at stitched seams. (c) Our method also allows a user to place a seam (series of decreases) along the highlighted region. Right: (a) A geometrically informed solution from our algorithm would automatically place short rows along the sides of the model. (b) Our masking procedure allows a user to remove singularities from appearing in a particular region of the graph, allowing for color-work or texturing.*