Surface Power Diagrams for Knit Singularity Placement

RAHUL MITRA* and MATTÉO COUPLET*, Boston University, USA
RUICHEN LIU, Boston University, USA
JONATHAN NG, VARIANT3D, USA
RUZA MARKOV, VARIANT3D, USA
WILLIAM BATARA JEREMIAH SAMOSIR, VARIANT3D, USA
MEGAN HOFMANN, Northeastern University, USA
EDWARD CHIEN, Boston University, USA

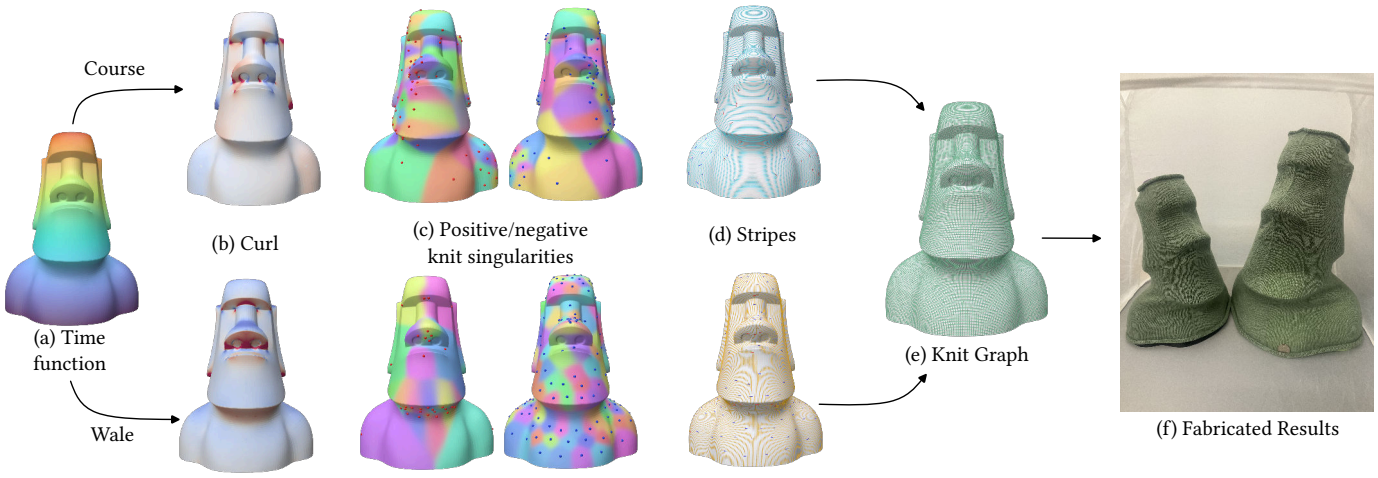


Fig. 1. Overview of our pipeline. (a) A knitting time function informs the direction of knitting. (b) A course and wale curl signal is computed from the normalized gradient of the time function. (c) Our method solves a semidiscrete optimal transport problem, represented by geodesic power diagrams, for both the positive and negative parts of the curl signal. (d) The centers of the cells inform placement of left/right short row ends for the courses and increases/decreases for the wales. (d) Using these singularities, a pair of orthogonal stripe patterns are computed for the course and wale directions, which are then (e) intersected to produce a helix-free knit graph. (f) Fabricated results of the Moai statue at two different knit graph resolutions dressed on 3D prints. The visualized graph corresponds to the larger knit.

We present an algorithm for global knit structure planning that leverages a generalization of power diagrams to triangulated surfaces. This generalization is based on modified geodesic heat kernels and is used to quantize the curl measure of a normalized knitting time function gradient. Knit singularity positions are optimized jointly in a global fashion via an iterative Lloyd-type algorithm, leading to faster and more optimal placement of singularities than prior work, allowing for practical creation of denser knit graphs. In this denser setting, we present singularity ordering constraints that more robustly achieve helix-free knit graphs. The speed and robustness of the method is demonstrated via a diverse array of knits, and a virtual gallery of helix-free knit graphs. We also provide further demonstration of user constraints for knit singularity masking, level set alignment constraints, and apparent seam placement via curl boosting.

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

Additional Key Words and Phrases: Computational Knitting, Vector Fields, Power Diagrams, Optimal Transport

ACM Reference Format:
Rahul Mitra, Mattéo Couplet, Ruichen Liu, Jonathan Ng, Ruza Markov, William Batara Jeremiah Samosir, Megan Hofmann, and Edward Chien. 2026. Surface Power Diagrams for Knit Singularity Placement. *ACM Trans. Graph.* 45, 4, Article 145 (July 2026), 20 pages. https://doi.org/10.1145/3811401

1 Introduction

Machine knitting has garnered much recent interest in the computer graphics, computational fabrication, and engineering communities for its ability to achieve complex shaping and geometry while avoiding the waste material inherent in sewing together woven fabrics. Beyond its ubiquitous use in everyday applications such as clothing

*Both authors contributed equally to this research.

Authors' Contact Information: Rahul Mitra, rahulm@bu.edu; Mattéo Couplet, mcouplet@bu.edu, Boston University, Boston, USA; Ruichen Liu, Boston University, Boston, USA, ruicliu@bu.edu; Jonathan Ng, VARIANT3D, USA, jonathan@variant3d.io; Ruza Markov, VARIANT3D, USA, ruza@variant3d.io; William Batara Jeremiah Samosir, VARIANT3D, USA, williamsamosir@variant3d.io; Megan Hofmann, Northeastern University, Boston, USA, m.hofmann@northeastern.edu; Edward Chien, Boston University, Boston, USA, edchien@bu.edu.

ACM 1557-7368/2026/7-ART145
https://doi.org/10.1145/3811401

and home decor, recent research has found applications of this geometric flexibility in sensing [Luo et al. 2021], architecture [Sharmin and Ahlquist 2016], composites [Leong et al. 2000], robotics [Zlokapa et al. 2022] and functional sportswear [Liu et al. 2021]. However, efficiently converting arbitrary 3D geometries into stitch structures ready to be machine-knit remains a complex challenge. There are various geometric and structural constraints that must be satisfied at the stitch level, often requiring painstaking manual specification by textile engineers. Furthermore, accommodating user input while satisfying these constraints adds additional complexity to the problem.

In this work, we present a method for planning a whole-garment stitch structure suitable for computational knitting, framed through the lens of global curl quantization. This framework allows for informed placement of *knit singularities*, i.e., *increases* and *decreases* in the wale direction and *short-row* ends in the course direction. These insertions are crucial for geometric shaping and maintaining uniform stitch sizes over an underlying 3D form. Our method comes with guarantees and capabilities that prior works lack: guaranteed machine-knittability [Nader et al. 2021; Wu et al. 2019], no post-processing (sewing) of misaligned seams [Jones et al. 2021], joint optimization of singularity positions [Mitra et al. 2024, 2023], and accommodation of user constraints [Narayanan et al. 2018].

Our work is in the vein of stripes-based [Knöppel et al. 2015] knit planning frameworks [Mitra et al. 2025, 2024, 2023; Nader et al. 2021] which extract a knit graph from two evenly-spaced orthogonal stripe patterns. Mitra et al. [2025] first considered curl measure as a heuristic for knit singularity placement. In particular, given a knitting time function $h : M \to \mathbb{R}$ that defines the ordering of course rows, the curl of its normalized gradient field $\nabla \times \frac{\nabla h}{||\nabla h||}$ (and its rotated counterpart for wales) serve as signals for placing singularities. Note that this curl is a signed quantity, with positive/negative parts denoting left/right short row ends (for the courses) and increases/decreases (for the wales). See Fig. 1 for an example. The method of [Mitra et al. 2025] is a greedy, iterative placement of knit singularities aiming to optimize for even spacing. Each singularity is placed individually, and each placement requires testing a set of candidate positions. For each such candidate, a convex quadratic optimization is required, leading to poor scaling with increase in knit resolution (more singularities).

We present a global method that jointly optimizes for all singularity positions, and then performs just one of these convex quadratic optimizations, avoiding the computational complexity of a greedy approach. We formulate the task of singularity placement as an optimal transport (OT) problem on the surface. In particular, we aim to quantize the curl measure $\nabla \times \overline{\nabla h}$ by finding a discrete set of knit singularities that minimizes the OT cost to the measure. The solution to such a problem is a *geodesic power diagram* on the surface as shown in Fig. 1. These are calculated on the surface utilizing heat kernel tools developed in [Crane et al. 2013b; Sharp et al. 2019c], and are alternated with Lloyd-type steps that update singularity positions. The computational efficiency of such a joint approach against the prior greedy method is highlighted in Table 1, allowing us to scale knit resolutions to highly dense, practically-sized models (see Fig. 1 and Fig. 17).

We also introduce novel ordering constraints and routing schemes on the stripe optimization (§4.4) which increase the robustness of helix-free knit graph construction (by satisfying the foliation guarantees of Mitra et al. [2024]). The robustness and fidelity of our method is validated on a variety of fabricated results throughout the paper and a virtual gallery of helix-free knit graphs which may be viewed at `rahulmitra.xyz/projects/powerknit/gallery`. Additionally, our method preserves the user capabilities of [Mitra et al. 2025] while introducing a few novel user editing tools. In particular, we allow for time function alignment (and thus course alignment) to feature curves (Fig. 13) and placement of "apparent" seams by a principled transferring of curl mass to user-specified curves (Fig. 14). We demonstrate these capabilities through a variety of practical examples highlighted in §5. We provide an open-source implementation which can be found at `github.com/mcouplet/powerknit`.

To summarize, our primary contributions include:

- A joint optimization scheme that globally solves for all singularity positions together, typically achieving 5-10x speedup over [Mitra et al. 2025]. (§5, Table 1)
- A heat-based method for calculation of geodesic power diagrams on surfaces, and a Lloyd-type alternating method for measure quantization on surfaces. (§3.3)
- Novel ordering constraints and path routing schemes that allow for robust helix-free knit graph generation in higher-frequency (knit density) settings. (§4.4)
- New user constraints allowing for direct editing of the knitting time function and apparent seam placement. (§4.6)

2 Related Work

Our work draws inspiration from several popular knitting pipelines and frameworks. The stitch meshing line of work [Wu et al. 2018; Yuksel et al. 2012] was the first to abstract yarn-level structures into quad-dominant meshes for rendering and visualization applications, but did not consider hand or machine knitting. Various works [Narayanan et al. 2019; Wu et al. 2019, 2022] built on this representation to allow for knittability. The first knitting compiler [McCann et al. 2016] was able to convert 2D primitives into machine knitting instructions. Autoknit [Narayanan et al. 2018] introduced an end-to-end pipeline for converting arbitrary 3D geometries into machine knitting instructions. We frame our knit structures through the popular *knit graph* abstraction introduced by this work and further described in §3.1.1 and Fig. 2.

There have been various follow-on knitting works focusing on knit structure semantics and topological correctness [Lin et al. 2023, 2024], designing programming language abstractions for knits [Hofmann et al. 2019, 2023], user-editing [Jones et al. 2021; Narayanan et al. 2019] and interfaces [Kaspar et al. 2019, 2021; Twigg-Smith et al. 2024]. Additionally, significant efforts have been focused in the domain of functional knitting. These include applications in actuation [Albaugh et al. 2019; Luo et al. 2021, 2022], deformation modeling [Liu et al. 2021], robotics [Zlokapa et al. 2022] and architecture [Popescu et al. 2021, 2020].

As noted, there are several other works that use stripe patterns for knit graph generation. These methods generate globally orthogonal stripe patterns and then intersect them to produce knit graphs.

Nader et al. [2021] were the first to consider such an approach by using the striping algorithms of Knöppel et al. [2015]. However, this method could not guarantee machine-knittability due to the presence of stripe helices which were locally fixed using tools from the quad meshing literature [Bommes et al. 2011]. Mitra et al. [2023] addressed this by presenting several linear constraints in the space of discrete spinning *1-forms* that allowed for the global modification of stripes to generate helix-free knit graphs. Follow-on work [Mitra et al. 2024] viewed stripe patterns as *foliations* of an underlying vector field and presented more robust criteria for guaranteeing the helix-free condition. Recent work [Mitra et al. 2025] tackled the problem of placing knit singularities using the curl of $\overline{\nabla h}$ as a heuristic signal. While [Mitra et al. 2024, 2023] satisfied machine-knitting constraints, they often required manual insertion and pairing of singularities or the solving of large mixed-integer problems. [Mitra et al. 2025] introduced a fully automatic pipeline for placing knit singularities based on a greedy, iterative approach that replaces manual intervention and pairing. Our work extends these frameworks by quantizing the curl of $\overline{\nabla h}$ for more principled knit singularity placement. A Lloyd-type algorithm is used, alternating semi-discrete optimal transport solves with weighted geodesic barycenter updates. The joint optimization leads to better computational scaling behavior and mass preservation of the curl measure.

Lastly, we note some connections to more general quad-dominant meshing methods, e.g., that of Instant Meshes [Jakob et al. 2015] (whose terminology we borrow below) and [Gao et al. 2017; Ray et al. 2006]. Our knit graph generation procedure may be viewed as generating a quad-dominant mesh with only "positional singularities," which correspond to the knit singularities (and stripe bifurcations) of our approach. The works of [Wu et al. 2018, 2019] leveraged such methods, but did not focus on producing machine-knittable knit graphs, allowing for the appearance of "rotational singularities" that resulted in mismatched course/wale directionalities. Furthermore, no specific care was given to a helix-free constraint (§3.1.3) in the meshing process. This constraint can be viewed as requiring a partial order on strips of quads, crucial for generating a valid yarnwise ordering on the loops of the knit structure, e.g., as done by the tracing step of the Autoknit pipeline [Narayanan et al. 2018]. Finally, using the curl measure to place such positional singularities can be viewed in analogy with the heuristic for placement of rotational singularities at places of significant Gaussian curvature in quad/quad-dominant meshes [Bommes et al. 2013].

3 Background

We briefly summarize the basics of computational knitting (§3.1), stripe pattern optimization for knit design (§3.2), and measure quantization in Euclidean space via optimal transport (OT) tools (3.3).

3.1 Knitting and Machine Knitting

We focus on the whole-garment computational knitting context. Details beyond those below may be found in, e.g., [Narayanan et al. 2018; Underwood 2009], for the interested reader. Also recommended are the following talk [McCann 2017] and summer school course [Lin et al. 2025] for summary overviews of relevant concepts.

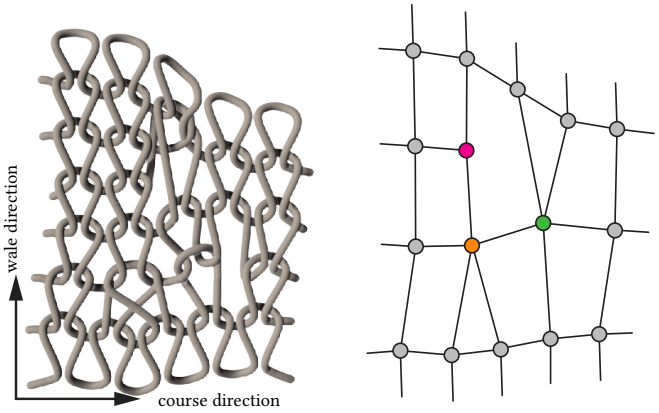


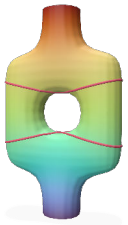
Fig. 2. A local knit structure with three knit singularities (left) and knit graph (right) representations. A decrease (orange), an increase (green), and the right end of a short-row end (pink) are shown. Modified from Mitra et al. [2025] with permission from the authors.

3.1.1 Basic knit structure. In this work, we restrict to the simple, foundational *single jersey* knit pattern, illustrated schematically in Fig. 2. This basic pattern is a regular square lattice grid of interlocking yarn loops, forming rows (*courses*) and columns (*wales*) of knit stitches. Irregularities in the form of *short row ends* and *increases/decreases* allow for starting and ending courses and wales, respectively. We call such irregularities, *knit singularities*, and they are necessary to induce geometric shaping for the underlying fabric.

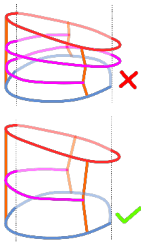
A *knit graph* can be used to represent a knit structure discretely. Following the Autoknit [Narayanan et al. 2018] convention, each knit graph node corresponds to two stacked knit stitches in the vertical direction, and one stitch width in the course direction. Stitches adjacent in the course direction are joined by *course edges* and stitch pairs interlaced with adjacent stitch pairs in the wale direction are joined by *wale edges*. If a node is lacking a course edge on either side, it denotes a *short row end*. If a node has multiple wale edges coming out/in (w.r.t wale direction), it denotes the beginning/terminal end of a wale column via an *increase/decrease*.

3.1.2 Whole-garment machine knitting. To computationally knit a manifold surface M in a *whole-garment* fashion, one uses a V-bed knitting machine, which produces the fabric one course row at a time and avoids any sewing post-processing to join boundaries of the machine output. Such a machine can produce two kinds of basic topological primitives, *tubes* (topological cylinders) and *sheets* (topological disks). As we are using the Autoknit compiler [Narayanan et al. 2018], we restrict to manifolds M that consist purely of tubes that may merge or split, with ∂M (boundary of M) consisting of a union of starting and ending course rows (see example in inset below). With geometric shaping incorporated into the tubes, such a framework can be used to produce a topologically and geometrically diverse array of 3D surfaces. Note that our knit singularity placement framework is not restricted to tubes and may easily be used on sheet primitives, as seen in Fig. 11.

A *time function* $h : M \to [0, 1]$ is usually used to model the course and wale alignment of a whole-garment knit (also shown on inset example). Course rows are meant to roughly align with level sets of h and to be produced in order of increasing h value. Wale columns are meant to be orthogonal to the h level sets. Critical level sets of h (red curves), in the Morse-theoretic sense, partition M into tubes, resulting in a cylindrical decomposition. Upon discretization of M, we take these level sets to be edge cycles μ_i for $i = 1, \ldots, n_{\text{Morse}}$, the construction of which is detailed in Supp. §4 of [Mitra et al. 2024]. In our work, we produce h in the typical fashion: by harmonically interpolating Dirichlet boundary conditions of 0 and 1 on user-specified boundary components.



3.1.3 Helix-free constraints. The candidate knit graph we obtain is passed to the Autoknit compiler [Narayanan et al. 2018], which has several requirements on the knit graph structure. The most important of these is the *helix-free* constraint, illustrated in the inset, which asks that there not be a sequence of wale edges joining any two nodes in the same course row. Wale edges encode an ordering on courses, and such a sequence would entail trying to create certain stitches of a course row before having any prior stitches to interlace them with. There are further constraints which are mostly for robustness of the machine knitting, and we refer the reader to [Narayanan et al. 2018] for a further detailing of these. These further constraints could all be violated when hand-knitting with a single yarn, for example, while the helix-free constraint remains a strict structural constraint.



3.2 Stripes for Knitting

Our method builds upon the prior works of [Knöppel et al. 2015; Mitra et al. 2025, 2024, 2023]. We refer to these works often for further details, but attempt to summarize the necessary basics below. Additionally, it utilizes discrete differential forms and concepts from discrete exterior calculus. We refer readers to [Crane et al. 2013a] for reference, if needed.

3.2.1 Spinning-Form-Based Stripes. To produce a uniform knit structure, one aims for evenly-spaced course rows and wale columns, achieved in a curved setting with knit singularities. This even spacing mirrors the desire for evenly-spaced stripe patterns. This was leveraged in [Mitra et al. 2023] and in [Nader et al. 2021], both of which aim to come up with two roughly orthogonal stripe patterns that represent a candidate knit structure. A key distinction is that the former explicitly incorporates the helix-free constraint of §3.1.3.

Both works used the framework of [Knöppel et al. 2015], whereby a stripe pattern is considered as a function $\varphi : M \to \mathbb{S}^1$ to the unit circle (technically with isolated points of non-definition at singularities). For the stripe pattern depicted in Fig. 3, the surface is colored gold when $\arg \varphi \in [0, \pi)$, and teal when $\arg \varphi \in [-\pi, 0)$. On a triangle mesh $M = (V, E, F)$, a discrete version of such a function can be expressed as a discrete differential 1-form $\sigma : E \to \mathbb{R}$ such that $(d_1\sigma)_t = 2\pi k_t$ for some integer $k_t \in \mathbb{Z}$ (called the *stripe index*)

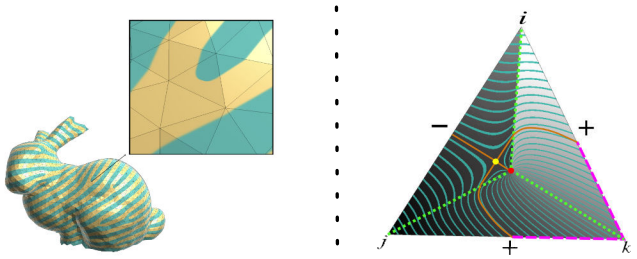
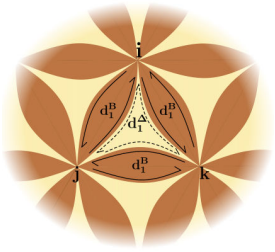


Fig. 3. Left: Stripes on a bunny model. The zoom-in shows behavior of the [Knöppel et al. 2015] interpolant on and around a singular face of stripe index +1, where a new stripe emerges. Right: Foliation behavior of the interpolant on the singular face, showing source (red) and saddle (yellow) singularities (of the vector field and foliation). Level sets of φ emanate to the right, with separatrices labelled in orange (from [Mitra et al. 2024] with permission from the authors).

on each triangle $t \in F$. This form can then be integrated along a spanning tree of edges to form a discontinuous arg φ function. The integer d_1 conditions ensure that this will represent a continuous φ function.

Faces for which $k_t \neq 0$ correspond to *stripe singularities*, where new stripes start or terminate. These singularities correspond to the knit singularities noted in §3.1.1 and are analogous to the "positional singularities" of Instant Meshes [Jakob et al. 2015] (a more general quad-dominant meshing pipeline). The form σ is referred to as the *spinning form* of the stripe pattern as it denotes the "spinning" (change in phase angle) of the unit circle function φ over each edge. In [Knöppel et al. 2015], a (discontinuous) rational stripe interpolant was defined (illustrated in Fig. 3, left) for producing a stripe pattern on these singular triangles. On nonsingular triangles, for which $k_t = 0$, the pattern is piecewise linear and has a well-defined gradient, which we denote $(\nabla\sigma)_t$. This may also be seen as the discretized Whitney interpolant evaluated at the triangle barycenter, as noted in [Mitra et al. 2025].

In [Mitra et al. 2025], the authors introduced a *lens complex* [Soliman et al. 2021] generalization of the spinning form, which allowed for stripe singularities to reside on edges. This extends the set of faces $\tilde{F} = F \sqcup B$ to include bigons B that correspond to edges of the original mesh. It also 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 simplifies knit graph extraction by allowing 1-forms to disagree on twin half-edges of the original triangle mesh when stripe singularities are placed on these bigons. The resulting stripe pattern is piecewise linear on all triangular faces, and example behavior near a singular edge may be seen in Fig. 8. The first exterior derivative d_1 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.



3.2.2 Form Optimization and Constraints. Given a time function h that roughly specifies course row alignment, one may optimize for a spinning form that achieves this. To guide such a form (and thus pattern), we need the normalized time function gradient $\overline{\nabla h} := \frac{\nabla h}{\|\nabla h\|}$, defined per face t, and desired stripe period P. Furthermore, let us assume that singularity placement on edge bigons has been determined and is given by a vector $\mathbf{k}_B \in \{-1, 0, +1\}^{|E|}$. The following quadratic optimization with linear constraints results:

$$\min_{\sigma: E \to \mathbb{R}} \quad \mathcal{F}(\sigma) := \sum_{t \in F} A_t \left\| (\nabla\sigma)_t - \left(\frac{\overline{\nabla h}}{P}\right)_t \right\|^2 \tag{1a}$$

$$\text{such that} \quad d_1^{\Delta}\sigma = 0 \tag{1b}$$

$$d_1^{B}\sigma = P\mathbf{k}_B \tag{1c}$$

$$\sigma|_{\partial M} = 0 \tag{1d}$$

$$\int_{\gamma_i} \sigma = 0 \qquad i = 1, \ldots, n_{\text{pairs}} \tag{1e}$$

$$\int_{\mu_i} \sigma = 0 \qquad i = 1, \ldots, n_{\text{Morse}} \tag{1f}$$

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

In $\mathcal{F}(\sigma)$, the weighting A_t is the area of triangle t. Within the constraints: Eqs. (1b), (1c) enforce stripe singularity placement on the specified bigons (and not on any faces); Eqs. (1d) impose boundary alignment of stripes; Eqs. (1e) ensure non-helicing as described below in §3.2.3; Eqs. (1f) ensure that the Morse-theoretical cylindrical decomposition is respected; Eqs. (1g) ensure a well-defined global function φ by fixing integer homology generator path integrals as given by a vector $\mathbf{k}_g \in \mathbb{Z}^{2g+n-1}$ (described further in §4.1 of [Mitra et al. 2024] and §4.3.2 of [Mitra et al. 2025]).

The path integral constraints above are oriented sums of σ along edge chains in the mesh M. Such oriented edge chains are homologous and equivalent to integrals along polylines over faces, as detailed in Lemma 1 of [Mitra et al. 2023]. These path integral constraints quantify the amount of (oriented) crossing of level sets of φ across the edge chain γ. In most instances, when $\int_\gamma \sigma = 0$, this enforces that following a level set of φ starting at $\gamma(0)$ will pass through $\gamma(1)$. This is used to route separatrices between paired singularities via constraints (3), as described in §3.2.3 below. An exception to this expected behavior may be seen in Fig. 10 which necessitates our novel singularity ordering constraints.

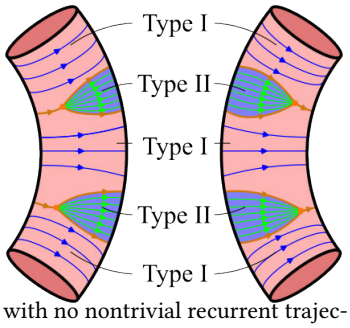
The analogous optimization for the wale columns has $\overline{\nabla h}$ replaced with $\overline{\nabla h}^{\perp}$ ($\overline{\nabla h}$ rotated by 90° counterclockwise per face) in Eq. (1a), and the dropping of Eqs. (1d), (1e), and (1f). Going forward, we will use σ_c and σ_w to denote the 1-forms for the course/wale patterns, and φ_c and φ_w to denote the integrated $\mathbb{S}^1$ functions, respectively. If an equation or statement applies to both patterns, σ or φ will be used.

Lastly, we note a final feasibility condition that must hold via the global index theorem (Thm 3.1) of [Noma et al. 2022], a consequence of Stokes' Theorem. Specialized to our lens complex setting, this says that for any discrete 1-form σ, we have the following identity:

$$\sum_{e \in E} (d_1^{B}\sigma)_e + \sum_{t \in F} (d_1^{\Delta}\sigma)_t + \int_{\partial M} \sigma = 0. \tag{2}$$

In particular, for Problem (1) above, for σ_c, the only non-zero term is the first, which tells us that the entries of $\mathbf{k_B}$ sum to 0. For the wale columns, in optimizing for σ_w, only the middle term disappears, and we note that the last term consists of multiple subterms, one corresponding to a path integral over each boundary component. These must be rounded to integer multiples of the period as detailed in §4.3.2 of [Mitra et al. 2025].

3.2.3 Orbit Complex & Foliation Behavior. The level sets of φ describe the trajectories of a vector field flow, and thus a *foliation*, as described in [Mitra et al. 2024]. On non-singular triangles t, this vector field is the rotated $(\nabla\sigma)_t^{-\perp}$ (90° clockwise), and on singular triangles, it is the rotated gradient of the rational interpolant (illustrated in Fig. 3). For a restricted class of vector fields, i.e., those with no nontrivial recurrent trajectories, e.g., limit cycles, the topological behavior of the vector field flow is fully characterized by the *orbit complex* (see §2.2.1 of [Aranson et al. 1996]). In our setting, this is a partitioning of M into *cells* of two types which have a particular topology and flow behavior. The pedagogical inset above illustrates the orbit complex for the course stripe pattern on a simple bent cylinder (both sides shown and set back-to-back). Flows on Type I cells are equivalent to periodic flow around an annulus (cylinder/tube) and those on Type II cells are equivalent to a horizontal flow on an infinite strip $\mathbb{R} \times (0, 1) \subset \mathbb{R}^2$. There are three Type I cells, holding blue trajectories, and two Type II Cells, holding green trajectories in the inset. Cells are separated from each other by *separatrices* of the flow, trajectories that limit to or from vector field singularities p and have nearby trajectories not converging to p. These are illustrated by the (darker) orange trajectories above. As typified in this example, Type II cells arise and end at stripe singularities (orange nodes).



In [Mitra et al. 2024], the authors show that appropriate control of the orbit complex can guarantee that all course trajectories (level sets of φ_c) will be helix-free. In particular, they showed that if all the separatrices bounding Type II cells are helix-free, then all course trajectories will be helix-free (Prop. 1 of [Mitra et al. 2024]).

3.2.4 Singularity Pairing Assignment. Practically, the non-helicing of bounding separatrices is achieved via the pairing of index +1 stripe singularities with index -1 stripe singularities and application of the constraints (3), to route these bounding separatrices. The number s of such positive/negative singularities for σ_c is equal by the global index theorem Eq. (2), and thus the cost of pairing is captured in a cost matrix $C \in (\mathbb{R}^+)^{s \times s}$. For positive singularity i and negative singularity j, the cost $C_{ij} = d_{ij}^2$, where d_{ij} is the cost of a Dijkstra shortest path between singular points on a custom weighted graph of mesh halfedges. In particular, if $\overline{\nabla h}^{-\perp}$ is the normalized clockwise (90°) rotated gradients of the time function then the weight of each halfedge e_{ij} is given by $\frac{1-\widehat{e_{ij}} \cdot \overline{\nabla h}^{-\perp}}{2}$, where $\widehat{e_{ij}}$ is the normalized halfedge vector. This encourages paths flowing from

positive singularities to negative singularities in the appropriate direction as described in §4.2 of Mitra et al. [2024].

The optimal assignment is encoded 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. This is found via a relaxation to a discrete optimal transport problem (see §2.3 of [Peyré and Cuturi 2019]) between uniform discrete measures on the set of +1 singularities and -1 singularities:

$$\min_{\mathbf{T}\in[0,1]^{s\times s}} \quad \langle \mathbf{C}, \mathbf{T}\rangle \tag{3a}$$

$$\text{such that} \quad \mathbf{T}\mathbb{1} = \mathbb{1} \tag{3b}$$

$$\mathbb{1}^T\mathbf{T} = \mathbb{1}^T \tag{3c}$$

Eq. (3a) denotes a sum of the entrywise products and reflects the total cost of a transport pairing, and Eqs. (3b), (3c) ensure that the transport plan $\mathbf{T}$ marginalizes appropriately to the measure on +1 singularities and the measure on -1 singularities. The basic theory of the Kantorovich relaxation ensures that any minimizer $\mathbf{T}$ is a permutation matrix (Prop. 2.1 of [Peyré and Cuturi 2019]).

3.2.5 Singularity placement via curl heuristic. The problem of evenly spaced stripes, and thus evenly spaced course rows and wale columns in a resultant knit graph, is reliant on an intelligent placement of stripe singularities. In [Mitra et al. 2025], an (ill-posed) continuous optimization problem was proposed. In this problem, one aims to find an optimal vector field V that roughly matches $(\overline{\nabla h}/P)$ and has quantized curl equal to the sum of Dirac delta measures (of mass P) placed at singular points $p_1, \ldots, p_n \in M$, scaled by ±1 indices $k_1, \ldots, k_n \in \{-1, +1\}$.

$$\min_{V,\, p_1,\ldots,p_n} \quad \int_M \left\| V - \frac{\overline{\nabla h}}{P} \right\|^2 \tag{4a}$$

$$\text{such that} \quad \nabla \times V = P \sum_{i=1}^{n} k_i \delta(p_i) \tag{4b}$$

This problem may be viewed as a continuous version of Problem (1) with the addition of the integer variables in $\mathbf{k}_B$. These additional variables would reflect the freedom to determine and shift singularity placements. Rather than solve the analogous large mixed-integer problem as was done in [Mitra et al. 2023], the authors of [Mitra et al. 2025] proposed the use of a curl heuristic for determining singularity positions, and greedily placed singularities at high curl $(\nabla \times \overline{\nabla h})$ in a bid to reduce the objective $\mathcal{F}(\sigma)$ iteratively. Fig. 4 demonstrates that this curl serves as an intuitive indicator for knit singularity placement, highlighting regions where level sets of h diverge quickly and even spacing could be preserved by stripe singularity insertion. On the bent cylinder model, this naturally places short-row ends along the "spine" where curl is most pronounced.

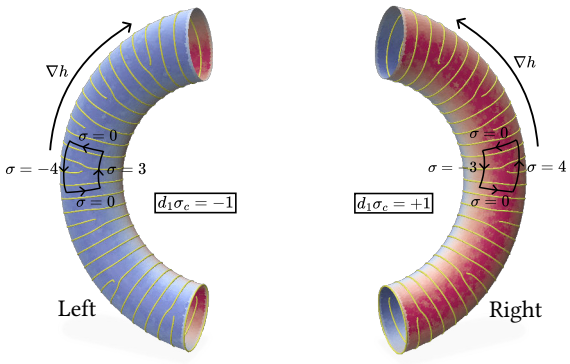
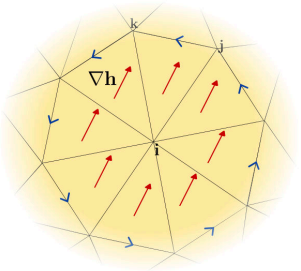


Fig. 4. $\nabla \times \overline{\nabla h}$ informs the placement of short row ends. Curl is the limit of integrals around infinitesimal loops. Integrals over portions closer to the "spine" of the bent cylinder have larger curl values. In the wale direction, curl serves as a good signal to place increases/decreases. Modified from [Mitra et al. 2025] with permission from the authors.



In [Mitra et al. 2025], $\nabla \times \overline{\nabla h}$ was discretized to vertices as the per-vertex curl of a face-based vector field via the standard formula below [de Goes et al. 2016; Wardetzky 2007]. The inset (borrowed from [Mitra et al. 2025] with permission from the authors) illustrates the 1-ring N_1 of vertex v_i, with a per-face ∇h field shown in red. The blue arrows denote the orientation of the vectors $\mathbf{p}_{jk}$ in the sum.

$$[\nabla \times \overline{\nabla h}]_i = \frac{1}{A_{v_i}} \sum_{ijk \in N_1(v_i)} (\overline{\nabla h})_{ijk} \cdot \mathbf{p}_{jk} \tag{5}$$

The equation integrates $\overline{\nabla h}$ over the boundary of $N_1(v_i)$, and A_{v_i} denotes a third of the area of $N_1(v_i)$ (the typical dual barycentric cell).

3.3 Measure Quantization via Semi-Discrete OT in $\mathbb{R}^n$

For absolutely continuous probability measures μ on $\mathbb{R}^n$, i.e., given by a density function $\rho : \mathbb{R}^n \to \mathbb{R}$ such that $\mu(X) = \int_X \rho\, dx$, there is a well-studied optimal-transport-based (OT-based) framework for discrete approximation of such measures. For further details we refer interested readers to [Peyré and Cuturi 2019; Santambrogio 2015].

3.3.1 Optimal Transport. The *Wasserstein distance* $W_2(\mu, \nu)$ between two probability measures $\mu, \nu \in \mathcal{P}(\mathbb{R}^n)$ is given by the minimizer of the following infinite-dimensional linear optimization problem.

$$W_2^2(\mu,\nu) = \min_{\gamma\in\mathcal{P}(\mathbb{R}^n\times\mathbb{R}^n)} \quad \int_{\mathbb{R}^n\times\mathbb{R}^n} \gamma(x,y)\, \|x-y\|^2\, dx\, dy \tag{6a}$$

$$\text{such that} \quad (\pi_x)_{\#}\gamma = \mu \quad \& \quad (\pi_y)_{\#}\gamma = \nu \tag{6b}$$

The optimal product measure γ expresses the minimum cost transport plan between μ and ν, with Eqs. (6b) expressing the fact that γ marginalizes onto μ and ν via the projection maps onto each factor of $\mathbb{R}^n$. Intuitively, this distance is often called the "Earth Movers' Distance", as Eq. (6a) reflects the cost to transform μ as a pile of dirt into ν as a pile of dirt, with cost $\|x-y\|^2$ associated with moving mass.

This primal form of the optimal transport problem is dual to the following linear optimization problem over functions φ, ψ on $\mathbb{R}^n$:

$$\max_{\varphi,\psi \in L^1(\mathbb{R}^n)} \quad \int_{\mathbb{R}^n} \varphi \, d\mu + \int_{\mathbb{R}^n} \psi \, d\nu \tag{7a}$$

$$\text{such that} \quad \varphi(x) + \psi(y) \leq \|x-y\|^2 \tag{7b}$$

The functions $\varphi(x), \psi(y)$ have economic interpretations that roughly correspond to the price of picking up and dropping off mass at points x and y respectively, as proposed by a third-party aiming to perform the transport for a price. Eqs. (7b) express the fact that for any two pickup and drop-off points, the sum of these prices should not exceed the cost of transport. This problem may also be expressed in a purely unconstrained way in terms of ψ alone by replacing φ with its *c-conjugate*.

$$\psi^c(x) \coloneqq \inf_{y \in \mathbb{R}^n} \|x-y\|^2 - \psi(y) \tag{8}$$

3.3.2 Semi-discrete optimal transport and power diagrams. In the semi-discrete setting, the source measure μ is absolutely continuous and the target measure ν is *discrete*, meaning it is uniformly supported on a set of N points $p_1, \ldots, p_N \in \mathbb{R}^n$: $\nu = (1/N)\sum_{i=1}^N \delta(p_i)$. We will see that the optimal transport plan then associates regions of $\mathbb{R}^n$ to individual points p_i. The dual OT problem (7) simplifies to a finite-dimensional concave maximization problem in weights $\psi_i = \psi(x_i)$ (where the c-conjugate has been substituted in).

$$\max_{\psi_1,\ldots,\psi_N} \quad \mathcal{G}(\vec{\psi}) \coloneqq \int_{\mathbb{R}^n} \min_{i=1,\ldots,N} \left(\|x-p_i\|^2 - \psi_i \right) d\mu(x) + \frac{1}{N}\sum_{i=1}^N \psi_i \tag{9}$$

These weights have a concrete computational geometry interpretation as *power weights* in a *power diagram* [Aurenhammer 1987]. Three examples are shown in right three panes of Fig. 5. These are generalizations of Voronoi diagrams, where each *power cell*, corresponds to regions that have lowest *power distance* $\|x-x_i\|^2 - \psi_i$. Examples may be seen in Fig. 1, 16.

$$V_i(\vec{\psi}) \coloneqq \{x \in \mathbb{R}^n \mid \|x-p_i\|^2 - \psi_i \leq \|x-p_j\|^2 - \psi_j \quad \forall j\} \tag{10}$$

The gradients of the objective $\mathcal{G}(\vec{\psi})$ are given by the mass discrepancies:

$$\frac{\partial \mathcal{G}}{\partial \psi_i} = \frac{1}{N} - \int_{V_i(\vec{\psi})} d\mu \tag{11}$$

3.3.3 Iterative Lloyd's algorithms for quantization. An OT-based *quantization* of a continuous measure μ is defined as a uniform discrete measure ν that minimizes the Wasserstein distance $W_2(\mu,\nu)$ by allowing movement of the support sites $p_1, \ldots, p_N$. This is a non-convex optimization problem in both the power weights ψ_i and the site positions p_i. One popular and effective approach to finding a local minimum, is to perform alternating optimization between ψ_i and p_i, keeping the other set of variables fixed in each step. This results in a Lloyd-type algorithm [Lloyd 1982] where one alternates between two steps:

(1) Optimize the power weights ψ_i such that every power cell is adjusted to cover an equal portion of source measure μ. This is done by solving the concave maximization problem in 3.3.2.
(2) Snap sites p_i to the μ-weighted barycenters of their respective power cells. Proof of optimality may be found in [Claici et al. 2018; de Goes et al. 2012], and is skipped for brevity.

An example of this algorithm in action on a simple Gaussian distribution may be seen in Fig. 5. While no guarantees on global optimality are had, the resulting local minima are usually quite similar to each other, signifying robustness to initialization and likely near-global optimality.

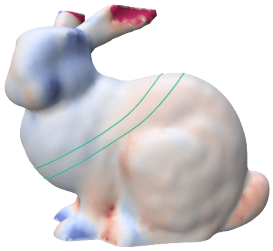
4 Method

4.1 Signed Measure Quantization

The constructions of §3.3 easily generalize to positive measures of equal total mass $\mu(\mathbb{R}^n) = \nu(\mathbb{R}^n)$ (i.e., this value may not be 1). Below, we use μ_c to denote the signed curl measure $\nabla \times \overline{\nabla h}$ of the normalized time gradient function, and μ_w to denote the signed curl measure $\nabla \times \overline{\nabla h}^{\perp}$ relevant to wale stripes. μ will be used if the discussion pertains to both course and wale curl measures.

As either of these measures is signed (typically), we separately quantize the positive and negative parts of the signed measure: $\mu = \mu^+ - \mu^-$. The discretized result is two positive measures supported on disjoint vertex sets of M. Note that this notion of quantization does not allow for any mass cancellation, as is considered in some notions of unbalanced transport, e.g., [Chizat et al. 2018]. Our approach is simple, effective, and allows us to adapt the machinery of the positive measure case in a straightforward fashion. Empirically, the curl measure varies smoothly, and does not often contain connected regions of positive and negative curl with mass of sub-period order.

4.1.1 Balanced Pairs for the Course Direction. Given the matching problem (3), one might ask if it is possible for μ_c^+ and μ_c^- to have the majority of their mass on different level sets of h. Such a distribution would complicate the matching problem, and make it quite challenging to produce helix-free course stripes aligned with level sets of h. Empirically, this does not seem to be the case, as observed in the inset, where the mass of μ_c^+ and μ_c^- seem to be equal between any two level sets of h. Indeed, we observe:



Proposition 1. *Consider a smooth time function $h : M \to [0,1]$ achieving 0 and 1 on connected components of ∂M, and $\mu_c \coloneqq \nabla \times \overline{\nabla h}$, the mass of μ_c^+ and μ_c^- is balanced between any two level sets of h. In particular, let $M_{ab} \coloneqq \{p \in M \mid a \leq h(p) \leq b\}$ for any $0 \leq a, b \leq 1$.*

$$\int_{M_{ab}} d\mu_c^+ = \int_{M_{ab}} d\mu_c^- \tag{12}$$

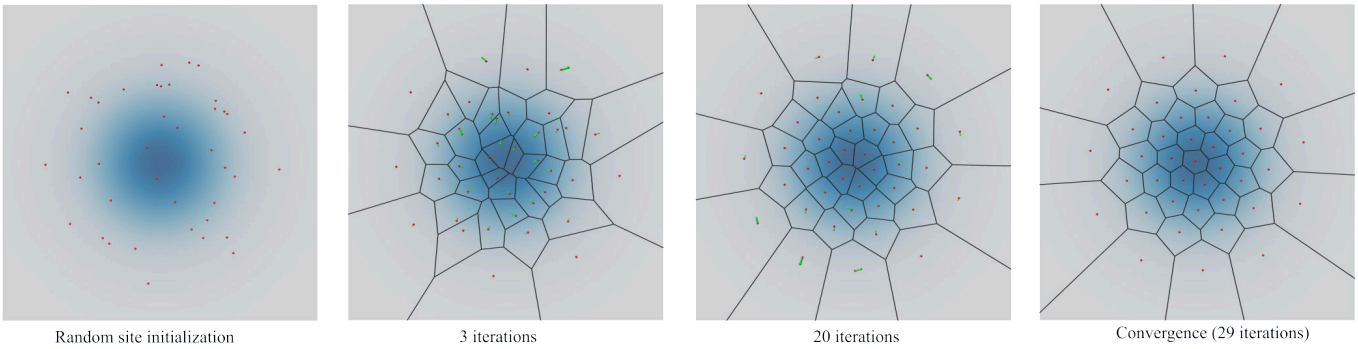


Fig. 5. We apply our method to a Gaussian measure in the plane. We show the power cells along with sites (red) and the next update step (green) at different stages of the Lloyd algorithm. A single iteration consists of a weight optimization (semi-discrete OT solve) and a power cell barycenter step. Note the quick qualitative convergence, with the basic structure already approximately correct after just 3 iterations.

Proof. Letting $\omega_c \in \Omega^1(M)$ denote the 1-form dual to $\overline{\nabla h}$, we have that $\mu_c = d_1\omega_c$. Then by Stokes' Theorem:

$$\int_{M_{ab}} d\mu = \int_{M_{ab}} d_1\omega = \int_{\partial M_{ab}} \omega = 0. \tag{13}$$

The last integral vanishes by the fact that $\overline{\nabla h}$ is orthogonal to the level sets of value a and b. As $\mu_c = \mu_c^+ - \mu_c^-$, the mass equality results. □

4.2 OT-Based Quantization on Surfaces

We first find the number of desired positive/negative support sites by rounding the total integrated positive/negative curl over the domain (after dividing by the striping period).

$$N^+ = \left\lfloor \frac{\int_M \mu^+}{P} \right\rceil \quad , \quad N^- = \left\lfloor \frac{\int_M \mu^-}{P} \right\rceil \tag{14}$$

To find the positions of these singularities, we generalize the machinery described in §3.3 to quantize the signed curl measure over the domain. In particular, for each of μ^+ and μ^-, we search for uniform discrete measures $\nu^{+/-}$ supported on points $p_1^{+/-}, \ldots, p_{N^{+/-}}^{+/-}$ that minimize the Wassertein distance to $\mu^{+/-}$. For simplicity below, we drop the superscripts, as the procedure is identical for each part of the measure.

4.2.1 Geodesic Tools. In order to generalize the story in an efficient manner, we need a quick way to approximate the geodesic (and power) distance on mesh M, to compute mass-weighted barycenters of power cells on surfaces, and to step to these barycenters along M. The heat diffusion-based tools of [Crane et al. 2013b; Sharp et al. 2019c] are crucial for achieving these aims. In particular, [Crane et al. 2013b] introduces the heat kernel $k_t(p, x)$ on a mesh M, which measures the heat transferred from a source $p \in M$ to a target $x \in M$ after time t. A simulated heat diffusion step can be calculated via a backward Euler step, resulting in a sparse positive-definite linear system that only depends on M and its discretized Laplacian L. It can be pre-factored with a sparse Cholesky method for efficient repeated evaluations. This heat kernel is related to geodesic distance $d(p, x)$ via Varadhan's formula, which implies the following proportionality for small times t:

$$k_t(p, x) \propto e^{-d^2(p,x)/4t^2} \tag{15}$$

The work of [Sharp et al. 2019c] generalizes this story to vector bundles, and the tangent bundle TM in particular, replacing L with a connection Laplacian L^∇, and using an associated vector heat equation. This vector heat diffusion allows one to utilize similar numerical tools to parallel transport and diffuse tangent vectors at source points p to target points x. Diffused radial and "horizontal" vector fields are computed and combined to determine a *logarithmic map* $\log_p : M \to T_pM \cong \mathbb{R}^2$ which maps M onto the tangent plane T_pM at p. This discrete logarithmic map is used to approximate *Karcher means* (weighted means on surfaces) by pulling back measures η on M onto T_pM via $(\log_p)_\# \eta$ and performing weighted averaging on $T_pM \cong \mathbb{R}^2$, which is straightforward. $\log_p$ is the inverse of an *exponential map* $\exp_p : T_pM \cong \mathbb{R}^2 \to M$ which maps a tangent vector $v \in T_pM$ to the point that would be reached by walking along a geodesic in the direction of v for distance $\|v\|$. This is calculated by just traversing the surface intrinsically. We utilize the Geometry Central [Sharp et al. 2019a] implementations of these for triangular meshes, and use the default diffusion time value t that they recommend, the square of the average mesh edge length.

4.2.2 Geodesic Power Diagrams & Lloyd Steps. In [Sharp et al. 2019c], the authors construct geodesic Voronoi diagrams, by representing Voronoi cells in a "fuzzy" manner, via measures that form a partition of unity and are concentrated on their respective cells. We do the same, but with a set of "power distance kernels":

$$V_i(\vec{\psi})(x) = \frac{e^{-\psi_i/4t^2} k_t(p_i, x)}{\sum_j e^{-\psi_j/4t^2} k_t(p_j, x)} \tag{16}$$

By (15), we have that $e^{-\psi_i/4t^2} k_t(p_i, x) \propto e^{-(d^2(p_i,x)-\psi_i)/4t^2}$. As in the geodesic Voronoi case, this measure is near 1 on the geodesic power cell, and quickly drops to 0 when it approaches points that are of lower power distance. Since a uniform constant shift of the ψ_i result in the same kernels, we stabilize the computation by subtracting $\max_j(\psi_j)$ off of each ψ_i to prevent numerical underflow.

This approximation of the power cell allows us to optimize for the power weights ψ_i using the following generalized (from Eq. (11))

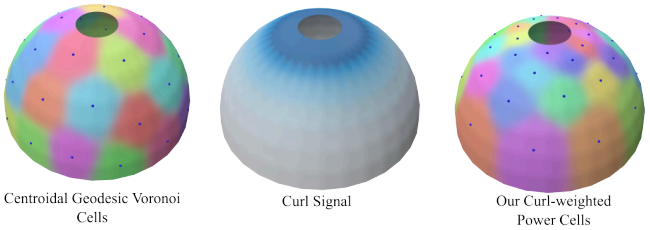


Fig. 6. A comparison on a hemisphere mesh of a geodesic power diagram and a geodesic centroidal Voronoi diagram. The power diagram is for the final configuration of our quantization method applied to μ_w^- on this model.

gradient step:

$$\frac{\partial \mathcal{G}}{\partial \psi_i} = P - \int_M V_i(\vec{\psi})\, d\mu \tag{17}$$

These gradients are used in an L-BFGS optimization scheme that solves for the weights ψ_i. We use the implementation [Qiu and Toewe 2019], with default parameters memory $m = 6$ for Hessian approximation, and $\varepsilon_{\text{L-BFGS}} = 10^{-5}$ as the threshold gradient norm for convergence.

For the Lloyd steps, where we update point positions to the weighted means of their corresponding power cells, we pullback the calculation to the tangent plane and push it through the exponential map.

$$p_i \mapsto \exp_{p_i}\left(\int_{\log_{p_i}(M)} \vec{x}\; d(\log_{p_i})_{\#} V_i(\vec{\psi})\right) \tag{18}$$

As in the proposed Lloyd-type method for geodesic centroidal Voronoi diagrams in [Sharp et al. 2019c], this only corresponds to a single step of the Karcher mean algorithm, but we find that it is faster to forgo multiple steps of this. A comparison of a geodesic centroidal Voronoi diagram and a power diagram resulting from quantization of μ_w^- on a simple hemispherical model is shown in Fig. 6. Additionally, we show an example of path trajectories on a sock model in Fig. 7.

In actuality, in the Lloyd step (18), we use a reduced update step size of 0.8 to discourage oscillatory behavior during optimization. We loop over alternating steps of L-BFGS optimization for the weights, and the above Lloyd steps until the relative improvement in the semidiscrete OT objective $\mathcal{G}(\vec{\psi}, p_i)$ drops below $\varepsilon_{\text{Lloyd}} = 10^{-4}$. Pseudocode for our quantization procedure is in Algorithm (1). Our method shares the same general robustness to initialization as the planar method, i.e., while the full objective (as a function of point positions and weights) is non-convex, the resulting singularity positions are qualitatively similar and have similar objective values. A plot of the OT objective versus iteration count is shown in Fig. 15.

This optimization is done once per signed measure. In the course direction, we apply this algorithm per cylindrical component to prevent singularity pairing across components. In the wale direction, the algorithm is carried out over the entire mesh.

Fig. 7. Point trajectories on a sock model, starting from a random initialization. We quantize the positive portion of $\nabla \times \overline{\nabla h}$ (left) with a sparse number of singularities (12).

Algorithm 1: Lloyd's algorithm for curl quantization

Input: Surface M; curl measure μ on M; number of sites n; tolerance $\varepsilon_{\text{Lloyd}} = 10^{-4}$; diffusion parameter t
Output: Site positions $\{p_i\}_{i=1}^n$
Initialize sites and weights;
Sample n sites $\{p_i\}$ uniformly at random on M;
Initialize $\vec{\psi} \leftarrow \mathbf{0}$;
$\mathcal{G}_{\text{prev}} \leftarrow +\infty$;
for $k = 1$ **to** $K_{\max}$ **do**
  (1) Heat-kernel evaluation;
  for $i = 1$ **to** n **do**
    Compute $k_t(p_i, x)$ for $x \in M$
  (2) Compute Power Cells;
  for $i = 1$ **to** n **do**

$$V_i(\vec{\psi})(x) \leftarrow \frac{e^{-\psi_i/4t^2} k_t(p_i, x)}{\sum_j e^{-\psi_j/4t^2} k_t(p_j, x)}$$

  (3) Optimize weights via L-BFGS;

$$\frac{\partial \mathcal{G}}{\partial \psi_i} = P - \int_M V_i(\vec{\psi})\, d\mu$$

$$\vec{\psi}, \mathcal{G}_{\text{curr}}(\vec{\psi}) \leftarrow LBFGS\left(\frac{\partial \mathcal{G}}{\partial \psi_i}\right)$$

  (4) Lloyd update;
  for $i = 1$ **to** n **do**
    Compute $\log_{p_i}(x) \in T_{p_i}M$ for $x \in M$
    Centroid Update;

$$\bar{v}_i \leftarrow \int_{\log_{p_i}(M)} \vec{x}\; d(\log_{p_i})_{\#} V_i(\vec{\psi}) \in T_p M$$

    Exponential map update;

$$p_i \leftarrow \exp_{p_i}(0.8\, \bar{v}_i) \in M$$

  (5) Check stopping criterion;
  $\delta \leftarrow \frac{|\mathcal{G}_{\text{prev}} - \mathcal{G}_{\text{curr}}|}{|\mathcal{G}_{\text{prev}}|}$;
  if $\delta < \varepsilon_{\text{Lloyd}}$ **then**
    break
  $\mathcal{G}_{\text{prev}} \leftarrow \mathcal{G}_{\text{curr}}$;
return $\{p_i\}$

4.3 Singularity Discretization & Pairing

At the end of our quantization procedure for the course stripes, we have sets of points on the domain representing positive and

negative singularities. As in Mitra et al. [2024], we must pair these singularities via a slight modification of the optimization problem (3). In particular, we use a modified version of the cost matrix to guide the optimal matching problem.

$$C_{ij} = |\Delta h|^2 + |d_{ij}|^2 \tag{19}$$

Here Δh is the absolute difference between time function values of the singular points, and further stabilizes the matching problem.

In addition to the pairing, we must also map singularity points to discrete mesh edges to form the vector $\mathbf{k_B}$ in Problem (1). For a matched pair $p_i^+, p_{\mathbf{T}(i)}^-$, we compute the average isoline value $h_{\text{avg}} := (h(p_i) + h(p_{\mathbf{T}(i)}^-))/2$. For each member of the pair we find the nearest edge in the mesh that passes through the h_{avg} isoline and is aligned with the time function gradient. An edge is aligned if the edge vector has (absolute) angle with ∇h less than or equal to $30°$, and distance to edges is estimated via the geodesic heat kernel method [Crane et al. 2017]. This alignment ensures the robustness of our path constraints described in Sec. 4.4.

In the wale direction, there is no pairing scheme required. We simply find the most anti-aligned edge with the time function gradient that is closest to the singular point to serve as our edge singularity.

4.4 Discretized Stripe Optimization

After an appropriate pairing of singularities, we must solve an optimization problem for the stripe pattern, akin to Problem (1). However, we must ensure the non-helicing of all level sets, as described in [Mitra et al. 2024], by controlling the routing of separatrices. This is more challenging in the dense stripe setting (higher frequency stripes), which are now computationally within reach. To achieve this behavior in these more challenging settings, we require 3 novel additions/modifications to Problem (1). Note that these constraints are very local in nature and thus mostly independent of the global measure quantization. This follows from Prop. (1) which implies that the quantized positive and negative course singularities should be "balanced" and paired singularities should exist on very similar level sets.

4.4.1 Symmetric Short Row Ends. The first of these is a specification of similar local stripe behavior at each end of a short row pair. If the edges of the short row ends are labelled as below in Fig. 8, then we impose an equality constraint $\sigma_{ij} = -\sigma_{kl}$ and an inequality constraint $\sigma_{ji} < 0$. As $\sigma_{ij}+\sigma_{ji} = P = -(\sigma_{kl}+\sigma_{lk})$, the first constraint imposes a symmetric structure on the short row ends: $\sigma_{lk} = -\sigma_{ji}$. This is crucial for enforcing appropriate orbit complex (and thus helix-free) behavior. If not imposed, joining separatrices with level set constraints may be impossible between paired singular edges.

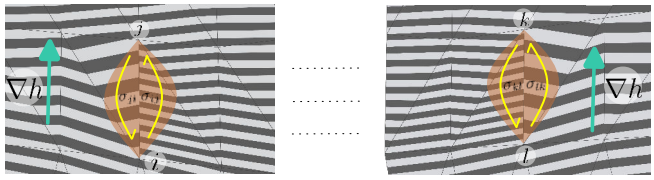
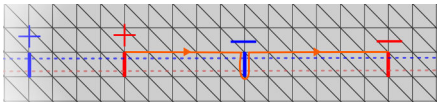


Fig. 8. Notation for the symmetric short row ends conditions described in §4.4.1

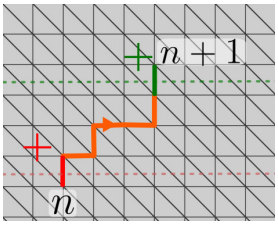
The second constraint strongly motivates the depicted singularity structure by forcing the form σ_c to increase in the direction of ∇h along both half-edges of the bigon. With this constraint and the inherent smoothness arising from an L^2 difference objective, it is very energetically favorable for all nearby edges to have the same behavior. To be more precise, the discrete index of the 1-form, defined as the number of "sign changes" in σ_c as you circle about a face or vertex [Gortler et al. 2006], should be zero on all faces (bigons included) and vertices of the mesh near stripe singularity edges.

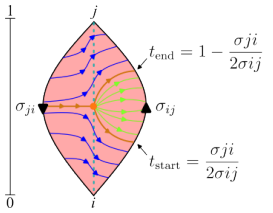
4.4.2 Separatrix Path Routing. The second modification is a routing rule for construction of the pairing edge chains for the constraints of Eq. (1e). When we pair course singularities, they are snapped to distinct edges intersecting their h_{avg} isoline. In dense stripe settings, the h_{avg} isolines for distinct pairs may be quite close to each other, making routing of edge chains potentially challenging. Recall that the choice of edge chain only matters up to homology in $M\backslash K$, where K denotes the singularities (Lemma 1 of [Mitra et al. 2023]). Thus, routing appropriately around other singular edges is the key factor in appropriately specifying the constraints.



In particular, for a given pair and average isoline value h_{avg}, the union of triangles that the h_{avg} level set passes through forms a triangle strip. Starting from the +1 singular edge, we walk along the top edge in the direction of $\overline{\nabla h}^{-\perp}$ until we reach the top of the -1 singular edge. If we encounter a singular edge from a different pair, then we route above or below, according to the relative average time function values as shown in the inset. Here the coloring indicates the matched singularities and the dotted lines are the respective average time function values. Since the blue singularity pair lies above the red, the path constraint connecting the red singularities (in orange) routes below the blue singular edge.

4.4.3 Ordering Constraints. Lastly, we are after an orbit complex where all trajectories born at singular edges form a single Type II cell that terminates at the paired singularity, as shown in the inset of §3.2.3. Furthermore, these cells must be "ordered" according to the time function h. This is achieved by imposing an ordering inequality constraint between each subsequent pair of course row singularities.





To implement these linear inequality constraints in the least restrictive way, we need to note an implicit abstract stripe interpolant on the edge bigon faces. This is illustrated schematically in the inset for a +1 singular edge (with the picture mirrored across the vertical axis for a -1 singularity). For comparison in the singular face setting, one can look at

the interpolant of [Knöppel et al. 2015] illustrated in Fig. 3 (right). The back separatrix enters at the midpoint of the left half-edge and the front separatrices exit at parametric points t_{start} and t_{end} of the right half-edge. The red source and yellow saddle singularities of the [Knöppel et al. 2015] interpolant are merged into a single orange singularity with two hyperbolic sectors and one parabolic sector. The new outgoing trajectories exit the right half edge between parametric points t_{start} and t_{end}.

There is an ordering constraint between each two consecutive +1 course singular edges, represented by a path integral along an edge chain between t_{end} of the lower singularity and t_{start} of the upper singularity. To route such a path appropriately, we start at the upper vertex of the lower singularity and attempt to reach the lower vertex of the upper singularity. First, we greedily move up until the h_{avg} value of the upper singularity is reached. Then we move along the triangle strip of this isoline, routing around other singular edges in the same way specified in §4.4.2 if they are encountered. Lastly, we append at the beginning and end of this edge chain the partial half-edges from t_{end} of the lower singularity and to t_{start} of the upper singularity. If the path between two consecutive course singularities is γ as shown in orange in the inset. Then we impose that $\int_\gamma \sigma_c \geq 0$. Example ordering constraints on the simple bent cylinder model may be seen in Fig. 9.

Intuitively, these constraints can be seen as asking that the stripe function be "non-decreasing" between the bounding separatrices of the Type II cells associated with the singularities. The pre- and post-pended partial chains make sure this constraint is not too restrictive and still operates well in the dense knit setting (when many singularity pairs have similar h_{avg} values). As an example of the necessity of such a constraint, we show a schematic of a helicing Type II cell that may occur when such ordering constraints are not imposed in Fig. 10. Note that this does not invalidate Prop. 1 of [Mitra et al. 2024], but merely provides the perspective that additional constraints (beyond level set constraints between paired singularities) are needed in the dense knit graph setting.

4.5 Knit Graph Extraction

4.5.1 Integration of 1-forms: Having solved for the two 1-forms σ_c and σ_w, the orthogonal stripe patterns are then traced to form the nodes of the knit graph. The forms are first integrated along a spanning tree of mesh edges and the integrated values, α for course and β for wale, are stored modulo P at every mesh corner. As in Mitra et al. [2025], our singularities lie on edges ensuring that stripe level sets are linear over faces. To extract stripe intersections over a face, we solve the linear system,

$$\begin{aligned} \alpha_i b_i + \alpha_j b_j + \alpha_k b_k &= Z_\alpha \\ \beta_i b_i + \beta_j b_j + \beta_k b_k &= Z_\beta \end{aligned} \tag{20}$$

where b_i, b_j and b_k are barycentric coordinates on f_{ijk} and Z_α and Z_β are the stripe level sets passing through f_{ijk}. Similar to [Mitra et al. 2025], this approach does not require the construction of a specialized [Mitra et al. 2024] or non-linear [Knöppel et al. 2015] interpolant.

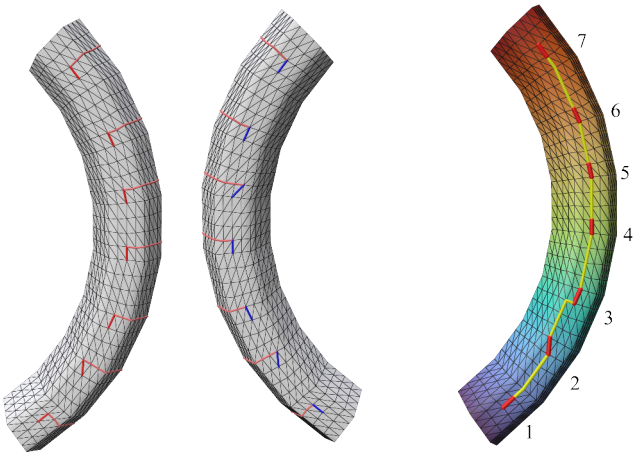


Fig. 9. Left: Singularity pairing constraints (3) positive (red) and negative (blue) edges on a pedagogical bent cylinder model (both sides shown). Right: Ordering constraints (§4.4.3) between the positive singularities (only one side shown). The singularities are ordered according to increasing time value.

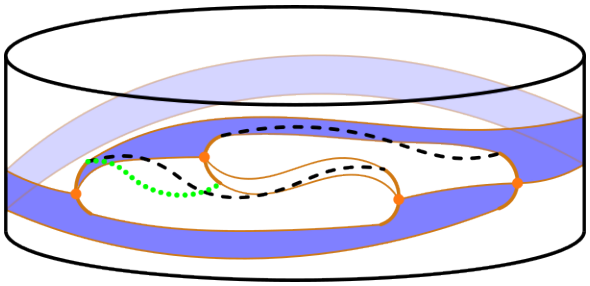
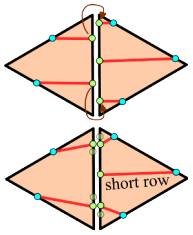


Fig. 10. A schematic showing the necessity of ordering constraints. The interaction of two pairs of course singularities with separatrices traced (dark orange). In this orbit complex example, note that the singularity pairing constraints (dashed-black) are satisfied (most easily seen via counting of separatrix crossings), but trajectories in the highlighted blue Type II cell still helix. An ordering inequality constraint imposed on the dashed green curve would enforce appropriate behavior of the separatrices and resolve the helicing cell by preventing any separatrices from crossing it.

4.5.2 Connections across faces: Similar to past stripe tracing strategies [Mitra et al. 2025, 2024, 2023], we connect vertices across faces through the matching of *virtual vertices* (green and blue nodes in inset) which are computed by intersecting stripe level sets with triangle borders. These virtual vertices are then connected to knit graph vertices computed via stripe intersections.

On non-singular edges, the number of virtual vertices on either side of the edge are equal and the matching of virtual vertices is trivial via a simple ordering. On course singular edges, virtual vertex matching follows the short-row pairing scheme described in Sec. 4.3. We check all candidate pairings from the left of the edge to

the right and, for each, verify that the resulting short-row construction produces the expected short-row (one that starts/ends at the positive/negative singular edge in the pairing scheme from Sec. 4.3). The pairing that satisfies this criterion is selected. On wale singular edges, the matching is simply a minimum distance approach. We pair virtual vertices from the left of the edge to its nearest counterpart on the right of the edge. As a final robustness step, once virtual vertices are matched, their positions are updated to the average position of the matched pair and the stripe level set is slightly modified (see inset). Unlike in [Mitra et al. 2025], this matching procedure enables us to generate graphs at higher resolutions than the length of mesh edges.



4.6 User constraints

We retain all the user capabilities of past stripe and curl-based knitting planning frameworks whilst introducing novel machinery for new methods of user control. Most of these amount to modification of the curl measures, which are then fed to the start of our pipeline, beginning with surface quantization §4.2.

Curl Masking: As in Mitra et al. [2025], we can mask out the curl from a user-specified region thereby inserting no singularities in the region. Such a masking is crucial to allow for complex color work or texturing, as can be seen in some rendered examples in Figs. 11, 12. If m is the user-specified masking region, we set $\mu_m = 0$ before running our singularity placement procedure.

In Fig. 11, a region is masked out on a shoe upper for insertion of a rasterized checkerboard SIGGRAPH logo. With the presence of shaping within the logo (green highlights) the raster grid is disturbed and rows merge together, leading to a loss of a 1-1 map between loops and the logo raster. The artifacts can clearly be seen "downstream" of the shaping, as there is no way to recover the checkerboard alternation exactly. Fig. 12 demonstrates the same concept via a moss stitch texture, which is formed by a checkerboard pattern of alternating front knits and back knits, giving it a distinct aesthetic look and feel. Again, when the raster grid is disturbed, there are clear artifacts caused by the shaping.

Level set constraint: We introduce a novel level set constraint that allows the user to align the knit graph with sharp features or seams in the underlying mesh. To do this, we ask that the time function take a constant value over the feature edges. In particular, if v is the set of vertices in the constrained edge set e, we set equality constraints on all vertices in the same feature line. This is merely an additional linear constraint in the standard quadratic Dirichlet energy minimization used to obtain harmonic time functions. We additionally impose $\sigma_e = 0$ to tightly align the stripes. An example of this constraint is shown in the box model in Fig. 13. This new constraint is more general than the boundary/patch alignment constraint of Mitra et al. [2025] as it allows for direct modification of the knitting time function.

Boosting parameter: Placing "apparent" seams on a model is an important task for knit designers. These seams show up when knit singularities are gathered onto lines or curves, and often are used to

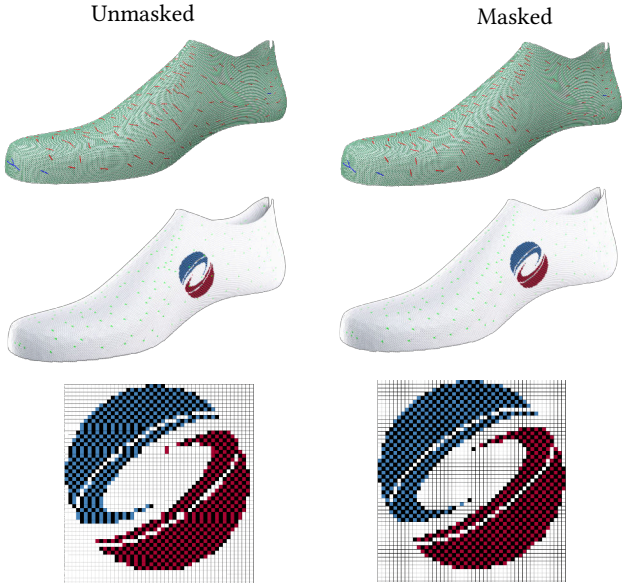


Fig. 11. Our curl masking procedure allows for the artifact-free placement of logos. Left column: A geometrically informed solution from our algorithm on a disk model of a shoe-upper. The render shows line artifacts arising due to singularities running through the logo. The singularities disturb the regular grid structure (bottom), making an exact checkerboard pattern impossible to recover. Right column: We can mask out the curl in a local region, where the logo is to be placed. No singularities pass through the logo as shown in the render and the logo is mapped onto a regular grid without any artifacts.

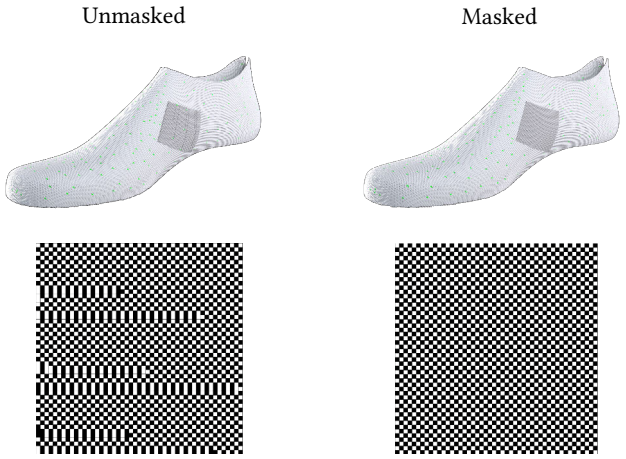


Fig. 12. The masking procedure also allows for an artifact-free application of a "moss texture", which is an alternating checkerboard pattern of front knits and back knits. We apply the same masking procedure to the shoe upper from Fig. 11. Left: The presence of singularities in the texturing area causes obvious visual artifacts to appear. Right: Our method accommodates the removal of all singularities from a texturing area resulting in artifact-free texturing. Bottom: The grid representations represent the texture with front knits corresponding to black blocks and back knits corresponding to white blocks.

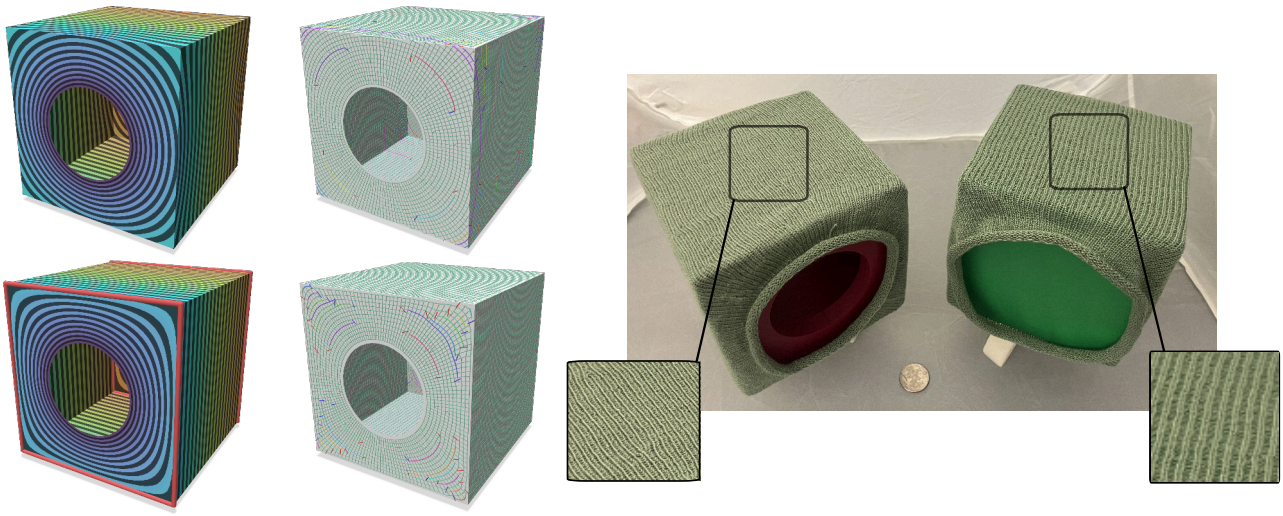


Fig. 13. Level set constraints. Top: Box model without constraining the time function isovalues results in the course rows that are not aligned with the edges of the box. Middle: Constraining the time function value along certain box edges (red) enforces course row alignment. Bottom: Knitted results without (left) and with (right) level set constraints. Note the non-orthogonality of the wales near the corners and the presence of shaping on the sides on the left (visible in zoom-in).

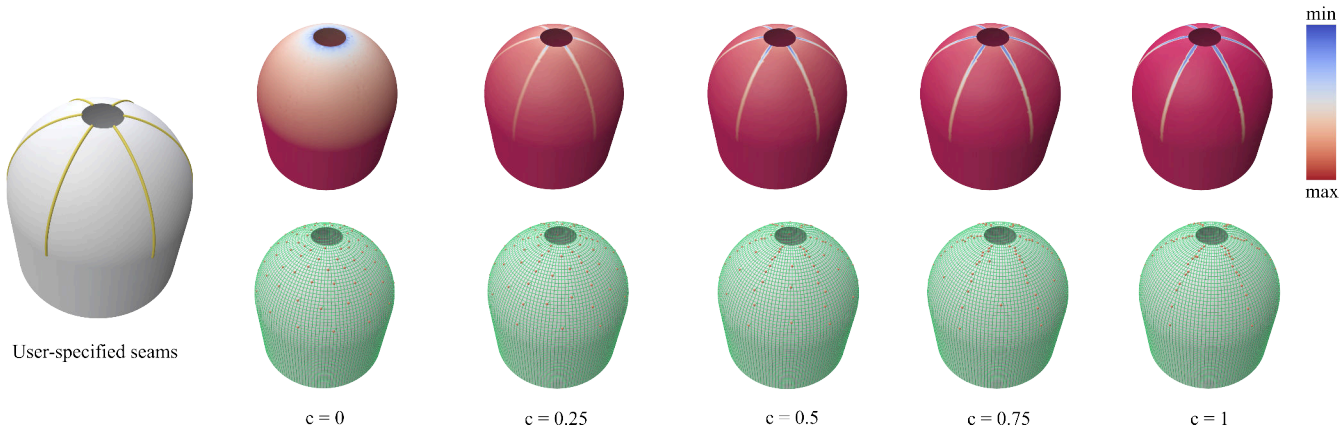


Fig. 14. Boosting parameter on a beanie model. Our method accommodates user-specified seams along with a boosting parameter which controls how much of the original curl should be concentrated at the seams. At $c = 0$, we have the original singularity distribution as can be seen with the uniformly dispersed decreases in orange. Increasing c concentrates more curl at the seams until eventually all the decreases are placed along it, creating an *apparent seam* in the knit graph. Knit versions of the $c = 0$ and $c = 1$ models are in Fig. 20.

gather shaping operations to allow nearby regions to remain regular. Our method allows for the placement of such seams via a boosting parameter c, which controls how much of the original curl should be transferred to the seam, while preserving total curl mass. This is a novelty with respect to Mitra et al. [2025] as there was no interpolation parameter and no curl mass preservation guarantees with its greedy insertion strategy. When $c = 0$, none of the underlying curl is moved to the seams while increasing c transfers more curl to the specified seams. In particular, if B is a set of boosting vertices where seams are to be placed, then we define a uniform measure μ_B on these vertices. The boosted curl with user-specified parameter c is then:

$$\mu' = (1 - c)\mu + c\left(\frac{\int_M d\mu}{\int_M d\mu_B}\right)\mu_B \tag{21}$$

Note that the constant ensures that $\int \mu' = \int \mu$ for all values of c. This allows users to control how much of the *apparent seams* will be visible in the final knit graph. Since we preserve total curl mass, the number of singularities inserted remains constant on variation of this parameter. An example of this interpolation scheme on a beanie model is shown in Fig. 14.

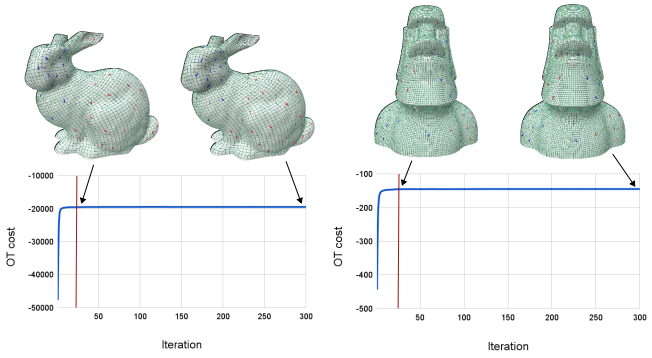


Fig. 15. Plots of OT energy vs iteration count during our Lloyd steps. Note that our formulation maximizes a concave objective term. Our method produces qualitatively similar singularity placements and knit graphs when run for 300 Lloyd iterations vs terminating at OT convergence with a $\varepsilon_{\text{Lloyd}} = 10^{-4}$. We highlight the short-row placements on the bunny and moai models run with the different stopping criteria.

Fig. 16. Top. Left: Course curl signal. Middle: Power cells quantizing positive course curl. Right: Knit graph with left ends of short rows in red and right ends of short rows in blue. Bottom. Left: Knitted bunny without any stuffing or dressing. Right: We dress our knits on underlying 3D prints.

5 Results

Our implementation is coded in C++, and the optimization problems are solved using the vanilla Gurobi solver [Gurobi Optimization, LLC 2022]. We use the open-source implementations of the scalar heat method [Crane et al. 2013b] and log-map computation [Sharp et al. 2019c] in Geometry-Central [Sharp et al. 2019a]. We use an open-source C++ implementation of LBFGS [Qiu and Toewe 2019] and Polyscope [Sharp et al. 2019b] for visualization. All timing experiments were run on an M3 Macbook Pro with 32GB of RAM.

Algorithm parameters: In early versions of our implementation, we prioritized achieving knitted models using a conservative estimate of 300 Lloyd iterations resulting in good solutions. Letting the algorithm run longer usually leads to oscillations about a local minimum. However, on further analysis, setting a permissive tolerance of $\varepsilon_{\text{Lloyd}} = 10^{-4}$ leads to convergence in 50-100 iterations with no material improvement in the OT energy or singular configuration. We show this behavior in Fig. 15, where we plot the OT energy against iteration number and qualitatively compare knit graphs produced with the different stopping criteria. Detailed runtimes are reported in Table 4. For the weights optimization, the L-BFGS tolerance is set to $\varepsilon_{\text{L-BFGS}} = 10^{-5}$ and the memory parameter is set to $m = 6$.

We validate our method through physical fabrication of several models on an industrial knitting machine. In particular, the open-source implementation of the Autoknit [Narayanan et al. 2018] scheduler is used to generate machine-knitting instructions from our helix-free knit graphs. Fabrication is done via a Shima Seiki SWG91N2 15-gauge v-bed knitting machine using 2/28NM rayon yarn at a 30-stitch size at a rate of 0.8 m/s. A diverse array of topologies, geometries, and density levels are shown throughout in Figs. 1, 13, 16, 17, 18, 19, and 20. Models have been dressed on 3D-printed mannequins to better show geometric shaping, especially in concave regions, where stuffing leads to internal pressure that pushes these regions outward (e.g., see the teaser duck model of [Mitra et al. 2025]

Fig. 17. From left to right: Cactus, pipes, heart, doughnut. Top row: Knit graphs highlighted with singularities. We show the short-rows on the cactus and doughnut models and the increases/decreases on the pipes and heart model. Bottom: Knitted results dressed on 3D print molds.

in the underside of the neck region). This was also noted in the crochet setting in [Edelstein et al. 2022]. Note that most mannequins were 3D-printed in multiple interlocking parts to accommodate dressing in the presence of nontrivial genus or high shaping.

The open-source implementation of the Autoknit scheduler occasionally struggles to complete on several of our models. This is likely due to topological complexity (split-2-3, full-sleeve dress) or dense

Fig. 18. From left to right: Glove, duck, sock with knit graphs highlighting increases/decreases on the glove and duck models and short-row ends on the sock model. Bottom: Knitted results dressed on 3D print molds.

Fig. 19. From left to right: Knit graphs for dress and pants model with highlighting of increases/decreases on dress and short-row ends on pants. Bottom: Knitted results. The dress is fitted onto an underlying 3D print of the torso region to better show shaping.

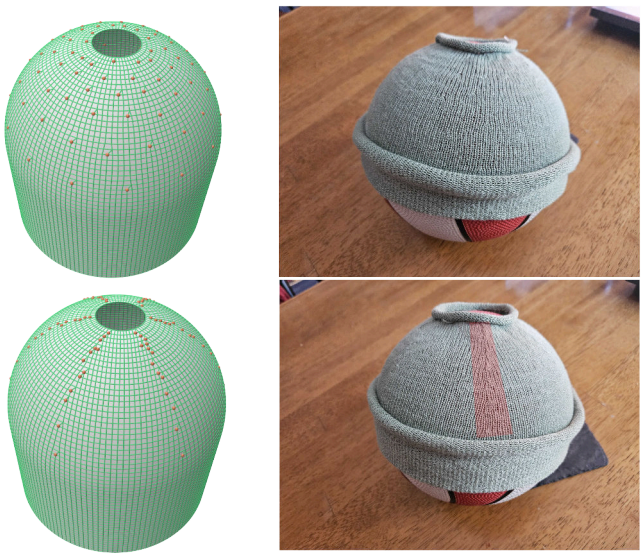


Fig. 20. Knitted results of the beanie model. Top: We set $c = 0$, transferring no curl to the seams. This results in a graph with uniformly dispersed decreases. Bottom: For $c = 1$, all the curl is transferred to the seams resulting in a series of decreases (apparent seams), one of which is highlighted in red.

local shaping (B71, B42). As such, we also present a virtual gallery of several helix-free knit graphs, with highlighted short rows, that can be visually inspected at `rahulmitra.xyz/projects/powerknit/gallery`. It includes knit graphs for all the knit models, rendered models, and examples that we could not get to schedule appropriately. The additional complexity present here serves as a demonstration of the robustness and versatility of our method. In particular, we note again that the ordering constraints described in §4.4 and shifting of local stripe positions in Sec. 4.5 enabled robust helix-free stripe optimization and knit graph extraction in high-density settings (when period is low, and singularities are many). The full source code/data for the website are also included in supplementary materials to ensure no new results are added post-deadline. Full mesh combinatorics, knit graph combinatorics, knit graph quality metrics, and running times are included for all knit and virtual results in Supp. §A. Each aspect is discussed via a few representative examples in sections below.

Comparison to Autoknit: While Autoknit's knit graph creation [Narayanan et al. 2018] (separate from the scheduler) produces helix-free machine-knittable graphs, they do not allow for any user-customization or editing beyond selection of starting and ending boundaries. As demonstrated in §4.6 and in associated Figs. 11, 13, 14, our methods allow for several desirable editing tools. Additionally, our knit graphs are much smoother making them suitable for rendering pipelines such as in Figs. 11, 12. We compare quad corner angle statistics on a few models in Fig. 21.

Comparison to Mitra et al. [2025]: Another closely related knit singularity placement work is that of Mitra et al. [2025]. They use an iterative greedy algorithm to check potential placement of singularities until objective $\mathcal{F}(\sigma)$ in Problem (1) stops improving. This iterative approach requires solving several linearly-constrained quadratic optimization problems for the placement of each singularity,

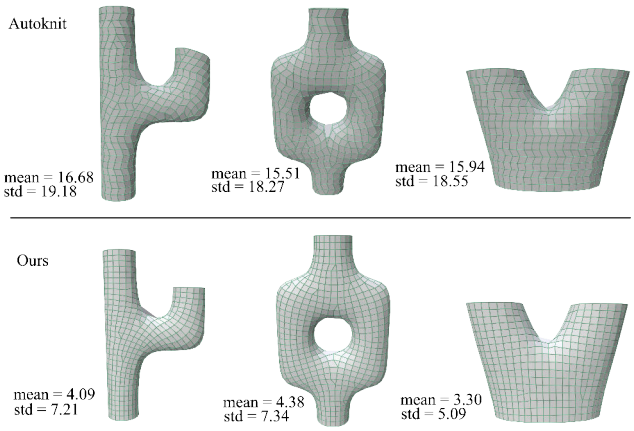


Fig. 21. Top row: Knit graphs produced by Autoknit [Narayanan et al. 2018]. Bottom row: Knit graphs produced by our method. We present the average and standard deviation of quad corner angle error from a target of 90°. While Autoknit does generate meshes with edge lengths errors < 10% (see Fig. 24 in Narayanan et al. [2018]), our average angle errors and standard deviations are far lower making our approach suitable for rendering pipelines requiring mesh regularity.

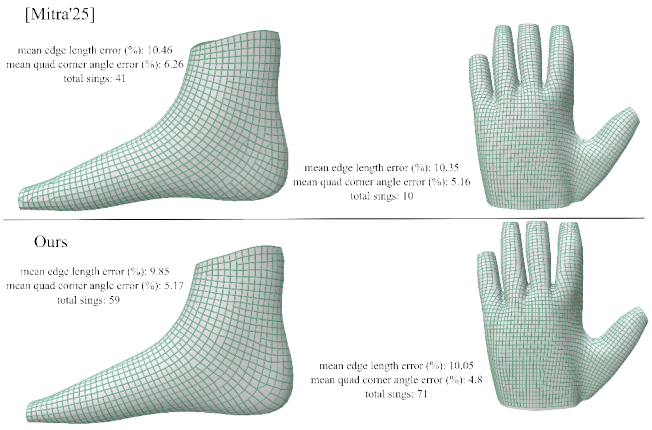


Fig. 22. Comparison with knit graphs generated by Mitra et al. [2025]. Our method typically inserts more singularities, enabling greater shaping in the resulting knit graph, while achieving slightly improved average edge length errors and mean quad corner angles.

resulting in poor scaling and performance. We replace such an iterative scheme with a single global algorithm that jointly solves for singularity positions. This results in a general speed-up in placement as shown in Table 1. Recall that our method inserts a more principled number of singularities via Eq. (14). This number better approximates the total curl measure mass, whereas there is no control on this number in [Mitra et al. 2025]. In Table 1, we ran our method with the same number of singularities that [Mitra et al. 2025] ultimately inserts for a normalized comparison. We note also that these models were run with relatively coarse stripe patterns

Table 1. We compare the total running time (m:ss) of our method (until OT convergence) on several models against the running time of Mitra et al. [2025]. Note that to achieve a fair comparison, we time our method on the same number of singularities inserted by Mitra et al. [2025]. The figure labels refer to the models considered, but the runtimes reflect higher periods (coarser knit graphs) than shown in the figures.

Model	Mitra'25 Sing.	Mitra'25 Time	Ours	Speed-up
Sock (Fig. 18)	31	0:29	0:19	1.6x
Dress (Fig. 19)	14	0:21	0:03	8.3x
Cactus (Fig. 17)	22	0:29	0:06	4.9x
Elbow Sleeve	22	0:06	0:01	11.8x
Pipes (Fig. 17)	62	1:15	0:12	6.0x
Heart (Fig. 17)	189	5:30	0:16	20.6x

Table 2. Comparison of runtimes (m:ss) against Mitra et al. [2025] in dense (low period) settings. The methods are cut off after a timeout of 10 minutes. Note that quality comparisons are challenging as the stripe optimization and knit graph construction of Mitra et al. [2025] are not robust enough in these dense settings.

Model	Mitra'25 Time	Ours	Speed-up
Glove	2:33	3:45	0.7x
Sock	1:15	0:37	2.0x
Pipes	4:05	0:24	10.2x
Duck	timeout	1:10	
Dress	3:34	0:36	5.9x
Moai	timeout	2:34	
Bunny	timeout	0:58	

Table 3. Comparison of knit graph quality with Mitra et al. [2025] on coarser models (higher periods). #S is the total number of knit singularities, e_{length} is the median error on edge lenghts, and e_{angle} is the median deviation of angles from 90°.

	Mitra'25			Ours		
Model	#S	e_{length}	e_{angle}	#S	e_{length}	e_{angle}
Sock	31	9.5%	5.4%	44	5.7%	3.2%
Dress (coarse)	17	9.0%	5.2%	48	5.6%	2.8%
Dress (fine)	42	13.6%	6.6%	231	6.0%	2.8%
Cactus	27	5.3%	3.8%	25	11.3%	1.8%
Heart	189	6.6%	4.1%	140	5.2%	2.8%
Pipes	68	5.7%	2.8%	79	7.3%	2.2%
Bunny	63	17.6%	11.3%	48	16.0%	7.6%
Pants 3D	9	1.7%	1.2%	28	2.1%	1.6%
Mean		**8.6%**	**5.0%**		**7.4%**	**3.1%**

that may not match the referenced figures. The speedup typically lies in the 5x - 10x range.

In addition to speed improvements, we also highlight improved robustness over [Mitra et al. 2025], gained by the additional constraints of §4.4. These constraints allowed for knit graph construction for

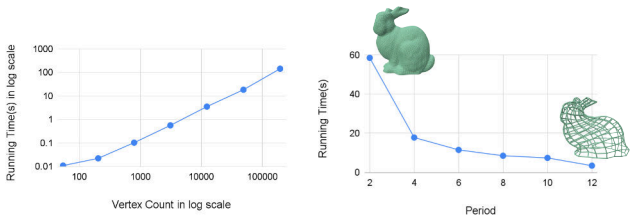


Fig. 23. Running time dependence on the bunny model. Left: Running time (s) vs Vertex Count on a log-log scale. Our algorithm runs at interactive speeds at coarser mesh resolutions. Right: Running time (s) vs Period. Our method slows at lower period (higher stripe resolutions) due to more singularity insertions required.

larger models with much denser shaping. Table 2 shows speed comparisons on such models with a timeout of 10 minutes. Notably, knit graph quality comparisons were not possible as the graph construction of [Mitra et al. 2025] generally did not successfully result in a helix-free graph. The average speedup for models that completed within time limits was approximately 3x.

Finally, we examine quality comparison metrics in the coarser setting, where [Mitra et al. 2025] is robust enough to produce valid output. Comparison of edge length error and quad corner angle statistics is shown in Table 3. On these models, our average median edge length error was 7.4% and our average median angle error was 3.1%, compared to 8.6% and 5.0% for [Mitra et al. 2025], respectively. As can be seen, our performance on these coarser models is better on average and at least comparable in all cases. Higher variability in the performance of [Mitra25] was found, due to its iterative and greedy approach. Additional visual comparisons on the sock and glove models are shown in Fig. 22. They both show improvements in these metrics, reflecting more evenly-spaced courses and wales.

Scaling: The runtime of our algorithm is governed primarily by mesh resolution and the striping period. On coarser meshes ($\sim 10^2$–10^4 vertices), the method typically finishes in a few seconds. The cost of a Lloyd iteration is dominated by the computation of the heat kernel and the log map. These amount to a fixed number of sparse linear solves with a pre-factorized Cholesky decomposition, making the complexity nearly linear in the mesh size. This trend is shown in the log–log plot in Fig. 23 (left). Runtime also increases as the striping period decreases (Fig. 23, right): a smaller period leads to more singularities to be placed as the total curl mass remains constant, Eq. 14. The per-Lloyd iteration runtime scales linearly with the number of singularities, and more Lloyd iterations are required to converge.

Notably, the mesh resolution and striping period are not entirely independent. For a given period P, a mesh with average edge length $\leq 2P$ is recommended. When the mesh is too coarse relative to the period, it opens up the possibility for discrete stripe singularities of absolute index greater than 1, e.g., ±2. This is undesirable from a numerical modelling standpoint as optimally quantized measures will have some separation between support points. Being unable to represent this discretely will lead to suboptimal stripe patterns and thus knit graphs.

6 Conclusion

Placing knit singularities is a crucial task for ensuring that fabricated results conform to an input geometry. In this work, we present a global algorithm that jointly optimizes for knit singularity positions leveraging an optimal-transport-based notion of quantization for the curl of the normalized time function gradient. This required the generalization of heat-kernel-based geodesic Voronoi diagrams to geodesic power diagrams. Our method avoids the need for expensive mixed-integer solves and scales better than previous iterative approaches. We also introduce novel ordering constraints that more robustly achieve helix-free (machine-knittable) graphs, and present new methods for incorporating user input. We showcase the efficacy of our method on both a diverse range of knitted results and a virtual gallery of helix-free knit graphs.

Limitations and future work: While our method is faster than prior approaches, there are several improvements to be explored. For instance, currently our support sites are randomly initialized. A more intelligent initialization, such as borrowing placements from Knöppel et al. [2015], may result in faster convergence. One might also consider multi-scale approaches achieved by mesh restriction/prolongation and/or a scheduled increase in period (and thus in number of singularities with successively smaller mass). We are also hopeful that our novel notion of geodesic power diagrams on surfaces may find application in other settings, such as extending blue noise algorithms [De Goes et al. 2012] to surface settings, optimal remeshing [Mullen et al. 2011], and addressing quantization artifacts for 3D printing [Morsy et al. 2022].

Acknowledgments

This work was partially supported by Wallonie-Bruxelles International and NSF (CISE 2341880). The authors would like to thank Jim McCann for assistance debugging knit tracing on several models, Daniel Scrivener for figure help, and Boston University's Engineering Product Innovation Center for use of their 3D printers.

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.

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.

F. Aurenhammer. 1987. Power Diagrams: Properties, Algorithms and Applications. *SIAM J. Comput.* 16, 1 (1987), 78–96. https://doi.org/10.1137/0216006 _eprint: https://doi.org/10.1137/0216006.

David Bommes, Timm Lempfer, and Leif Kobbelt. 2011. Global structure optimization of quadrilateral meshes. In *Computer graphics forum*, Vol. 30. Wiley Online Library, 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. https://doi.org/10.1111/cgf.12014 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12014

Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. 2018. An interpolating distance between optimal transport and Fisher–Rao metrics. *Foundations of Computational Mathematics* 18, 1 (2018), 1–44.

Sebastian Claici, Edward Chien, and Justin Solomon. 2018. Stochastic Wasserstein Barycenters. In *Proceedings of the 35th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 999–1008. proceedings.mlr.press

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013a. 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, Clarisse Weischedel, and Max Wardetzky. 2013b. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)* 32, 5 (2013), 1–11.
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, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue Noise through Optimal Transport. *ACM Transactions on Graphics (SIGGRAPH Asia)* 31, 6, Article 171 (Nov. 2012), 171:1–171:11 pages. https://doi.org/10.1145/2366145.2366190
Fernando De Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue noise through optimal transport. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–11.
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.
Michal Edelstein, Hila Peleg, Shachar Itzhaky, and Mirela Ben-Chen. 2022. AmiGo: Computational Design of Amigurumi Crochet Patterns. In *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication (SCF '22)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3559400.3562005 arXiv:2211.01178 [cs.GR]
Xifeng Gao, Wenzel Jakob, Marco Tarini, and Daniele Panozzo. 2017. Robust hex-dominant mesh generation using field-guided polyhedral agglomeration. *ACM Trans. Graph.* 36, 4, Article 114 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073676
Steven J. Gortler, Craig Gotsman, and Dylan Thurston. 2006. Discrete one-forms on meshes and applications to 3D mesh parameterization. *Computer Aided Geometric Design* 23, 2 (2006), 83–112.
Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual.
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* (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.
K.H. Leong, S. Ramakrishna, Z.M. Huang, and G.A. Bibo. 2000. The potential of knitting for engineering composites—a review. *Composites Part A: Applied Science and Manufacturing* 31, 3 (2000), 197–220. https://doi.org/10.1016/S1359-835X(99)00067-6
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.
Jenny Han Lin, Benjamin Jones, and Edward Chien. 2025. Computational Knitting. Lecture at SGP Graduate School 2025. https://school.geometryprocessing.org/summerschool-2025/index.html#course3 Symposium on Geometry Processing (SGP) Summer School.
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.
Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489
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. 2017. On-Demand Machine Knitting for Everyone. Lecture at Robotics Institute Seminar, CMU. https://www.youtube.com/watch?v=iEaK68VRAng
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, Mattéo Couplet, Tongtong Wang, Megan Hoffman, Kui Wu, and Edward Chien. 2025. Curl Quantization for Automatic Placement of Knit Singularities. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 158, 10 pages. https://doi.org/10.1145/3721238.3730715
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.
Mostafa Morsy Abdelkader Morsy, Alan Brunton, and Philipp Urban. 2022. Shape dithering for 3D printing. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–12.
Patrick Mullen, Yiying Tong, Pierre Restrepo, Peter Schröder, and Mathieu Desbrun. 2011. HOT: Hodge-optimized triangulations. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–11. https://doi.org/10.1145/2010324.1964998
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.
Gabriel Peyré and Marco Cuturi. 2019. Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends® in Machine Learning* 11, 5-6 (2019), 355–607. https://doi.org/10.1561/2200000073
Mariana Popescu, Matthias Rippmann, Andrew Liew, Lex Reiter, Robert J Flatt, Tom Van Mele, and Philippe Block. 2021. Structural design, digital fabrication and construction of the cable-net and knitted formwork of the KnitCandela concrete shell. In *Structures*, Vol. 31. Elsevier, 1287–1299.
Mariana Popescu, Matthias Rippmann, Tom Van Mele, and Philippe Block. 2020. KnitCandela-Challenging the construction, logistics, waste and economy of concrete-shell formworks. *Fabricate 2020: Making Resilient Architecture* (2020), 194–201.
Yixuan Qiu and Dirk Toewe. 2019. Lbfgs++. https://lbfgspp.statr.me/
Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic global parameterization. *ACM Trans. Graph.* 25, 4 (Oct. 2006), 1460–1485. https://doi.org/10.1145/1183287.1183297
Filippo Santambrogio. 2015. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Progress in Nonlinear Differential Equations and Their Applications, Vol. 87. Birkhäuser, Cham. https://doi.org/10.1007/978-3-319-20828-2
Shahida Sharmin and Sean Ahlquist. 2016. Knit Architecture: Exploration of Hybrid Textile Composites Through the Activation of Integrated Material Behavior.
Nicholas Sharp et al. 2019b. Polyscope. www.polyscope.run
Nicholas Sharp, Keenan Crane, et al. 2019a. GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing. https://geometry-central.net/.
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019c. The vector heat method. *ACM Transactions on Graphics (TOG)* 38, 3 (2019), 1–19.
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. 2024. 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.

Jenny Underwood. 2009. *The Design of 3D Shape Knitted Preforms*. Ph. D. Dissertation. RMIT University, Melbourne, Australia. researchbank.rmit.edu.au Doctor of Philosophy.

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.

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.

A Knit graph metrics

We measure the fidelity of our knit graphs across two metrics: edge length errors and quad corner angle errors. We aim for a target edge length of the striping period, P. For angle errors, we only measure angles on quad faces of the knit graph. In general, our average edge length errors are $\leq 10\%$ and average angle errors are $< 5\%$.

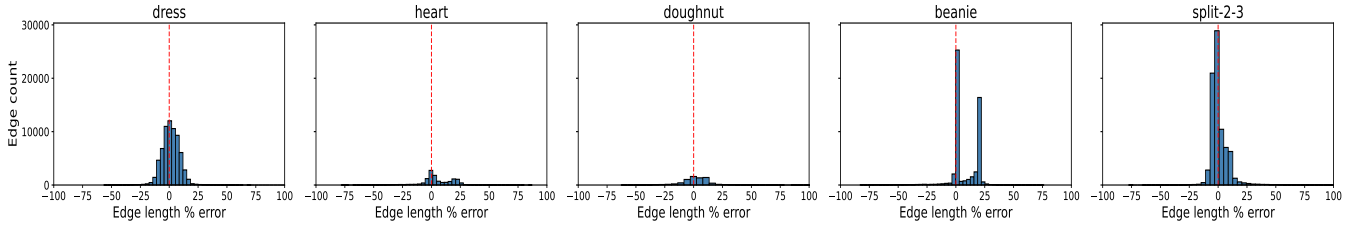


Fig. 24. We present edge length errors on various knit graphs. Our average edge length error is $< 10\%$.

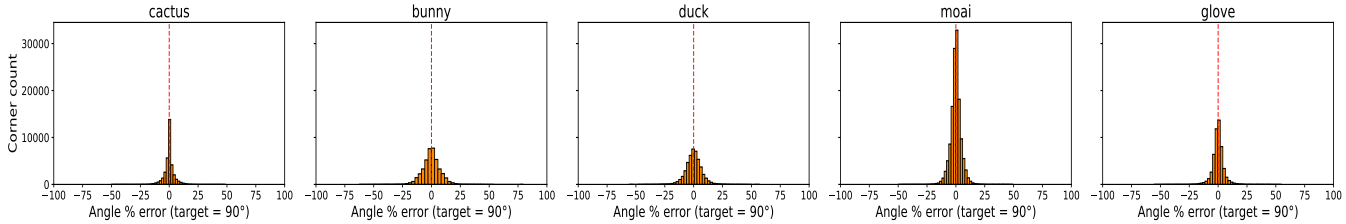


Fig. 25. We also show quad corner angle errors on various knit graphs, with a target angle of 90°. We achieve an average error % of less than 5%.

Table 4. Model statistics. For each model, we report mesh combinatorics and size of knit graph. We also compute error statistics for mean edge length/quad corner angle error. For quad corners, we aim for a target angle of 90°. Our pooled mean edgle length error is less than $< 10\%$ and our pooled angle error is $< 5\%$.

Model	#V	#F	#Sing.	Knit graph rows	Runtime (m:ss)	Mean edge length error (%)	Mean quad corner angle error (%)
Cactus	7646	15114	110	150	0:15	9.19	2.89
Bunny	8909	17620	359	178	0:58	12.44	5.86
Glove	37493	74524	336	227	3:45	8.54	3.57
Sock	7908	15700	139	135	0:37	7.05	5.31
Doughnut	797	1525	95	73	0:01	9.11	5.27
Dress	8634	16995	231	176	0:36	5.96	2.83
Pipes	9790	19456	142	131	0:24	8.90	3.42
Duck	9645	19234	295	195	1:10	15.1	5.32
Heart	5565	11032	126	87	0:12	10.9	4.28
Pants	9479	18777	62	164	0:11	3.84	2.19
Moai	16984	33544	393	250	2:34	7.48	3.33
Beanie (seams)	12911	25594	85	60	0:12	7.59	2.23
Box (constrained)	5678	11264	92	120	0:11	12.87	4.95
Dress (full sleeve)	10278	20384	86	125	0:10	6.99	3.04
Split-2-3	22333	44032	271	243	2:03	2.44	7.94
B71	4688	9204	89	146	0:04	8.87	4.32
B42	4541	8692	84	150	0:02	4.69	2.39
Shoe (disk)	2950	5310	225	613	0:14	7.04	6.09
Split-1-2	3295	6400	82	102	0:06	4.37	2.22
Elbow Sleeve	1343	2558	24	34	0:01	6.82	4.25
Retinal	7418	14797	383	206	2:30	12.54	5.33