Closed space-filling curves with controlled orientation for 3D printing

Adrien Bedel, Yoann Coudert-Osmont, Jonàs Martínez, Rahnuma Islam Nishat, Sue Whitesides, Sylvain Lefebvre

▶ **To cite this version:**

Adrien Bedel, Yoann Coudert-Osmont, Jonàs Martínez, Rahnuma Islam Nishat, Sue Whitesides, et al.. Closed space-filling curves with controlled orientation for 3D printing. 2021. hal-03185200v1

HAL Id: hal-03185200

https://inria.hal.science/hal-03185200v1

Preprint submitted on 30 Mar 2021 (v1), last revised 19 Apr 2022 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Closed space-filling curves with controlled orientation for 3D printing

ADRIEN BEDEL, Université de Lorraine, CNRS, Inria, LORIA
YOANN COUDERT-OSMONT, ENS Lyon
JONÀS MARTÍNEZ, Université de Lorraine, CNRS, Inria, LORIA
RAHNUMA ISLAM NISHAT, University of Victoria
SUE WHITESIDES, University of Victoria
SYLVAIN LEFEBVRE, Université de Lorraine, CNRS, Inria, LORIA

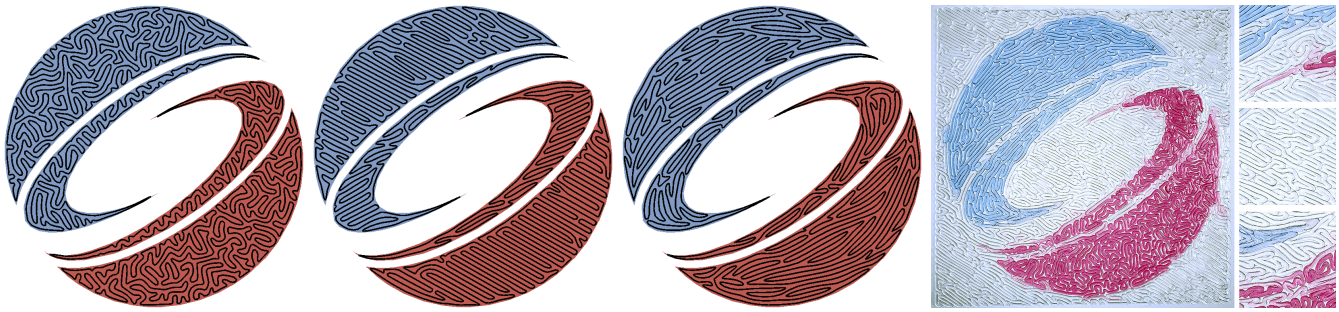


Fig. 1. Closed space-filling curves optimized under **(a)** isotropy, **(b)** anisotropy and **(c)** boundary alignment objective functions. Isotropy and anisotropy are optimized in terms of distribution of angles, while alignment is specified in a vector field (here parallel to the boundary). By extruding material along the closed curve we fabricate a multi-material plate **(d)** in a single, continuous motion, with progressive gradations of the material mixture. Our optimizer discourages the curve from crossing from one area to another, allowing the logo to be properly reproduced despite slow transitions between material mixtures.

We explore the optimization of closed space-filling curves under orientation objectives. By solidifying material along the closed curve, solid plates can be manufactured in a single continuous extrusion motion. The control over orientation enables the deposition to align with specific directions in different areas, or to produce a locally uniform distribution of orientations, patterning the solidified volume in a precisely controlled manner.

Our optimization framework proceeds in two steps. First, we cast a combinatorial problem, optimizing for Hamiltonian cycles within a specially constructed graph. We rely on a stochastic optimization process based on local operators that modify a cycle while preserving its Hamiltonian property. Second, we use the result to initialize a geometric optimizer that improves the smoothness and uniform coverage of the cycle while further optimizing for alignment and orientation objectives.

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

Additional Key Words and Phrases: space-filling curve, Hamiltonian cycle, orientation distribution, 3D printing

1 INTRODUCTION

In this work, we consider the problem of covering a planar surface with a non-intersecting, space-filling closed trajectory while precisely controlling its orientation (see Figure 1). The spacing around the trajectory is everywhere close to a same nominal value, making it possible to fabricate solid plates (or layers) through material extrusion along the cycle (3D printing).

The orientation control enables us to precisely pattern the deposited material. The closed (cyclic) nature of the trajectory permits choosing a starting point, simplifying motion planning. The continuous extrusion flow results in a uniform, uninterrupted extrusion process. Multi-material plates can be fabricated while progressively grading properties through continuous material mixtures.

Our trajectories' orientation can be controlled both in terms of *global distribution* of orientations (isotropy/anisotropy) and *local alignment* specified as a vector field. This control may be different in different zones. The trajectory can also be optimized to remain for as long as possible in a designated area – an objective we refer to as *zoning*. The zoning objective ensures that multi-material mixtures remain stable while in a specific area.

The requirements of 1) a smooth, closed, and non-intersecting trajectory 2) covering the entire surface with everywhere the same spacing while 3) being freely orientable, form a seemingly impossible problem, only approachable through optimization of the various objectives. We approach this by decomposing the optimization into two steps, manipulating different aspects of the problem.

The first step is a *combinatorial optimizer*, exploring Hamiltonian cycles in a specially constructed graph covering the surface. The optimizer starts from an initial cycle and modifies it through operators local to the graph, improving the objective functions. The operators locally update the cycle while preserving its Hamiltonian property.

Authors' addresses: Adrien Bedel, Université de Lorraine, CNRS, Inria, LORIA; Yoann Coudert-Osmont, ENS Lyon; Jonàs Martínez, Université de Lorraine, CNRS, Inria, LORIA; Rahnuma Islam Nishat, University of Victoria; Sue Whitesides, University of Victoria; Sylvain Lefebvre, Université de Lorraine, CNRS, Inria, LORIA.

Once the global connectivity is determined, our second *geometric optimizer* further refines the closed curve's geometry. The result is a smooth, evenly distributed trajectory. Alignment, orientation, and zoning objectives are considered at every step of the process, using the same base formulations.

In this paper, a *cyclic curve* means either a simple closed curve on a surface (the plane) or a cycle in a graph embedded in the surface; the context will make the meaning clear. As the cycles in the graphs become space-filling, the two meanings approach each other.

2 RELATED WORK

Space-filling curves and Hamiltonian cycles in graphs find many uses in Computer Graphics. In particular, they can be used to define coherent linear traversals of graphical data, such as organizing triangle and quad meshes into strips [Arkin et al. 1996; Gopi and Eppstein 2004; Gurung et al. 2011; Taubin 2003], or improving image compression rates [Dafner et al. 2000].

Space-filling curves have unique aesthetic qualities, which led to their use for artistic creations. Bosch et al. [2010; 2004] and Kaplan et al. [2005] explore continuous line drawings using TSP solvers to find a Hamiltonian cycle. Wong and Takahashi [2013] stylize images, growing a cycle in oriented grids obtained from quad meshing. A similar growth process is performed in [Dafner et al. 2000]. The approach of Pedersen and Singh [2006] grows space-filling curves from initial simple closed curves (e.g., circles), driving the growth through a variety of user-driven guidance fields. This generates beautiful and intricate patterns. The growth is performed with a simulation scheme akin to a particle system, with repulsion and attraction forces. Sharp and Krane [2018] generate space-filling curves as a side effect of their technique for surface cutting through a similar growth strategy driven by a level-set optimization framework.

Growth approaches give extensive control over the final trajectories, as demonstrated in [Pedersen and Singh 2006; Wong and Takahashi 2013]. However, setting up robust and stable growth parameters can be challenging. The time it takes for the curve to invade the domain depends heavily on the region's complexity and size to be filled, with graph irregularities potentially leading to the algorithm failing and requiring user intervention [Wong and Takahashi 2013]. Finally, continuously growing from a simple closed curve limits the cycles that can be achieved; most notably, this implies there will always be an even number of trajectories between two outer contours, making trajectories such as the one shown in Figure 2 impossible to achieve. Conversely, a graph theory approach can find general cycles much more efficiently, as it is combinatorial and – on specific graphs – supported by a linear time algorithm [Taubin 2003]. This is demonstrated in Aklenam et al. [2013] to generate Hamiltonian cycles along surfaces for various artistic effects.

Space-filling curves also have properties that make them interesting as deposition trajectories for Additive Manufacturing [Jiang and Ma 2020]. Chen et al. [2020] enumerate all cycles within a square regular grid, evaluating each cycle for the resulting thermal properties during deposition. Lin et al. [2019] explore space-filling curves to obtain close to isotropic mechanical behaviors in the produced part – while typical raster fill patterns (*zig-zags*) lead to anisotropic mechanical responses. Papacharalampopoulos et al. [2018] explore the use of Hilbert curves for continuous deposition, noting that uninterrupted extrusion reduces print time. Continuity is also an important factor in final quality when extruding pastes or ceramics [Hergel et al. 2019]. Giannatsis et al. [2015] construct space-filling curves following a material gradient. Chen et al. [2017] vary the local density of the trajectories following an input image. Zhao et al. [2016] generate contour parallel fill trajectories that form singly connected spirals within a contour, leading to a continuous fill with smooth trajectories. Soler et al. [2017] create 3D trajectories as Hamiltonian cycles in grids for wireframe 3D printing. In the context of two-photon lithography, Dahaeck et al. [2018] divide a surface to be manufactured in a process akin to a Centroidal Voronoi Tessellation and compute a Hamiltonian path within to obtain a continuous trajectory. Other techniques use space-filling curves to define sparse infills with density gradients (e.g., [Kuipers et al. 2019]); however, our focus in this work is on dense, solid infills.

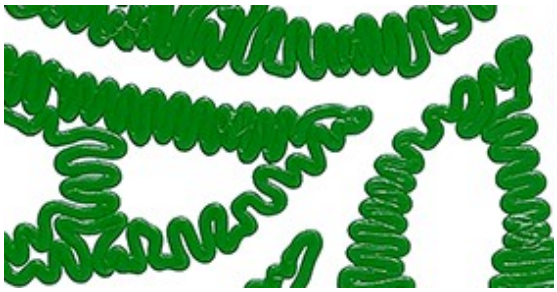


Fig. 2. Note how the cycle locally has a single trajectory around the holes of this surface. Such cycles cannot be obtained by growing from a small circle to invade the surface: this always gives an even number of paths in between outer contours.

One limitation of the aforementioned techniques for AM is that they offer no direct control over the trajectories' orientation. Several recent works note that aligning the trajectories with the principal stress directions improves the mechanical robustness [Fang et al. 2020; Kubalak et al. 2019; Liu and Yu 2017; Steuben et al. 2016; Tam and Mueller 2017; Xia et al. 2020]. Our technique provides both a continuous, close and dense space-filling trajectory *as well as* the ability to locally follow specific orientations.

We manufacture colored plates by progressively grading material mixtures (filament in our case) along the deposition trajectory. For this application, ensuring that the deposition curve stays in areas of similar composition for as long as possible is important for the final quality [Giachini et al. 2020]. This is due to the inertia in the physical material mixture process. We refer to this property as *zoning*. Researchers in AM considered this problem: Muller et al. [2014] and Xiang et al.[2018] grow paths in a grid under a greedy strategy to produce paths along mixture gradients.

3 METHOD

Our algorithm takes as input a polygon $\mathcal{P} \subset \mathbb{R}^2$, a distance d that (indirectly) controls the spacing around the curve, an orientation field O, a set of isotropic zones $\mathcal{I}_i \subset \mathcal{P}$, a set of anisotropic zones $\mathcal{A}_i \subset \mathcal{P}$, and a partition of $\mathcal{P}$ into disjoint material zones $\mathcal{Z}_i$. $\mathcal{P}$ and all the zones are represented as sets of oriented boundary contours

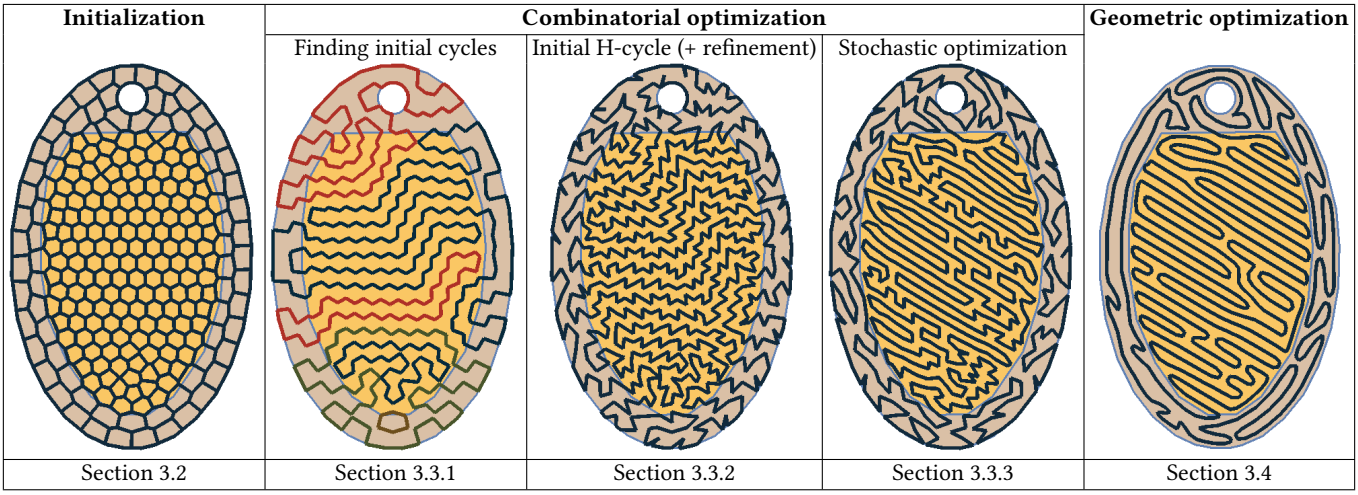


Fig. 3. General overview of our approach

and may have holes. $\mathcal{P}$ specifies the area to be covered. Without loss of generality, we assume that $\mathcal{P}$ is a connected set, as otherwise, we process each connected component independently. The orientation field O is piecewise constant. Its underlying data structure is a regular grid V covering $\mathcal{P}$, with center vectors having a norm of one — where an alignment is specified — or zero — where no specific alignment is requested. Each vector defines a constant orientation across the square grid cell area. Isotropic and anisotropic zones may overlap. In each of these zones, we seek to obtain respectively a uniform and a non-uniform orientation distribution. Each material zone $\mathcal{Z}_i$ is associated with a material that will be used during the printing process. We assume that two adjacent material zones have different materials (otherwise, they can be merged).

The algorithm outputs a smooth, non-intersecting closed trajectory $\mathcal{T}$ covering $\mathcal{P}$. $\mathcal{T}$ is composed of N vertices $(\boldsymbol{v}_i)_{1\leqslant i\leqslant N}$ and N straight line segments between consecutive points. We assume that the trajectory is oriented such that each vertex of index i has a successor i_+ and a predecessor i_-. A per-segment area $A(\boldsymbol{v}_i, \boldsymbol{v}_{i_+})$ is defined for each segment; it represents the local spacing, e.g., how much material to extrude when solidifying the bead.

We optimize the trajectory $\mathcal{T}$ to achieve a space-filling property with an interspace close to a nominal value. Intuitively, the trajectory meanders throughout $\mathcal{P}$ while covering it uniformly. Formally, this is achieved by minimizing the distance from any point of $\mathcal{P}$ to $\mathcal{T}$ (coverage) while constraining $\mathcal{T}$ to have a specific length. The length indirectly controls the nominal spacing around the curve, as given more length, the curve will everywhere get closer to points in $\mathcal{P}$.

We further incorporate the *alignment*, *orientation*, and *zoning* objectives. The alignment objective is specified as non-zero vectors in the field O and encourages *cycle* to follow these directions where specified. The orientation objective controls the edge orientations' distribution, minimizing or maximizing the distance to a uniform distribution. This optimizes for either isotropy or anisotropy. Zoning minimizes the number of crossings between $\mathcal{T}$ and the boundary between adjacent polygons $\mathcal{Z}_i$,$\mathcal{Z}_j$. All these objectives can be freely combined; the orientation objective is independently controlled in every zone. We detail all objectives and their computations in Section 3.1 and the Appendix.

Our approach operates in three main steps (see Figure 3):

- (Section 3.2) We build a graph embedded in $\mathcal{P}$, so that the nodes are evenly spaced, have exactly degree three, and so that the edge orientations are biased towards any local alignment specified in the input.
- (Section 3.3) After building an initial Hamiltonian cycle in the graph, we perform a stochastic combinatorial optimization of the alignment, orientation, and zoning objectives.
- (Section 3.4) We further optimize its geometry for even coverage, smoothness, orientation, and zoning, starting from the resulting space-filling curve while targeting a specific curve length.

3.1 Objectives

We formulate our objective functions along the segments of a Hamiltonian cycle $\mathcal{T}$ (H-cycle for brevity) covering $\mathcal{P}$.

The objectives we define here are used by both the combinatorial optimizer and the continuous geometry optimizer. They are combined in a weighted sum, always using the same weights.

In the following, $\mathcal{E}$ denotes the objective terms, and γ denotes the weights of the objective terms. We start by giving an intuitive definition of each objective.

- The *space-filling* objective produces a space-filling curve within the surface of $\mathcal{P}$. It comprises the $\mathcal{E}_{\text{CVT}}$ and $\mathcal{E}_{\text{Len}}$ objectives.
- $\mathcal{E}_{\text{CVT}}$ is the Centroidal Voronoi Tesselation objective induced by the segments of $\mathcal{T}$, restricted to the polygon $\mathcal{P}$. It encourages $\mathcal{T}$ to be as close as possible to every point of $\mathcal{P}$, maximizing the surface coverage.
- $\mathcal{E}_{\text{Len}}$ controls the length L of $\mathcal{T}$. This indirectly controls the spacing between the folds of $\mathcal{T}$, as given more length, the $\mathcal{E}_{\text{CVT}}$ will redistribute the curve to be close to all points of $\mathcal{P}$.

- $\mathcal{E}_{\text{Lap}}$ encourages $\mathcal{T}$ to be smooth; it is a Laplacian term applied to the vertices of $\mathcal{T}$.
- $\mathcal{E}_{\text{Ali}}$ aligns $\mathcal{T}$ with the input vector field O that specifies alignment vectors (vectors with unit norm, while zero vectors indicate that no specific alignment is requested).
- $\mathcal{E}_{\text{Ori}}$ controls the distribution of edge orientations, either maximizing (anisotropy) or minimizing (isotropy) the distance to a uniform distribution.
- $\mathcal{E}_{\text{Zon}}$ discourages edges from crossing between zones using different materials.

We optimize with two different approaches, one combinatorial and the other geometric. The combinatorial optimizer chooses which edges $\mathcal{T}$ uses in a fixed graph obtained from $\mathcal{E}_{\text{CVT}}, \mathcal{E}_{\text{Len}}$. It optimizes for $\mathcal{E}_{\text{Ali}}, \mathcal{E}_{\text{Ori}}, \mathcal{E}_{\text{Zon}}$. In this optimizer, the multi-objective equation parameterized by the cycle $\mathcal{T}$ is:

$$\mathcal{E}_{\text{Comb}}(\mathcal{T}) = \mathcal{E}_{\text{Zon}} + \gamma_{\text{Comb}} \left(\mathcal{E}_{\text{Ali}} + \mathcal{E}_{\text{Ori}}\right) \tag{1}$$

The geometric optimizer takes this output as initialization and further optimizes $\mathcal{T}$ for $\mathcal{E}_{\text{CVT}}, \mathcal{E}_{\text{Len}}, \mathcal{E}_{\text{Lap}}, \mathcal{E}_{\text{Ali}}, \mathcal{E}_{\text{Ori}}$, while trying to keep $\mathcal{E}_{\text{Zon}}$ unchanged. It directly manipulates the vertices and segments of $\mathcal{T}$. The corresponding multi-objective equation is:

$$\mathcal{E}_{\text{Geo}}(\mathcal{T}) = \mathcal{E}_{\text{CVT}} + \gamma_{\text{Lap}}\mathcal{E}_{\text{Lap}} + \gamma_{\text{Len}}\mathcal{E}_{\text{Len}} + \gamma_{\text{Obj}} \left(\mathcal{E}_{\text{Ali}} + \mathcal{E}_{\text{Ori}}\right) \tag{2}$$

Each optimizer performs the following optimization:

$$\min_{\mathcal{T}} \mathcal{E}(\mathcal{T}) \tag{3}$$

with $\mathcal{E}$ being either $\mathcal{E}_{\text{Comb}}$ or $\mathcal{E}_{\text{Geo}}$.

A detailed definition of each optimization term and their weights is provided in the Appendix, where we also derive the objective gradients, as these are required for the geometric optimizer.

There are several original aspects in the way we formulate our objectives. First, we formulate $\mathcal{E}_{\text{CVT}}$ on the segments of $\mathcal{T}$. This results in robust and precise optimization, naturally avoiding self-intersections without requiring a point resampling of $\mathcal{T}$ at every solver iteration. Equipped with the gradient, we optimize $\mathcal{E}_{\text{CVT}}$ with the quasi Newtonian solver L-BFGS [Liu and Nocedal 1989].

Second, we formulate the orientation objective in terms of the distribution of angles along $\mathcal{T}$. By minimizing the distance to a uniform distribution, we obtain isotropic trajectories; by maximizing it, we encourage anisotropy – without choosing an a priori orientation. In the geometric solver context, this leads to an optimal transport problem through the Wasserstein distance [Villani 2003].

Third, we consider the alignment objective by discrete integration of the target directions over the cells of the diagram of segments. This is achieved precisely and efficiently through a pre-computed table.

The objectives maintain their formulations in both the combinatorial and geometric optimizers.

3.2 Covering the surface with a graph

The first step of our approach is to construct a graph on which the combinatorial cycle optimizer will later operate. We seek to produce a planar graph embedded in $\mathcal{P}$. We know from [Gopi and Eppstein 2004] that a Hamiltonian cycle can be efficiently computed in a bridgeless graph of degree three, where bridgeless means that the graph does not have any edge which deletion would increase the number of connected components. Besides, we seek to obtain a graph with edges distributed evenly over $\mathcal{P}$.

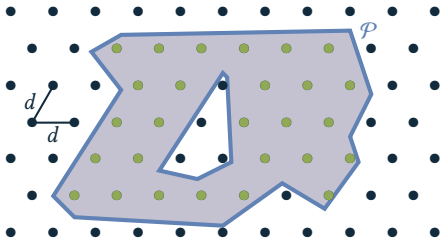


Fig. 4. Initial seeds (in green) are created by intersecting a triangular grid with $\mathcal{P}$. Every pair of adjacent nodes are at a distance d.

From these two requirements, we choose to create the graph from the vertices and edges of a Centroidal Voronoi Tesselation (CVT). We generate initial seeds by intersecting a triangular tiling with our polygon $\mathcal{P}$ (see Figure 4). The spacing d between adjacent vertices parameterizes the tiling. Each seed position is perturbed by a random offset, sampled in a disc of radius $\frac{d}{20}$. Tiling vertices inside $\mathcal{P}$ will be the initial seeds of our Voronoi diagram.

The spacing d is the driving parameter for the curve interspace in the final result: with a lower d, the number of seeds S increases, producing a longer cycle, which leads to a smaller curve interspace. However, choosing d to obtain a specific interspace is not simple, as this also depends on the polygon $\mathcal{P}$ and the objectives we are minimizing (see Figure 15). In practice, d is adjusted empirically by restarting the process with a lower d when the interspace is too high and with a higher d when the interspace is too low. After the Voronoi tesselation, the graph is composed of the Voronoi edges of the diagram, clipped by the polygonal contour whose boundary is also part of the graph, see Figure 5. A CVT graph is extremely likely to have nodes of degree three everywhere, as co-circular cases are extremely rare – but nodes of degree higher than three may exist due to numerical precision limitations. In this unlikely event, we restart the process with a different initialization. The graph is bridgeless by construction, as each edge belongs to a cyclic cell.

Directly using a point CVT graph would, however, yield poor results under alignment objectives. Indeed, during the combinatorial optimization, we select which edges belong to the cycle. If the user controls alignments through the vector field O, we can only hope to find a cycle with aligned edges if they indeed exist in the graph. Therefore, the graph has to ideally contain some edges aligned along the non-zero vectors of O.

To align the edges, we introduce anisotropy in the Voronoi diagram. Instead of casting this as a metric distortion problem [Lévy and Bonneel 2013], we consider a diagram of many small segments – called *needles*. The lengths and orientations of the needles induce cells that are elongated and oriented with respect to the control field O, see Figure 5. Each needle is centered on a supporting seed. The needle lengths and orientations are implicitly defined by the (point) Voronoi diagram of their supporting seeds. Let us consider the needle centered on a seed p, whose Voronoi cell we denote as C_p.

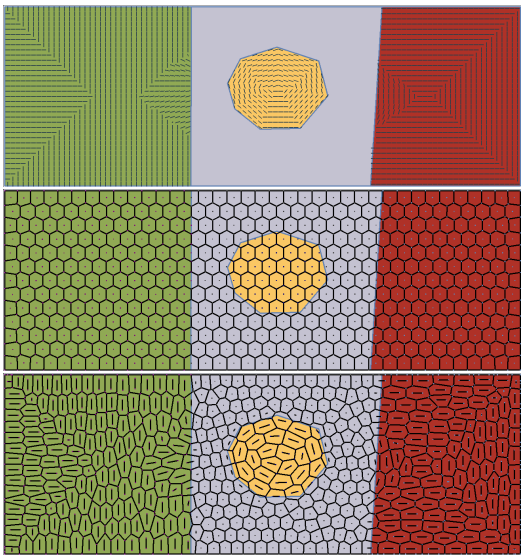


Fig. 5. Obtaining aligned edges with the needles. **Top:** vector field: the green, yellow, and red zones have alignment vectors specified. **Middle:** standard point Voronoi tessellation. **Bottom:** Voronoi tessellation obtained using needles. Many of the diagram edges now follow the input vector field.

The input vector field O specifies desired directional alignments as unit vectors (recall that the field may contain zero vectors where no alignment is specified). Since we only take into account the orientation – the vector angle modulo π – we consider a vector field V obtained by doubling the angle of O:

$$V = \|O\| \left(\cos\left(2\angle O\right) \quad \sin\left(2\angle O\right)\right) \tag{4}$$

where $\angle O$ is the angle of O.

We orient the needle using the mean of V inside the cell $v = \mathcal{V}_p/\mathrm{area}(C_p)$, where $\mathrm{area}(.)$ computes the area of a polygon, and $\mathcal{V}_p$ is the integral of V over the cell as defined in equation (36). Note that $||\mathrm{u}_p|| \leq 1$. The needle length is computed as $\mathrm{L}_p = ||\mathrm{u}_p|| \cdot \sqrt{\frac{\mathrm{area}(\mathcal{P})}{\pi S}}$, where the square root term is a global scaling factor based on the total area covered with S seeds. To ensure that needles never cross, L_p is clamped so that the needle is fully contained in C_p. Finally, the angle of the needle is given by $\frac{1}{2}\angle \mathrm{u}_p$.

We optimize for the seeds' positions under a *segment* CVT objective, using the L-BFGS algorithm [Liu and Nocedal 1989]. We use the simplifying assumption that u_p undergoes only small changes at each optimization iteration (the integral $\mathcal{V}_p$ changes progressively at each iteration). Therefore, we only optimize seed positions, recomputing u_p, L_p at every iteration of the L-BFGS algorithm. The gradient of the segment CVT objective is computed at the needle extremities and used to compute the supporting seed positions' gradient. After convergence, we extract the graph as the edges of the segment Voronoi diagram and clip it with the boundary of $\mathcal{P}$. This gives us an initial graph satisfying our requirements, as illustrated in Figure 5.

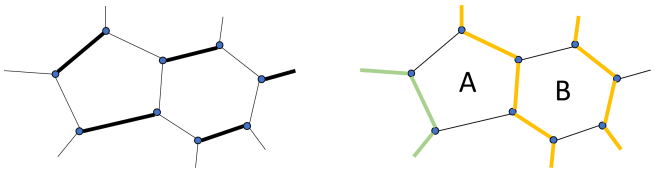


Fig. 6. Closeup on two cells of the three connected, bridgeless planar graph. **Left:** a perfect matching. **Right:** inverting the perfect matching creates cycles covering all nodes within the graph. Here two cycles are produced from which we see only a few edges (in green and orange).

3.3 Combinatorial optimization

Once a bridgeless planar graph with nodes of degree three is obtained, we proceed with the combinatorial cycle optimization. Our combinatorial optimizer's core principle is to start from a valid H-cycle and perform local operations on its edges, modifying the cycle while preserving its Hamiltonian property. These local changes allow us to perform an efficient stochastic exploration, optimizing the alignment, orientation, and zoning objectives.

The use of local operations is inspired by recent graph theory results that reconfigure H-cycles in regular grids [Nishat 2020; Nishat and Whitesides 2017]. We introduce a different set of operations as our graphs are irregular and use them in a stochastic optimization framework (Section 3.3.3).

The optimizer proceeds in three main steps. The first obtains a set of cycles fully covering the input graph but not yet connected in a global H-cycle (Section 3.3.1). For this, we rely on existing techniques. The second operates a graph refinement and reconnects the cycles into a single Hamiltonian cycle using local operations (Section 3.3.2). The graph refinement is a key step that significantly enriches the local operations performed on cycles in general and H-cycles in particular. The third is the optimization process (Section 3.3.3).

3.3.1 Finding initial cycles. As a starting point, we follow the principles established in Gopi and Eppstein [2004]. We compute a *perfect matching* on the graph [Edmonds 1965]: a selection of edges such that every node is adjacent to exactly one selected edge. This can be done in linear time on bridgeless planar graphs of degree three [Biedl et al. 2001]. Then, by inverting this edge selection, a set of cycles covering all graph nodes is obtained. However, these cycles are not connected in a single Hamiltonian cycle, see Figure 6.

Gopi and Eppstein [2004] reconnect the cycles through two strategies. The first is to identify specific configurations of edges allowing to split and reconnect neighboring cycles – however, such configurations are relatively rare in three-connected planar graphs. The second is to create double edges connecting cycles, which is always possible but locally creates a strong deviation from the even spatial distribution of edges we seek to achieve.

3.3.2 Graph refinement and initial H-cycle. Our combinatorial optimizer relies on local modifications of a Hamiltonian cycle. We follow a similar approach to reconnect the initial cycles into a single H-cycle. Unfortunately, the low connectivity and mostly hexagonal faces of the three-connected planar graph make it difficult to

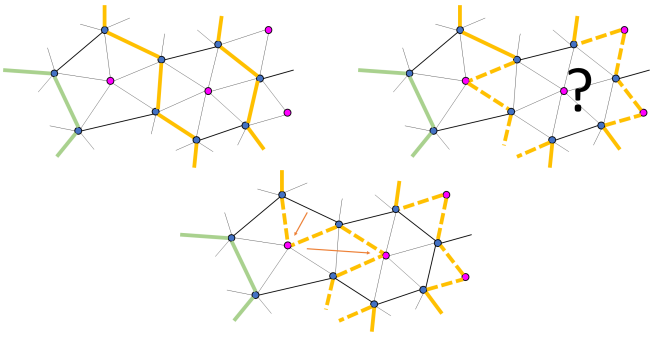


Fig. 7. **Top left:** inserting a new node in each cell produces triangular faces. **Top right:** the cycles are modified to go through the center node. Here cell B no longer has a choice. **Bottom:** B stole the edge from A, cascading in an update in A.

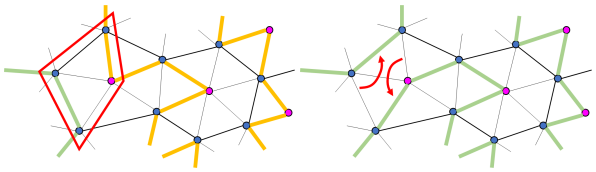


Fig. 8. Cycles are connected through split-and-reconnect configurations. **Left:** an opportunity to split-and-reconnect highlighted in red. **Right:** after swapping the edges, the two cycles are joined together.

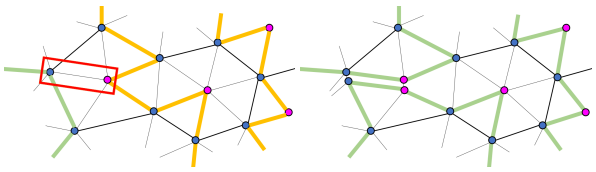


Fig. 9. Cycles are connected through edge doubling. **Left:** Un unused edge between two cycles in red. **Right:** after doubling the edge and its endpoints, the two cycles are joined together.

perform simple modifications: any change impacts many edges at once.

Therefore, we propose to refine the graph, increasing the density of edges and thus making it easier to find opportunities for local, simpler modifications. In the following, we refer to the faces of the three-planar graph as *cells* as they originated from Voronoi cells. We insert one additional node at the barycenter of every cell, splitting it into triangular faces. This offers a much richer set of opportunities to perform local manipulations of cycles.

After introducing a center node and splitting the cells into triangular faces in the graph (Figure 7), we update the cycles obtained from perfect matching to end up in a similar situation: a set of cycles covering all nodes.

We examine each cell in turn. In each cell, we choose one of the boundary edges (a, b) that is part of the cycle. This edge is then split to go through the added center node n as $(a, n) \rightarrow (n, b)$. The cycle now goes through n in addition to (a, b); see dashed lines in Figure 7, top right.

Note that by construction, all cells have at least two cycle edges along their boundary. Indeed, we started from a perfect matching (Section 3.3.1) and inverted the selection. This ensures that every node is connected to two cycle edges and a third edge not in the cycle. Therefore, we cannot have two successive edges that do not belong to the cycle. Consequently, a cell made up of k edges has at least $\lceil k/2 \rceil$ of them in a cycle. Since a cell is made up of at least 3 edges, this implies that all cells have at least two edges in a cycle.

As we connect boundary cycle edges to cell centers, we may end up with a cell that no longer has any cycle edge along its boundary, making the new node n unreachable (Figure 7, top right). When this happens, the cell steals this edge back from its neighbor, and the cell is tagged as *frozen*: no other cell can steal this edge from it. The neighbor now needs to connect its center again. It will either use another available boundary cycle edge or will have to steal it from another neighbor. The process propagates until all cases are resolved. This is illustrated in Figure 7. In all our experiments, the process terminates successfully, with minimal propagation.

Finally, once the new graph and cycles are obtained, we search for *split-and-reconnect* configurations to merge the cycles, as illustrated in Figure 8: an edge from a different cycle across two free edges. The much more densely connected graph provides a larger number of reconnection opportunities. The algorithm visits all cycle edges, searching for the candidate configuration. As soon as a split-and-reconnect operation is discovered, it is applied, joining the two cycles. The algorithm continues until no further *split-and-reconnect* operation is possible.

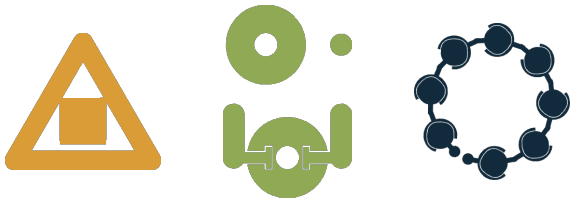
At this stage, if multiple cycles remain, double edges are added, as illustrated in Figure 9. The algorithm goes along the cycles again, searching for the first free edge connecting another cycle and doubles it. Typically, only very few double edges are necessary, as experimentally verified in Table 1.

At this point, we have a valid initial H-cycle. This entire process is efficient: perfect matching is linear in the number of nodes, the cycle reconnection linear in the number of edges.

3.3.3 *Stochastic H-cycle optimization.*

Objectives. Given a H-cycle, we evaluate the objectives $\mathcal{E}_{\text{Ali}}, \mathcal{E}_{\text{Ori}}$ and $\mathcal{E}_{\text{Zon}}$ along the edge that belongs to the cycle. Please refer to Section 3.1 for their exact computations. However, some differences have to be clarified. For the alignment objective, as it would be very costly to compute a Voronoi diagram each time we update the score, the vector field integrals are no longer computed inside segment Voronoi cells but inside small rectangles surrounding edges. For the orientation objective, we track the angle distribution in a histogram having n bins where n is set to 100 in our implementation. This histogram affords for computing the objective value in constant complexity. The combinatorial optimization does not consider the *surface-filling* objectives ($\mathcal{E}_{\text{CVT}}, \mathcal{E}_{\text{Len}}, \mathcal{E}_{\text{Lap}}$). The vertices are fixed, and the surface filling property is enforced by construction of the initial graph and the Hamiltonian nature of the cycle.

We update all objectives' current values as we locally modify the cycle by removing/adding edges. The objectives are weighted and combined as discussed in Section 3.1.



hourglass android balljoint

Model	With refinement		Without refinement	
	s.a.r	db. edges	*s.a.r*	db. edges
hourglass	1323	0	558	768
android	3589	16	1421	2186
balljoint	1552	47	649	955

Table 1. Comparison of the number of different operations used to reconnect cycles — either *split-and-reconnect* (*s.a.r*) or edge doubling — when graph refinement is used or not. Models contain between 100 and 200 layers. Statistics are aggregated over all of them.

Evolving an H-cycle: local operations. The local operations we perform on the H-cycle are shown in Figure 10. In each operation template, there is a starting configuration and an updated configuration. The starting configuration considers a set of nodes with a specific configuration of edges. The green thick edges are those that belong to the cycle; the black thin edges are those that do not currently belong to the cycle. Other edges may exist between the nodes and may belong to the cycle or not – these do not influence the operation.

Given a configuration that matches the template starting configuration, we can, in each case, modify the edges into the updated configuration *while preserving the H-cycle*. Each operation modifies the set of edges belonging to the cycle and thus the value of the objectives. We search for configurations by selecting a node at random and then considering the edges surrounding it. Matching the templates can be achieved in constant time for all operations but one (rightmost in Figure 10). A global connectivity check is required between two of the nodes, indicated in Figure 10 by a dashed green curve. We perform it in constant time using an auxiliary data-structure. However, this data structure requires a linear time update after each actual modification of the graph – these only happen when a local change is accepted, typically when it improves the objective function's value.

Optimization loop. Our algorithm follows a standard stochastic optimization framework. Starting from an initial H-cycle, we produce a population of $K \times C$ cycles, cloning the initial one. Each of these H-cycles is evolved with the *shuffle* algorithm shown in Figure 11.

We next select K champions from the $K \times C$ cycles, promoting diversity. We measure the distance between two H-cycles as the number of edges they do not have in common. We initialize the set of champions with the H-cycle having the best objective. Then we iteratively add the H-cycles that are the most distant from the already selected champions (Hochbaum-Shmoys algorithm [1985]), until K are selected.

We then enter the main optimization loop, where champions are iteratively improved. At each iteration, each champion is cloned into C candidates evolved with Algorithm 11. The champion is replaced by the best of its candidates, but only if it improves the objective compared to the champion. This is performed for a fixed number of iterations I. In practice, we use $C = 2, K = 12, I = 3$, `ShuffleIter`$=8N$, `ImproveIter`$=8N$, `ThreshHigh`$=0.5$, `ThreshLow`$=1$ and `Mod`$=400$. Here, N is the number of nodes inside the cycle. The algorithm uses some randomness through the function `rand()` that returns a uniform number between 0 and 1. Finally, this is easily parallelized; we use one thread per champion.

3.4 Cycle geometry optimization

The input of the geometry optimization is the H-cycle $\mathcal{T}$ obtained by the combinatorial optimizer. By construction, it is a non-intersecting closed polyline, already forming a surface-filling cycle. Our goal is now to optimize the position of the vertices of $\mathcal{T}$ in order to obtain smooth trajectories evenly distributed while following all our other objectives (alignment, orientation, zoning).

Initialization. An important question for the geometric optimizer is which length to target for the cycle with $\mathcal{E}_{\text{Len}}$. Let us recall that the total length indirectly controls the final spacing between folds of the trajectory.

The cycle produced by the combinatorial optimizer has a global directionality in agreement with the alignment and orientation objectives. However, it also has many local corners (high frequencies) coming from the initial tesselation. Now that the cycle connectivity is determined, the geometric optimizer can smooth out these higher frequencies. The smoothing objective $\mathcal{E}_{\text{Lap}}$ achieves this.

However, the high frequencies coming out of the combinatorial optimizer artificially lengthens the cycle. Therefore, before computing the target length, we apply a few explicit smoothing steps. This results in a filtered cycle whose length is more representative of our spacing objective.

We apply 5 times the following smoothing operator to each vertex $\boldsymbol{v}_i$, while ensuring that no self-intersections appear:

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \frac{1}{10}\left(\boldsymbol{v}_{i_+} + \boldsymbol{v}_{i_-} - 2\boldsymbol{v}_i\right) \tag{5}$$

We then define the desired length L_0 as the length of $\mathcal{T}$ after applying these operations. The geometric optimizer starts from this smoothed cycle.

Optimization. We then minimize the geometric objective $\mathcal{E}_{\text{Geo}}$ defined in (2). We sincerely encourage you to have a look at the different terms of this objective in appendix. Our geometric optimizer is supported by a quasi Newtonian solver L-BFGS [Liu and Nocedal 1989]. This is an iterative solver that only needs to compute the value of the objective and the gradient or at least an approximation of it to minimize the objective. The solver then approximates the Hessian using the gradient of the last M iterations. M is fixed before optimization and has a value generally less than 10; we use 7 in our implementation.

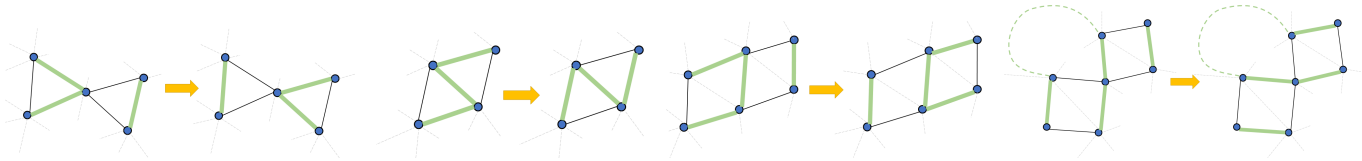


Fig. 10. The four local operations evolving a H-cycle while preserving its Hamiltonian property. Thick green edges belong to the cycle; solid black edges do not belong to the cycle, the state of other edges (gray dashed) has no influence. Note that the bottom-right operation requires knowing connectivity along an entire path, as indicated by the dashed green curve.

```
algorithm Shuffle(HCycle 𝒯) {
  for (shuffle = 0 to ShuffleIter) {
    𝒯′ = applyRandomOp(𝒯)
    if (ℰComb(𝒯′) <= ℰComb(𝒯)) {
      𝒯 = 𝒯′
    } else if (ℰComb(𝒯′) < ℰComb(𝒯) + ThreshHigh)) {
      if (N * (ℰComb(𝒯′) - ℰComb(𝒯)) * rand() < ThreshLow)
        𝒯 = 𝒯′
      } else if (shuffle % Mod == 0) {
        𝒯 = 𝒯′
      }
    }
  }
  for (improve = 0 to ImproveIter) {
    𝒯′ = applyRandomOp(𝒯)
    if (ℰComb(𝒯′) <= ℰComb(𝒯)) {
      𝒯 = 𝒯′
    }
  }
  return 𝒯;
}
```

Fig. 11. The shuffle algorithm applies local operators to evolve and explore input H-cycle improvements with respect to our objectives.

Computing the per-segment area. After the process has converged, we are ready to compute the per-segment area along $\mathcal{T}$. Once multiplied by a thickness, it represents the volume of material that has to be extruded around the segment. This allows the deposition to adapt to each segment's surroundings (see Section 4.2).

The area is computed from the segment's Voronoi cell, which is added half the area of the point Voronoi cells of its extremities.

4 RESULTS

This section presents numerical evaluations and experimental results, including plates 3D-printed with multiple materials.

4.1 Numerical evaluations

Figure 13 shows the evolution of the two optimization stages (combinatorial and geometrical). As can be seen, both optimization solvers operate effectively to reduce the value of the objectives. For the geometrical optimization, $\gamma_{\text{Len}}\mathcal{E}_{\text{Len}}$ acts as a regularizer of the total length of $\mathcal{T}$, and the optimizer attempts to maintain its value.

Figure 14 demonstrates the effect of the Laplacian term γ_{Lap} (see Section A.3). While both results are compelling, using the Laplacian term avoids sharp turns, effectively smoothing the cycle.

Figure 15 shows the link between the parameter d controlling the initial spacing between adjacent seeds (see Section 3.2), and the interspace τ encountered along the cycle. The interspace τ is computed from the Voronoi cells, taking the distance between both sides across the trajectory.

The overall curve shows that choosing d to obtain a given interspace, while not direct, is made simple thanks to the well-behaved, monotonous increase. We can also observe that the variation across models is small (within ± 0.1 mm of each other's).

In addition, Figure 16 shows that the interspace values roughly follow a Gaussian distribution. This simplifies pushing a varying flow (by adjusting motion speed) during fabrication. Interestingly we observe a change in statistics between the objective. However, they all remain within small bounds, as their means are all within 0.05 mm of each other's.

Figure 12 shows some timing results. Both optimizers can execute iterations in a reasonable time, with a roughly polynomial correlation with respect to d.

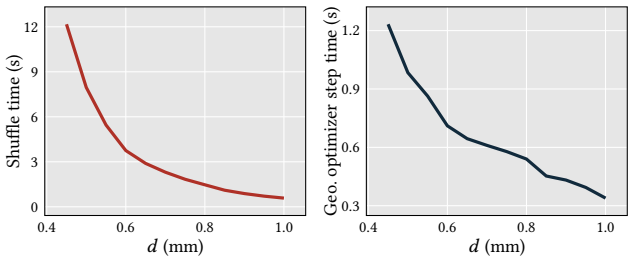


Fig. 12. Timings results of the brain model (see Figure 13). **Left**: computing time of the function `Shuffle` (Fig. 11) with respect to d. **Rght**: computing time of a single iteration of the L-BFGS solver in the geometric step.

4.2 3D printed results

We carried out 3D printing of surfaces from H-cycles optimized with our method. We employed filament fused fabrication with PLA.

Multi-material illustrations. We fabricated multi-material illustrations, producing plates with complex trajectory patterns. These prints were fabricated using a constant deposition width to better highlight the trajectory.

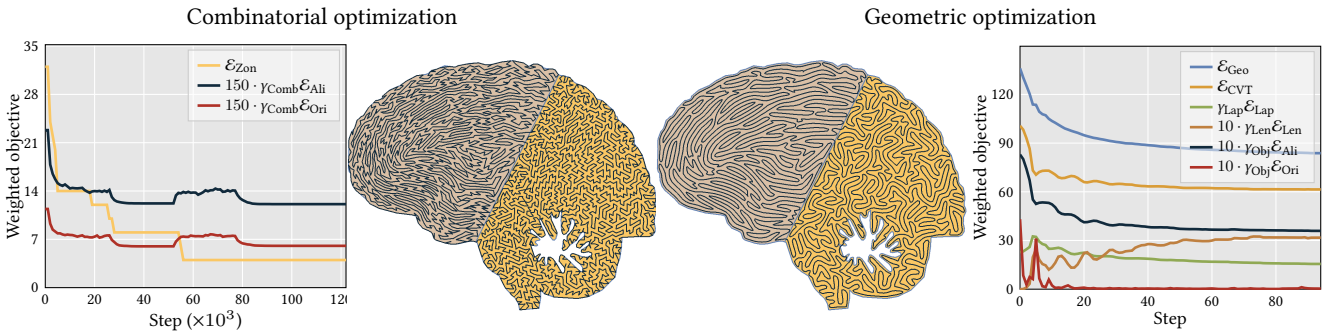


Fig. 13. Two-stage optimization of shape depicting a brain. The trajectory of the left part is under an alignment objective, while the trajectory in the right part follows an isotropy objective. The plots depict the evolution of the value of each objective function during the optimization steps. Note that some values are multiplied by a constant factor in order to improve the visualization.

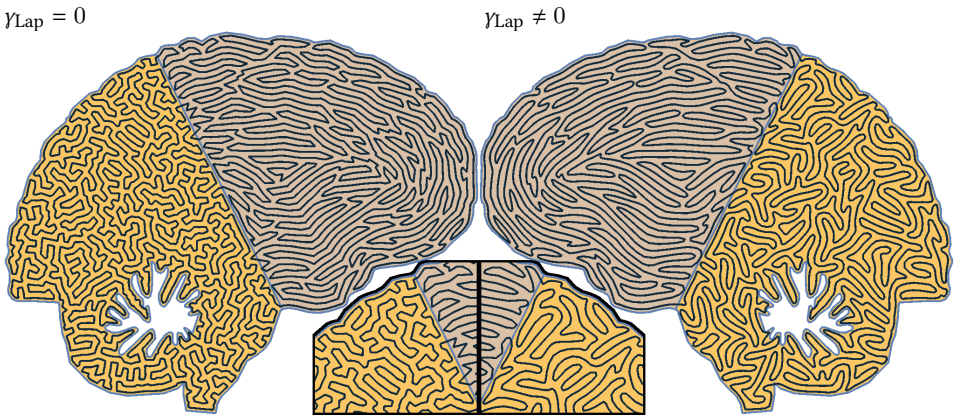


Fig. 14. Optimization results showing the effect of using or not the smoothing term γ_{Lap}.

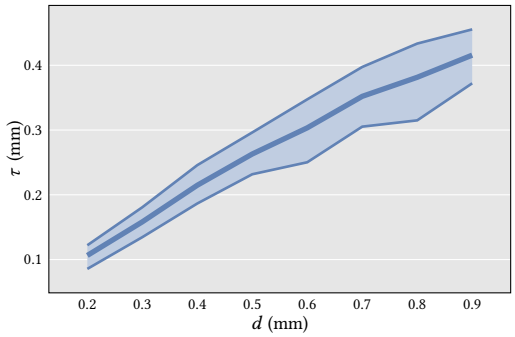


Fig. 15. Correlation between d (controlling the seed density – see Figure 4) and the interspace τ around the optimized cycle. The thick curve is the mean observed over 30 models using different objectives. The top and bottom curves are the reported min/max values of the mean across models.

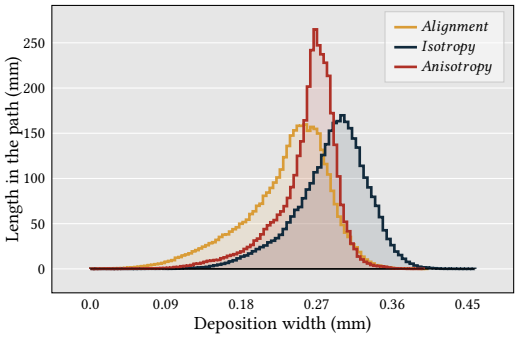


Fig. 16. Interspace distribution over the brain model with $d = 0.5mm$, for different objectives.

We produced multi-material prints using a 3-way *Diamond* nozzle by *Reprap.me*. This nozzle takes three different filaments as input, mixed into the nozzle chamber, and exit from a single hole. The mixture is passive, so the materials are co-extruded in a tooth-paste like pattern at a small scale. Our printer is a Prusa-like assembly, with three feeders for each of the three filaments. The printer firmware accepts special GCode instructions defining the mixing ratio at each extrusion move. In our experience transitioning from one mixture to another requires pushing at least $6.1mm^3$, resulting in a relatively long transition, e.g., 76 mm of deposition trajectory for a 0.4 mm nozzle at 0.2 layer height. In this context, the zoning and alignment objectives are important to respectively minimize the number of transitions and align them along zone boundaries. We printed at a speed of 10 mm/sec, using colored translucent PLA filament.

Figure 1 **(d)** shows a 3D printed SIGGRAPH logo using alignment along boundaries in the blue region, anisotropy in the white region,

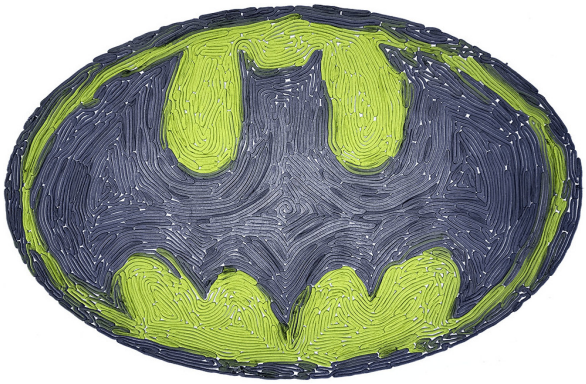


Fig. 17. Multi-material printing of the Batman logo following the parallel vector field.

and isotropy in the red region. Note how well the different zones are reproduced, with little smearing between colors.

Figure 19 shows the *brain* model 3D printed with a variety of materials (colors) under different objectives combined (see Figure caption for details). Two versions were printed with different spacings d. As can be seen, the 3D printed versions closely follow the input specifications, with clearly defined zones both in material and alignment/orientation objectives. The single continuous deposition produces an even, reliable flow of material, leading to nearly optimal operational conditions for the extrusion process.

While the *brain* model features relatively large zones, the *leaf* model shown printed in Figure 20 has many narrow regions, and an internal skeletal structure. This shows that despite being a single cycle, the trajectory can fill narrow regions even orthogonally to their main directionality – this would be very difficult to achieve with growth techniques that have to fit an even number of trajectories everywhere (see Section 2). The skeleton is well captured and faithfully reproduced, despite a challenging directionality for the multi-material transitions.

Figures 17 and 18 show other examples of shapes with narrow spaces and multi-material transitions. The *batman* logo is using an alignment objective parallel to the boundaries. The *zebra* is using anisotropy for the background and alignment along boundaries within the zebra stripes.

Dense plates. To evaluate how dense our prints can be, we compared two methods on an isotropic cycle filling a square. Isotropy is challenging due to the high number of turns. The first method uses our per-segment area computation, varying the amount of deposited material along the cycle. Note that the material's feed rate was kept constant (the speed of the motor pushing the filament) while the XY motion speed was varied.

The second method extrudes at a constant speed an average amount of material, computed from the plate's total area – this is equivalent to considering that the average 'width' along the cycle is the total area divided by cycle length.

As shown in Figure 21 the per-segment area achieves a denser result, leaving only rare gaps, even in very challenging areas of the leaf model. These examples are fabricated at 0.2 mm layer height, with two layers, using a 0.6 mm nozzle at an average speed of 20 mm/sec on a Prusa i3 printer. They are printed in sequence and at

Fig. 18. Zebra printed with black and white filaments.

the same bed location. Both use the same total volume of material: 281 mm^3 for the squares, 1955 mm^3 for the leaves. The average spacing around the cycle is approximately 1.3 mm.

Note that we printed at an increased temperature (215 degrees) to allow the molten plastic to spread more easily. The nozzle width also has to be within reasonable margins of the nominal spacing (obtained as total area divided by cycle length).

5 CONCLUSIONS

Limitations. In the presence of holes with sharp pointy features, the geometric optimizer may produce a trajectory that crosses the contour, as can be seen Figure 22. These defects are usually located at the tip of narrow, pointy holes, which limits their impact.

To obtain denser plates, our approach requires printing while controlling motion speed for a varying flow deposition (as has been successfully demonstrated in recent works [Hornus et al. 2020]). However, some tiny porosities remain in challenging cases (see the leaf in Figure 21).

Future work. Our technique offers unprecedented control over material deposition orientation while manufacturing a solid layer along a strictly continuous extrusion trajectory. This is of particular interest for multi-material prints and aesthetics purposes, as we have demonstrated. Continuity is also an important consideration for materials such as clay, ceramics, silicon, and concrete. Beyond what we demonstrated here, our framework supports a wide range of optimization objectives. We believe this will prove useful for Additive Manufacturing processes using challenging materials, such as the fabrication of plates having close to isotropic elastic behaviors [Lin et al. 2019] (thanks to our isotropy objective term $\mathcal{E}_{\text{Ori}}$). To foster adoption and further research, we will release our code as open-source and are planning to submit it to the Graphics Replicability Stamp Initiative.

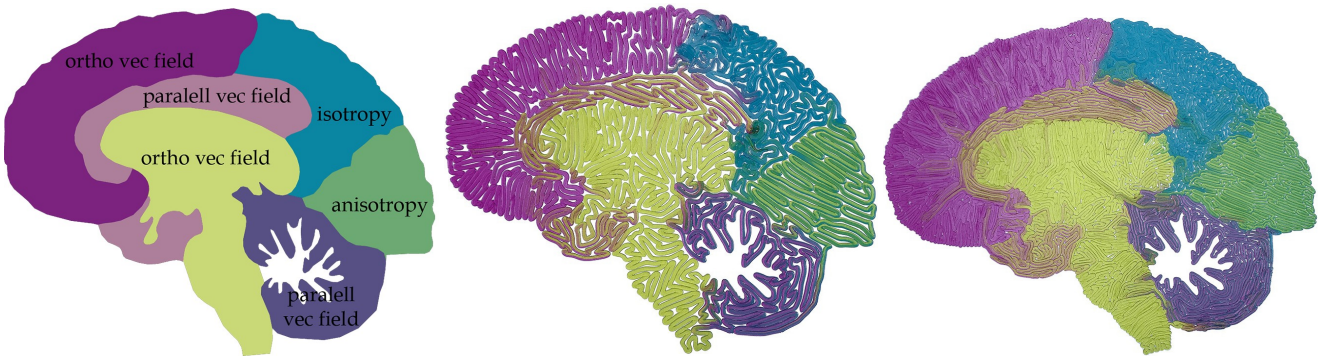


Fig. 19. **Left:** input specifying zones and objective selection in each. *Parallel vec field* means that we align the path with a vector field parallel to the zone border and *ortho vec field* means that we use alignment with a vector field orthogonal to the border. **Middle:** cycle optimized for a width of $1.7mm$. **Right:** cycle optimized for a width of $0.8mm$.

Fig. 20. **Left:** Single material leaf printed with a constant width smaller than nominal to appreciate the cycle folds. Note how the cycle goes around holes with a single folded trajectory in the center part. **Middle:** Multi-material version using a narrower spacing, with the inner skeleton in a different material. **Right:** Variant using different colors.

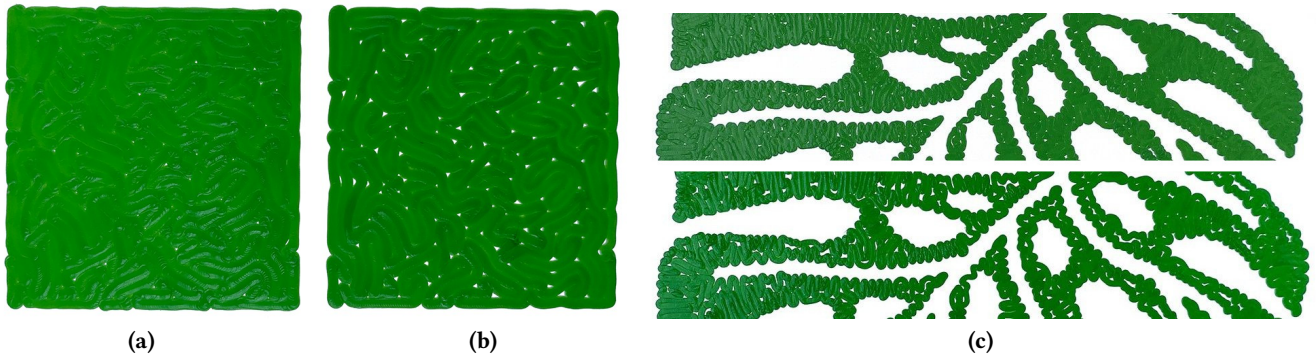


Fig. 21. **(a):** plate printed using the per-segment area. **(b):** plate printed with a constant flow all along the cycle. Both use the same total volume of filament, are printed at the same bed location. **(c) top:** closeup on the leaf model shown in Figure 20. **(c) bottom:** closeup on the same leaf printed without the per-segment area.

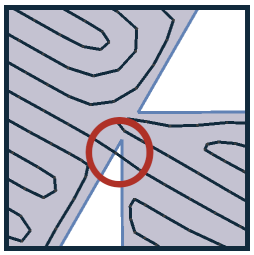


Fig. 22. An edge may cross the contour near sharp corners after the geometric optimization.

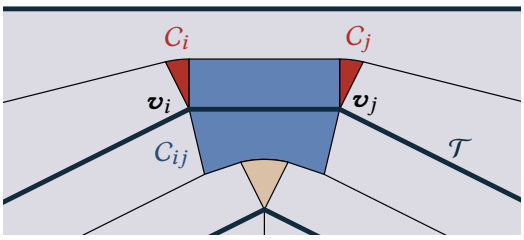


Fig. 23. Illustration of a segment cell in blue and two point cells in red.

A APPENDIX

Recall that $\mathcal{T}$ is composed of N vertices $(\boldsymbol{v}_i)_{1\leqslant i\leqslant N}$ and N straight line segments between consecutive points. Each vertex of index i has a successor i_+ and a predecessor i_-. We denote by $\mathcal{T}_E$ the set of edges of the path:

$$\mathcal{T}_E = \{(i, i_+) \mid i = 1, ..., N\} \tag{6}$$

A.1 CVT objective

We perform a CVT optimization of a diagram of segments. Let $x \in \mathbb{R}^2$ be a point and $S \subset \mathbb{R}^2$ be a bounded set. The infimum Euclidean distance between x and S is:

$$\operatorname{dist}(x, S) = \inf_{y\in S} \|x - y\| \tag{7}$$

In order to cover the entire polygon $\mathcal{P}$ without keeping empty areas, one could consider minimizing the following objective:

$$\int_{\mathcal{P}} \operatorname{dist}(x, \mathcal{T})^2 \, dx \tag{8}$$

Minimizing the above objective could result in new crossings between the cycle and material zone boundaries, undermining the zoning objective. To avoid this issue, we minimize the *CVT* objective independently for each material zone:

$$\mathcal{E}_{\text{CVT}} = \sum_i \int_{\mathcal{Z}_i} \operatorname{dist}(x, \mathcal{T} \cap \mathcal{Z}_i)^2 \, dx \tag{9}$$

Computing the intersection of $\mathcal{T}$ with $\mathcal{Z}_i$ leads to the creation of new vertices at the intersections between segments of $\mathcal{T}$ and zone boundaries. Consider an intersecting vertex u laying inside a segment $]\boldsymbol{v}_i, \boldsymbol{v}_j[$. Then there exists a real value $t \in]0, 1[$ such that:

$$u = (1 - t) \cdot \boldsymbol{v}_i + t \cdot \boldsymbol{v}_j \tag{10}$$

To obtain the gradient with respect to the initial path's vertices, we will have to make the following update at the end of our computations:

$$\mathcal{G}^{(i)}_{\text{CVT}} \leftarrow \mathcal{G}^{(i)}_{\text{CVT}} + (1-t) \cdot \mathcal{G}^{(u)}_{\text{CVT}} \qquad \mathcal{G}^{(j)}_{\text{CVT}} \leftarrow \mathcal{G}^{(i)}_{\text{CVT}} + t \cdot \mathcal{G}^{(u)}_{\text{CVT}} \tag{11}$$

where $\mathcal{G}^{(i)}_{\text{CVT}}$, $\mathcal{G}^{(j)}_{\text{CVT}}$, and $\mathcal{G}^{(u)}_{\text{CVT}}$ are respectively the gradients of $\mathcal{E}_{\text{CVT}}$ with respect to $\boldsymbol{v}_i$, $\boldsymbol{v}_j$, and u.

In the following, we consider computing the objective and its gradient inside a single zone. For simplicity, and to not introduce new notations, we suppose the zone to be $\mathcal{P}$, and we still use the notation $\mathcal{T}$ for the intersection of the path inside the zone. Notice that $\mathcal{T}$ could no longer be a cycle but a set of paths having two endpoints. In the same way, we still denote by $\boldsymbol{v}_i$ the path's vertices, including the ones that intersect the boundary of the zone.

The segment Voronoi diagram over $\mathcal{T}$ gives the following partition of $\mathcal{P}$ into two types of Voronoi cells:

- **Point cell** A cell C_i for each vertex $\boldsymbol{v}_i$ of $\mathcal{T}$, containing all points of the shape $\mathcal{P}$ that are nearest to $\boldsymbol{v}_i$ than to any other point of $\mathcal{T}$:

$$C_i = \{x \in \mathcal{P} \mid \forall y \in \mathcal{T},\ \|x - \boldsymbol{v}_i\| \leqslant \|x - y\|\} \tag{12}$$

- **Segment cell** A cell C_{ij} for each open segment $]\boldsymbol{v}_i, \boldsymbol{v}_j[$ of $\mathcal{T}$ that contains all points of the shape $\mathcal{P}$ that are nearest to this segment than to any other point of $\mathcal{T}$:

$$C_{ij} = \left\{x \in \mathcal{P} \mid \forall y \in \mathcal{T},\ \operatorname{dist}\left(x,]\boldsymbol{v}_i, \boldsymbol{v}_j[\right) \leqslant \|x - y\|\right\} \tag{13}$$

This partition is illustrated Figure 23. Once this partition is obtained, the distance to the path can be obtained as a close form inside each cell. The CVT objective is rewritten as:

$$\begin{aligned}\mathcal{E}_{\text{CVT}} &= \sum_{i=1}^{N} \int_{C_i} \|x - \boldsymbol{v}_i\|^2 dx + \sum_{(ij)\in\mathcal{T}_E} \int_{C_{ij}} \operatorname{dist}\left(x,]\boldsymbol{v}_i, \boldsymbol{v}_j[\right)^2 dx \\ &= \sum_{i=1}^{N} \mathcal{E}_{\text{CVT},i} + \sum_{(ij)\in\mathcal{T}_E} \mathcal{E}_{\text{CVT},(ij)}\end{aligned} \tag{14}$$

The boundary of a Voronoi cell consists of bisectors sections between two sites (which can be a vertex or an open segment). All the bisectors are straight lines apart from the parabolic bisectors between a vertex and an open segment. In order to simplify computations, we approximate the section of a parabola with k straight line segments. In our implementation, we choose $k = 10$, which provides a fine tesselation.

In the following, we precise how we derive the objective and gradient terms. From now on, C_i and C_{ij} are assumed to be composed only of straight-line segments due to the parabolic bisector sections' approximation.

Point cell. We want to derive the objective $\mathcal{E}_{\text{CVT},i}$ associated with one point cell C_i. First, we rewrite it as:

$$\begin{aligned}&\mathcal{E}_{\text{CVT},i} = \int_{C_i} \|x - \boldsymbol{v}_i\|^2 dx \\ &= \int_{C_i} \|x\|^2 dx - 2\int_{C_i} (x.\boldsymbol{v}_i) dx + \int_{C_i} \|\boldsymbol{v}_i\|^2 dx \\ &= \int_{C_i} \left(x_x^2 + x_y^2\right) dx - 2\int_{C_i} \left(\boldsymbol{v}_{i,x} x_x + \boldsymbol{v}_{i,y} x_y\right) dx + \int_{C_i} \|\boldsymbol{v}_i\|^2 dx\end{aligned} \tag{15}$$

where x_x and $\boldsymbol{v}_{i,x}$ are, respectively, the x coordinate of the point x and the cycle vertex $\boldsymbol{v}_i$ (and the same applies for the subindex y).

The gradient of $\mathcal{E}_{\text{CVT},i}$ with respect to $\boldsymbol{v_i}$ is:

$$\mathcal{G}_{CVT,i} = 2\left[\boldsymbol{v_i}\left(\int_{C_i} 1dx\right) - \left(\int_{C_i} xdx\right)\right] \tag{16}$$

New terms that appear in Equations 15 and 16 are moments of order $n = 0, 1, 2$ of the point cell. The Shoelace formula allows the computation of the signed area of a polygon (the moment of order 0) and generalizes to moments greater than $n > 0$.

We denote by p_j the vertices of the polygon C_i and by (jk) its edges oriented in clockwise order. For simplicity, we write $(j,k) \in C_i$ to iterate over the edges of the cell. We first compute the integral of x_y^n over the cell:

$$\begin{aligned}\int_{C_i} x_y^n dx &= \sum_{(jk)\in C_i} (p_{k,x} - p_{j,x}) \int_0^1 \int_0^{p_{j,y}+t(p_{k,y}-p_{j,y})} y^n dydt \\ &= \sum_{(jk)\in C_i} \frac{p_{k,x} - p_{j,x}}{n+1} \int_0^1 \left(p_{j,y} + t(p_{k,y} - p_{j,y})\right)^{n+1} dt \\ &= \sum_{(jk)\in C_i} \frac{p_{k,x} - p_{j,x}}{(n+1)(n+2)} \frac{p_{k,y}^{n+2} - p_{j,y}^{n+2}}{p_{k,y} - p_{j,y}}\end{aligned} \tag{17}$$

and we finally obtain:

$$\int_{C_i} x_y^n dx = \frac{1}{(n+1)(n+2)} \sum_{(jk)\in C_i} (p_{k,x} - p_{j,x}) \sum_{l=0}^{n+1} p_{k,y}^l p_{j,y}^{n+1-l} \tag{18}$$

Analogously, by swapping the two axis and by multiplying by -1 to keep a clockwise orientation we get the integral of x_x^n:

$$\int_{C_i} x_x^n dx = \frac{1}{(n+1)(n+2)} \sum_{(jk)\in C_i} (p_{j,y} - p_{k,y}) \sum_{l=0}^{n+1} p_{k,x}^l p_{j,x}^{n+1-l} \tag{19}$$

Segment cell. For $(i,j) \in \mathcal{T}_E$, we want to compute the objective associated with the segment cell C_{ij}. First, we project our points on a new orthogonal basis to facilitate subsequent derivations. This basis is composed of the two following unit vectors $\hat{v}$ and $\hat{w}$:

$$\hat{v} = \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\|\boldsymbol{v}_j - \boldsymbol{v}_i\|}, \qquad \hat{w} = R_{\pi/2}\hat{v} \tag{20}$$

Where $R_{\pi/2}$ is the matrix of a rotation by an angle $\pi/2$. We then consider the following affine isometric transformation m:

$$m : x \mapsto \begin{pmatrix}(x - \boldsymbol{v}_i).\hat{v} & (x - \boldsymbol{v}_i).\hat{w}\end{pmatrix} \tag{21}$$

Two functions parameterize the transformed cell $m(C_{ij})$: $f < 0$ and $g > 0$ representing the lower border and the cell's upper border. They are illustrated Figure 24. The support used for these two functions is the interval $]0,1[$. We then rewrite the segment objective as:

$$\begin{aligned}\mathcal{E}_{\text{CVT},(ij)} &= \int_{C_{ij}} \text{dist}\left(x,]\boldsymbol{v}_i, \boldsymbol{v}_j[\right)^2 dx \\ &= \|\boldsymbol{v}_j - \boldsymbol{v}_i\| \int_0^1 \int_{f(t)}^{g(t)} y^2 dydt\end{aligned} \tag{22}$$

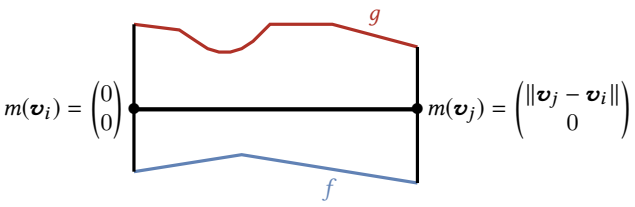


Fig. 24. The segment cell boundary is represented by two functions f, g.

In order to derive the gradient we write how the objective evolves when we slightly move by ϵ the two endpoints of the segment:

$$\frac{\mathcal{E}_{\text{CVT},(ij)}\begin{pmatrix}\boldsymbol{v}_i + \epsilon_i\\ \boldsymbol{v}_j + \epsilon_j\end{pmatrix}}{\|\boldsymbol{v}_j - \boldsymbol{v}_i\|} = \int_0^1 \int_{f(t)}^{g(t)} \left(y - \hat{w}.\left(\epsilon_i + t(\epsilon_j - \epsilon_i)\right)\right)^2 dydt \tag{23}$$

We then derive the gradient with respect to $\boldsymbol{v}_i$:

$$\mathcal{G}^{(i)}_{\text{CVT},(ij)} = -\|\boldsymbol{v}_j - \boldsymbol{v}_i\|\hat{w} \int_0^1 (1-t)\left(g(t)^2 - f(t)^2\right) dt \tag{24}$$

and with respect to $\boldsymbol{v}_j$:

$$\mathcal{G}^{(j)}_{\text{CVT},(ij)} = -\|\boldsymbol{v}_j - \boldsymbol{v}_i\|\hat{w} \int_0^1 t\left(g(t)^2 - f(t)^2\right) dt \tag{25}$$

For each vertex p_k of the cell C_{ij}, we let $q_k = m(p_k)$ be the associated vertex in the new transformed cell. Using the expression of the y moment of order 2 (see Equation 18), we get:

$$\mathcal{E}_{\text{CVT},(ij)} = \frac{1}{12} \sum_{(kl)\in C_{ij}} (q_{l,x} - q_{k,x}) \sum_{m=0}^{3} q_{k,y}^m q_{l,y}^{3-m} \tag{26}$$

For the gradient with respect to the two endpoints we first introduce new values:

$$u_k = \frac{q_{k,x}}{\|\boldsymbol{v}_j - \boldsymbol{v}_i\|}, \qquad u_l = \frac{q_{l,x}}{\|\boldsymbol{v}_j - \boldsymbol{v}_i\|} \tag{27}$$

and we then have:

$$\begin{aligned}\mathcal{G}^{(i)}_{\text{CVT},(ij)} &= \hat{w} \sum_{(kl)\in C_{ij}} (q_{k,x} - q_{l,x}) \int_0^1 \left(q_{k,y} + t(q_{k,y} + q_{l,y})\right)^2 \\ &\qquad (1 - (u_k + t(u_l - k))) \\ &= w \sum_{(kl)\in C_{ij}} \frac{q_{k,x} - q_{l,x}}{12} \sum_{m=1}^{3} (4 - mu_k - (4-m)u_l)\, q_{k,y}^{m-1} q_{l,y}^{3-m}\end{aligned} \tag{28}$$

$$\begin{aligned}\mathcal{G}^{(j)}_{\text{CVT},(ij)} &= \hat{w} \sum_{(kl)\in C_{ij}} (q_{k,x} - q_{l,x}) \int_0^1 \left(q_{k,y} + t(q_{k,y} + q_{l,y})\right)^2 \\ &\qquad (u_k + t(u_l - k)) \\ &= w \sum_{(kl)\in C_{ij}} \frac{q_{k,x} - q_{l,x}}{12} \sum_{m=1}^{3} (mu_k + (4-m)u_l)\, q_{k,y}^{m-1} q_{l,y}^{3-m}\end{aligned} \tag{29}$$

A.2 Length preservation objective

We denote by L_0 the target length. The length preservation objective keeps the length L of $\mathcal{T}$ as close as possible to L_0:

$$\mathcal{E}_{\text{Len}} = (L - L_0)^2 = \left(\sum_{(ij)\in\mathcal{T}_E} \|\boldsymbol{v}_j - \boldsymbol{v}_i\| - L_0 \right)^2 \tag{30}$$

The gradient with respect to $\boldsymbol{v}_i$ is given by:

$$\mathcal{G}^{(i)}_{\text{Len}} = 2\,(L - L_0) \left(\frac{\boldsymbol{v}_i - \boldsymbol{v}_{i_+}}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_+}\|} + \frac{\boldsymbol{v}_i - \boldsymbol{v}_{i_-}}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_-}\|} \right) \tag{31}$$

This objective is associated with the following normalized weight:

$$\gamma_{\text{Len}} = 0.4 \cdot \text{area}(\mathcal{P})/N^2 \tag{32}$$

A.3 Smoothing objective

In order to have a smoother trajectory $\mathcal{T}$ we minimize the following Laplacian:

$$\mathcal{E}_{\text{Lap}} = \sum_{i=1}^{N} \left\| \boldsymbol{v}_i - \frac{1}{2}\left(\boldsymbol{v}_{i_+} + \boldsymbol{v}_{i_-}\right) \right\|^2 \tag{33}$$

This objective smooths the sharp angles of *cycle*, redistributing its vertices. The gradient with respect to $\boldsymbol{v}_i$ is:

$$\mathcal{G}^{(i)}_{\text{Lap}} = 3\boldsymbol{v}_i + \frac{1}{2}\left(\boldsymbol{v}_{i_{++}} + \boldsymbol{v}_{i_{--}}\right) - 2\left(\boldsymbol{v}_{i_+} + \boldsymbol{v}_{i_-}\right) \tag{34}$$

We use the following normalized weight:

$$\gamma_{\text{Lap}} = 0.3 \cdot \text{area}(\mathcal{P})/N \tag{35}$$

A.4 Alignment objective

Recall that we compute a vector field V by doubling the angle of O (Section 3.2). We then denote by $\mathcal{V}_i$ the integral of V over C_i and by $\mathcal{V}_{ij}$ the integral of V over C_{ij}:

$$\mathcal{V}_i = \int_{C_i} V(x)dx \qquad \mathcal{V}_{ij} = \int_{C_{ij}} V(x)dx \tag{36}$$

The objective function $\mathcal{E}_{\text{Ali}}$ seeks to minimize the angle difference between $\mathcal{T}$ and the integrated vector field inside each cell:

$$\begin{aligned} \mathcal{E}_{\text{Ali}} \quad &= \sum_{i=1}^{N} \frac{\|\mathcal{V}_i\|}{Z} \left(\angle\left(\boldsymbol{v}_{i_+} - \boldsymbol{v}_{i_-}\right) - \frac{\angle\mathcal{V}_i}{2} \right)^2 \\ &\quad + \sum_{(ij)\in\mathcal{T}_E} \frac{\|\mathcal{V}_{ij}\|}{Z} \left(\angle\left(\boldsymbol{v}_j - \boldsymbol{v}_i\right) - \frac{\angle\mathcal{V}_{ij}}{2} \right)^2 \end{aligned} \tag{37}$$

Angles can be defined using $\arctan$ function. We also write down its gradient for later:

$$\angle v = \begin{cases} \arctan(v_y/v_x) & \text{if } v_x > 0 \\ \arctan(v_y/v_x) + \pi & \text{if } v_x < 0 \\ (\pi v_y)\,/\,(2|v_y|) & \text{if } v_x = 0 \end{cases} \qquad \nabla\angle v = \frac{R_{\pi/2}v}{\|v\|^2} \tag{38}$$

The values of $\angle\mathcal{V}_i$ and $\angle\mathcal{V}_{ij}$ in (37) are taken such that the absolute differences between angles are less than $\pi/2$. Finally, Z allows us to normalize the sum to obtain a weighted mean of squared angle differences. We have:

$$Z = \left(\sum_{i=1}^{N} \|\mathcal{V}_i\| + \sum_{(ij)\in\mathcal{T}_E} \|\mathcal{V}_{ij}\| \right) \left(\int_{\mathcal{P}} \|V(x)\|dx \right)^{-1} \tag{39}$$

The integral of the norm over $\mathcal{P}$ is the total area in which the vector field is non-zero. This is useful to properly weight this objective with the isotropic/anisotropic one.

The gradient of this objective is approximate since the variations of the integrals $\mathcal{V}_i$ and $\mathcal{V}_{ij}$ are ignored. The gradient with respect to $\boldsymbol{v}_i$ is approximated as follows:

$$\begin{aligned} \mathcal{G}^{(i)}_{\text{Ali}} \quad = \; & \frac{\|\mathcal{V}_{i_-}\|}{Z} \frac{R_{\pi/2}(\boldsymbol{v}_i - \boldsymbol{v}_{i_{--}})}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_{--}}\|^2} \left(\angle\left(\boldsymbol{v}_i - \boldsymbol{v}_{i_{--}}\right) - \frac{\angle\mathcal{V}_{i_-}}{2} \right) \\ & - \frac{\|\mathcal{V}_{i_+}\|}{Z} \frac{R_{\pi/2}(\boldsymbol{v}_{i_{++}} - \boldsymbol{v}_i)}{\|\boldsymbol{v}_{i_{++}} - \boldsymbol{v}_i\|^2} \left(\angle\left(\boldsymbol{v}_{i_{++}} - \boldsymbol{v}_i\right) - \frac{\angle\mathcal{V}_{i_+}}{2} \right) \\ & + \frac{\|\mathcal{V}_{i_-i}\|}{Z} \frac{R_{\pi/2}(\boldsymbol{v}_i - \boldsymbol{v}_{i_-})}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_-}\|^2} \left(\angle\left(\boldsymbol{v}_i - \boldsymbol{v}_{i_-}\right) - \frac{\angle\mathcal{V}_{i_-i}}{2} \right) \\ & - \frac{\|\mathcal{V}_{ii_+}\|}{Z} \frac{R_{\pi/2}(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i)}{\|\boldsymbol{v}_{i_+} - \boldsymbol{v}_i\|^2} \left(\angle\left(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i\right) - \frac{\angle\mathcal{V}_{ii_+}}{2} \right) \end{aligned} \tag{40}$$

Vector field discretization. Our field V is represented by a 2D array of size $W \times H$ whose elements are 2D vectors. For $x \in \{0, \dots, W-1\}$ and $y \in \{0, \dots, H-1\}$, we denote by $V[x][y]$ the vector at coordinate (x, y) in the array. If our shape $\mathcal{P}$ is included inside a rectangle $[0, \mathscr{W}] \times [0, \mathscr{H}]$ of the Euclidean plane, then the value $V[x][y]$ in our array corresponds to the vector field inside the small rectangle centered in $\left(\left(x + \frac{1}{2}\right)\frac{\mathscr{W}}{W}, \left(y + \frac{1}{2}\right)\frac{\mathscr{H}}{H}\right)$ with a width $\frac{\mathscr{W}}{W}$ and a height $\frac{\mathscr{H}}{H}$. To compute the integral $\mathcal{V}$ over a cell C, we again use the Shoelace formula:

$$\mathcal{V} = \sum_{(ij)\in C} (p_{j,x} - p_{i,x}) \int_0^1 \int_0^{p_{i,y} + t(p_{j,y} - p_{i,y})} V(p_{i,x} + t(p_{j,x} - p_{i,x}), y) dy dt \tag{41}$$

In order to efficiently compute the integral over y, we pre-compute a second array SV:

$$SV[x][y] = \sum_{j=0}^{y} V[x][j] \tag{42}$$

For two points (x_0, y_0) and (x_1, y_1) in the quad (X, Y) of our grid. That is to say, when these inequalities are verified:

$$X\frac{\mathscr{W}}{W} \leqslant x_0, x_1 \leqslant (X+1)\frac{\mathscr{W}}{W} \tag{43}$$

$$Y\frac{\mathscr{H}}{H} \leqslant y_0, y_1 \leqslant (Y+1)\frac{\mathscr{H}}{H} \tag{44}$$

the integral of the vector field under the segment connecting these two points is:

$$\begin{aligned} \int_0^1 \int_0^{y_0 + t(y_1 - y_0)} & V(x_0 + t(x_1 - x_0), y) dy dt \\ & = SV[X][Y] - \left(Y + 1 - \frac{y_0 + y_1}{2\mathscr{H}} H\right) V[X][Y] \end{aligned} \tag{45}$$

Then we decompose each edge (ij) of the cell C in small parts contained in a single quad of our grid (represented by alternating between dark and light in Figure 25) and compute the double integral over each small part ($SV[X][Y]$ is the integral over the union of the blue and red areas, and $\left(Y + 1 - \frac{y_0+y_1}{2\mathscr{H}} H\right) V[X][Y]$ is the integral over the red area).

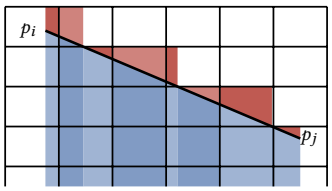


Fig. 25. The vector field's integral under a segment is obtained by decomposing it in smaller segments contained in single quads of the grid.

A.5 Orientation objective

We control the probability distribution of the orientations along $\mathcal{T}$ to be either as isotropic or as anisotropic as possible. To do so, we minimize/maximize the squared 2-Wasserstein distance between a uniform distribution $\mathcal{U}$ over $[0, \pi[$ and the distribution of the orientations in $\mathcal{T}$ that we denote $\mathcal{D}_\mathcal{T}$. The Wasserstein distance is a very convenient metric between probability distributions coming from optimal transport [Villani 2003]. The orientation distribution of $\mathcal{T}$ can be defined as:

$$\mathcal{D}_\mathcal{T} = \frac{1}{L} \sum_{(ij)\in\mathcal{T}_E} \|\boldsymbol{v}_j - \boldsymbol{v}_i\| \delta_{\angle(\boldsymbol{v}_j - \boldsymbol{v}_i)} \tag{46}$$

Where $\angle v$ is the angle of v modulo π taken inside the interval $[0, \pi[$, and L is the total length of $\mathcal{T}$. We know from [Rabin et al. 2011] that the Wasserstein distance on a circle can be expressed as:

$$\mathscr{W}_2^2(\mathcal{U}, \mathcal{D}_\mathcal{T}) = \inf_\alpha \int_0^1 \left((F^\alpha)^{(-1)}(t) - G_\mathcal{T}^{(-1)}(t) \right)^2 dt \tag{47}$$

where F and $G_\mathcal{T}$ are, respectively, the cumulative distribution of $\mathcal{U}$ and $\mathcal{D}_\mathcal{T}$. That is to say:

$$F(\theta) = \int_0^\theta \frac{1}{\pi} d\phi = \theta/\pi, \quad G_\mathcal{T}(\theta) = \frac{1}{L} \sum_{(ij)\in\mathcal{T}_E} \|\boldsymbol{v}_j - \boldsymbol{v}_i\| \mathbb{1}_{\{\angle(\boldsymbol{v}_j - \boldsymbol{v}_i) < \theta\}} \tag{48}$$

F^α is defined as $F - \alpha$ and the pseudo inverses $(F^\alpha)^{(-1)}$ and $G_\mathcal{T}^{(-1)}$ are defined as:

$$(F^\alpha)^{(-1)}(t) = \inf\{\theta \mid F(\theta) - \alpha > t\} = \pi(t + \alpha) \tag{49}$$

$$G_\mathcal{T}^{(-1)}(t) = \inf\{\theta \mid G_\mathcal{T}(\theta) > t\} = \sum_{(ij)\in\mathcal{T}_E} \angle(\boldsymbol{v}_j - \boldsymbol{v}_i) \, \mathbb{1}_{\left\{t \in \left[T_{ij}^0, T_{ij}^1\right[\right\}} \tag{50}$$

where:

$$T_{ij}^0 = \frac{1}{L} \sum_{\substack{(kl)\in\mathcal{T}_E \\ \angle(\boldsymbol{v}_l - \boldsymbol{v}_k) < \angle(\boldsymbol{v}_j - \boldsymbol{v}_i)}} \|\boldsymbol{v}_l - \boldsymbol{v}_k\|, \qquad T_{ij}^1 = \frac{1}{L} \sum_{\substack{(kl)\in\mathcal{T}_E \\ \angle(\boldsymbol{v}_l - \boldsymbol{v}_k) \leqslant \angle(\boldsymbol{v}_j - \boldsymbol{v}_i)}} \|\boldsymbol{v}_l - \boldsymbol{v}_k\| \tag{51}$$

In the distance (47), the infimum is obtained at the value α_0 defined as the mean of $\frac{1}{\pi} G_\mathcal{T}^{(-1)}(t) - t$ over $[0, 1]$, which is:

$$\alpha_0 = \frac{1}{\pi} \mathbb{E}[\mathcal{D}_\mathcal{T}] - \frac{1}{2} \tag{52}$$

We then obtain:

$$\mathscr{W}_2^2(\mathcal{U}, \mathcal{D}_\mathcal{T}) = \sum_{(ij)\in\mathcal{T}_E} \int_{T_{ij}^0}^{T_{ij}^1} \left(\pi t + \pi\alpha_0 - \angle(\boldsymbol{v}_j - \boldsymbol{v}_i) \right)^2 dt \tag{53}$$

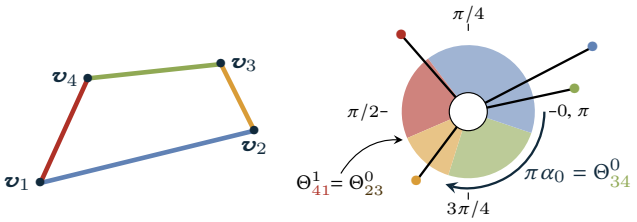


Fig. 26. **Left:** A path having four segments. **Right:** Transport map of the path orientation distribution to the uniform distribution. Sticks are Dirac located at the orientation of the segment having the same color. They are a map to domains of $[0, \pi[$ represented by portions of the ring having an area proportional to the segment length.

To simplify the following expressions, we let:

$$\Theta_{ij}^k = \pi T_{ij}^k + \pi\alpha_0 \tag{54}$$

Finally, we define an isotropic objective as the squared Wasserstein distance ($\mathcal{E}_{\text{Iso}}(\mathcal{T}) = \mathscr{W}_2^2(\mathcal{U}, \mathcal{D}_\mathcal{T})$). Computing the integral in (53), we get:

$$\mathcal{E}_{\text{Iso}}(\mathcal{T}) = \frac{1}{3\pi} \sum_{(ij)\in\mathcal{T}_E} \left(\Theta_{ij}^1 - \angle(\boldsymbol{v}_j - \boldsymbol{v}_i) \right)^3 - \left(\Theta_{ij}^0 - \angle(\boldsymbol{v}_j - \boldsymbol{v}_i) \right)^3 \tag{55}$$

As for the alignment objective, we only compute an approximation of the gradient by ignoring the variations of Θ_{ij}^k. We tried to add the variations of α inside the gradient, but the impact was really negligible. Our approximation of the gradient with respect to $\boldsymbol{v}_i$ is as follows:

$$\begin{aligned} \mathcal{G}_{\text{Iso}}^{(i)} = {} & \frac{R_{\pi/2}(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i)}{\pi \|\boldsymbol{v}_{i_+} - \boldsymbol{v}_i\|^2} \left[\left(\pi\Theta_{ii_+}^1 - \angle(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i) \right)^2 - \left(\pi\Theta_{ii_+}^0 - \angle(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i) \right)^2 \right] \\ & - \frac{R_{\pi/2}(\boldsymbol{v}_i - \boldsymbol{v}_{i_-})}{\pi \|\boldsymbol{v}_i - \boldsymbol{v}_{i_-}\|^2} \left[\left(\pi\Theta_{i_-i}^1 - \angle(\boldsymbol{v}_i - \boldsymbol{v}_{i_-}) \right)^2 - \left(\pi\Theta_{i_-i}^0 - \angle(\boldsymbol{v}_i - \boldsymbol{v}_{i_-}) \right)^2 \right] \end{aligned} \tag{56}$$

One can notice that $\mathcal{E}_{\text{Iso}}$ is included in $\left[0, \frac{\pi^2}{12}\right]$. To have an anisotropic objective included in the same interval we define it as:

$$\mathcal{E}_{\text{anIso}}(\mathcal{T}) = \frac{\pi^2}{12} - \mathcal{E}_{\text{Iso}}(\mathcal{T}) \tag{57}$$

The gradient of the anisotropic objective is simply the opposite of the isotropy gradient (56).

Then we regroup these two objectives inside the orientation objective defined as:

$$\mathcal{E}_{\text{Ori}} = \sum_i \text{area}(\mathcal{I}_i) \, \mathcal{E}_{\text{Iso}}(\mathcal{T} \cap \mathcal{I}_i) + \sum_i \text{area}(\mathcal{A}_i) \, \mathcal{E}_{\text{anIso}}(\mathcal{T} \cap \mathcal{A}_i) \tag{58}$$

As for the CVT energy, the intersection of $\mathcal{T}$ with zone boundaries creates new vertices. Again the gradient is transferred to the initial vertices of $\mathcal{T}$ using the chain rule (11).

Finally, a last weight term γ_{Obj} allows us to control the influence of $\mathcal{E}_{\text{Ali}}$ and $\mathcal{E}_{\text{Ori}}$. We use the following normalized weight:

$$\gamma_{\text{Obj}} = \frac{\text{area}(\mathcal{P})^3}{50 \cdot L_0^2} \left(\int_{\mathcal{P}} \|V(x)\| dx + \sum_i \text{area}\left(\mathcal{I}_i\right) + \sum_i \text{area}\left(\mathcal{A}_i\right) \right)^{-1} \tag{59}$$

A.6 Zoning objective

The zoning objective is only used during the combinatorial optimization. It counts the number of intersections between $\mathcal{T}$ and material zone boundaries:

$$\mathcal{E}_{\text{Zon}} = \sum_{(i,j) \in \mathcal{T}_E} \sum_{k \neq l} \#\left([\boldsymbol{v}_i, \boldsymbol{v}_j] \cap \mathcal{Z}_k \cap \mathcal{Z}_l \right) \tag{60}$$

Where $\#S$ is the cardinal of a set S. If we assume that no segment of the path is perfectly contained in a zone boundary, the sum's cardinals are always finite.

The weight γ_{Comb} attempts to prioritize the zoning objective $\mathcal{E}_{\text{Zon}}$. To do so, we set γ_{Comb} such that $\gamma_{\text{Comb}}(\mathcal{E}_{\text{Ali}} + \mathcal{E}_{\text{Ori}})$ is included in $[0, 1]$. The squared angle difference in the alignment objective is bounded by $\pi^2/4$, and the squared Wasserstein distance is bounded by $\pi^2/12$. We then set γ_{Comb} to:

$$\gamma_{\text{Comb}} = \frac{4}{\pi^2} \left(\int_{\mathcal{P}} \|V(x)\| dx + \sum_i \text{area}\left(\mathcal{I}_i\right) + \sum_i \text{area}\left(\mathcal{A}_i\right) \right)^{-1} \tag{61}$$

REFERENCES

Ergun Akleman, Qing Xing, Pradeep Garigipati, Gabriel Taubin, Jianer Chen, and Shiyu Hu. 2013. Hamiltonian cycle art: Surface covering wire sculptures and duotone surfaces. *Computers & graphics* 37, 5 (2013), 316–332.

Esther M Arkin, Martin Held, Joseph SB Mitchell, and Steven S Skiena. 1996. Hamiltonian triangulations for fast rendering. *The Visual Computer* 12, 9 (1996), 429–444.

Therese C. Biedl, Prosenjit Bose, Erik D. Demaine, and Anna Lubiw. 2001. Efficient Algorithms for Petersen's Matching Theorem. *Journal of Algorithms* 38, 1 (2001), 110–134.

Robert Bosch. 2010. Simple-closed-curve sculptures of knots and links. *Journal of Mathematics and the Arts* 4, 2 (2010), 57–71.

Robert Bosch and Adrianne Herman. 2004. Continuous line drawings via the traveling salesman problem. *Operations research letters* 32, 4 (2004), 302–303.

Zhonggui Chen, Zifu Shen, Jianzhi Guo, Juan Cao, and Xiaoming Zeng. 2017. Line drawing for 3D printing. *Computers & Graphics* 66 (2017), 85–92.

Puikei Cheng, Wing Kam Liu, Kornel Ehmann, and Jian Cao. 2020. Enumeration of Additive Manufacturing Toolpaths Using Hamiltonian Paths. *Manufacturing Letters* 26 (2020), 29–32.

Revital Dafner, Daniel Cohen-Or, and Yossi Matias. 2000. Context-based Space Filling Curves. *Computer Graphics Forum* 19, 3 (2000), 209–218.

Sam Dehaeck, B Scheid, and P Lambert. 2018. Adaptive stitching for meso-scale printing with two-photon lithography. *Additive Manufacturing* 21 (2018), 589–597.

Jack Edmonds. 1965. Paths, trees, and flowers. *Canadian Journal of mathematics* 17 (1965), 449–467.

Guoxin Fang, Tianyu Zhang, Sikai Zhong, Xiangjia Chen, Zichun Zhong, and Charlie CL Wang. 2020. Reinforced FDM: multi-axis filament alignment with controlled anisotropic strength. *ACM Transactions on Graphics* 39, 6 (2020), 1–15.

PAGS Giachini, SS Gupta, Wendong Wang, Dylan Wood, Muhammad Yunusa, E Baharlou, Metin Sitti, and Achim Menges. 2020. Additive manufacturing of cellulose-based materials with continuous, multidirectional stiffness gradients. *Science advances* 6, 8 (2020).

John Giannatsis, A Vassilakos, Vassilios Canellidis, and Vassilis Dedoussis. 2015. Fabrication of graded structures by extrusion 3D Printing. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 175–179.

M Gopi and David Eppstein. 2004. Single-strip triangulation of manifolds with arbitrary topology. *Computer Graphics Forum* 23, 3 (2004), 371–379.

Topraj Gurung, Mark Luffel, Peter Lindstrom, and Jarek Rossignac. 2011. LR: Compact Connectivity Representation for Triangle Meshes. In *ACM SIGGRAPH 2011 Papers*. Article 67, 8 pages.

Jean Hergel, Kevin Hinz, Sylvain Lefebvre, and Bernhard Thomaszewski. 2019. Extrusion-Based Ceramics Printing with Strictly-Continuous Deposition. *ACM Transactions on Graphics* 38, 6, Article 194 (2019), 11 pages.

Dorit S. Hochbaum and David B. Shmoys. 1985. A Best Possible Heuristic for the K-Center Problem. *Mathematics of Operations Research* 10, 2 (1985), 180âĂŞ184.

Samuel Hornus, Tim Kuipers, Olivier Devillers, Monique Teillaud, Jonàs Martínez, Marc Glisse, Sylvain Lazard, and Sylvain Lefebvre. 2020. Variable-width contouring for Additive Manufacturing. *ACM Transactions on Graphics* 39, Article 131 (2020), 17 pages.

Jingchao Jiang and Yongsheng Ma. 2020. Path planning strategies to optimize accuracy, quality, build time and material use in additive manufacturing: a review. *Micromachines* 11, 7 (2020), 633.

Craig S Kaplan, Robert Bosch, et al. 2005. TSP art. In *Renaissance Banff: Mathematics, music, art, culture*. Bridges Conference, 301–308.

Joseph R Kubalak, Alfred L Wicks, and Christopher B Williams. 2019. Deposition path planning for material extrusion using specified orientation fields. *Procedia Manufacturing* 34 (2019), 754–763.

Tim Kuipers, Jun Wu, and Charlie CL Wang. 2019. CrossFill: foam structures with graded density for continuous material extrusion. *Computer-Aided Design* 114 (2019), 37–50.

Bruno Lévy and Nicolas Bonneel. 2013. Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In *Proceedings of the 21st international meshing roundtable*. Springer, 349–366.

Sen Lin, Lingwei Xia, Guowei Ma, Shiwei Zhou, and Yi Min Xie. 2019. A maze-like path generation scheme for fused deposition modeling. *The International Journal of Advanced Manufacturing Technology* 104, 1-4 (2019), 1509–1519.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1-3 (1989), 503–528.

Jikai Liu and Huangchao Yu. 2017. Concurrent deposition path planning and structural topology optimization for additive manufacturing. *Rapid Prototyping Journal* 23, 5 (2017), 930–942.

Rahnuma Islam Nishat. 2020. *Reconfiguration of Hamiltonian cycles and paths in grid graphs*. Ph.D. Dissertation. University of Victoria.

Rahnuma Islam Nishat and Sue Whitesides. 2017. Bend complexity and Hamiltonian cycles in grid graphs. In *International Computing and Combinatorics Conference*. Springer, 445–456.

Muller P., Hascoet J.-Y., and Mognol P. 2014. Toolpaths for additive manufacturing of functionally graded materials (FGM) parts. *Rapid Prototyping Journal* 20 (2014), 511–522.

Alexios Papacharalampopoulos, Harry Bikas, and Panagiotis Stavropoulos. 2018. Path planning for the infill of 3D printed parts utilizing Hilbert curves. *Procedia Manufacturing* 21 (2018), 757–764.

Hans Pedersen and Karan Singh. 2006. Organic Labyrinths and Mazes. In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*. 79âĂŞ86.

Julien Rabin, Julie Delon, and Yann Gousseau. 2011. Transportation Distances on the Circle. *Journal of Mathematical Imaging and Vision* 41, 1-2 (2011), 147–167.

Nicholas Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Transactions on Graphics* 37, 4, Article 156 (2018), 13 pages.

Vicente Soler, Gilles Retsin, and Manuel Jimenez Garcia. 2017. A generalized approach to non-layered fused filament fabrication. In *Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture*. 562–571.

John C Steuben, Athanasios P Iliopoulos, and John G Michopoulos. 2016. Implicit slicing for functionally tailored additive manufacturing. *Computer-Aided Design* 77 (2016), 107–119.

Kam-Ming Mark Tam and Caitlin T Mueller. 2017. Additive manufacturing along principal stress lines. *3D Printing and Additive Manufacturing* 4, 2 (2017), 63–81.

Gabriel Taubin. 2003. Constructing hamiltonian triangle strips on quadrilateral meshes. In *Visualization and Mathematics III*. Springer, 69–91.

C. Villani. 2003. *Topics in Optimal Transportation*. American Mathematical Society.

Fernando J Wong and Shigeo Takahashi. 2013. Abstracting images into continuous-line artistic styles. *The Visual Computer* 29, 6-8 (2013), 729–738.

Lingwei Xia, Sen Lin, and Guowei Ma. 2020. Stress-based tool-path planning methodology for fused filament fabrication. *Additive Manufacturing* 32 (2020), 101020.

Xinyi Xiao and Sanjay Joshi. 2018. Automatic Toolpath Generation for Heterogeneous Objects Manufactured by Directed Energy Deposition Additive Manufacturing Process. *Journal of Manufacturing Science and Engineering* 140 (2018), 1087–1357.

Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2016. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics* 35, 4 (2016), 1–10.