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

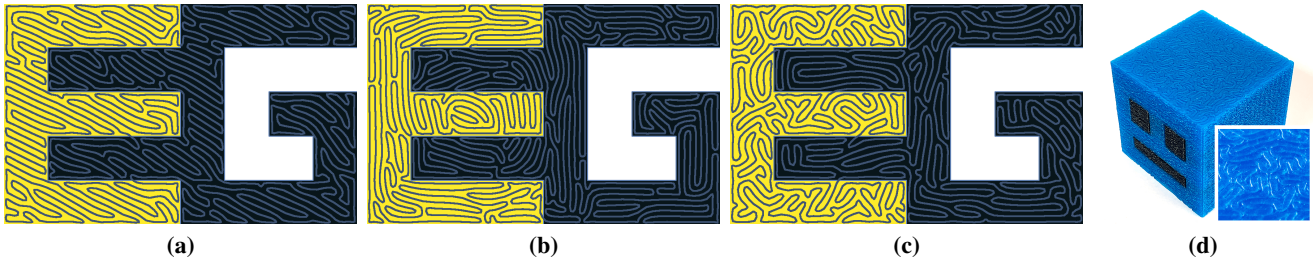
Adrien Bedel, Yoann Coudert-Osmont, Jonàs Martínez, Rahnuma Islam Nishat, Sue Whitesides, Sylvain Lefebvre

Volume xx (200y), Number z, pp. 1–20

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

A. Bedel[1†], Y. Coudert-Osmont[1†], J. Martínez[1], R. I. Nishat[2], S. Whitesides[3], S. Lefebvre[1]

[1]Université de Lorraine, CNRS, Inria, LORIA [2]Ryerson University [3]University of Victoria
[†] Joint first authors



(a) **(b)** **(c)** **(d)**

Figure 1: *Closed space-filling curves optimized under **(a)** anisotropy, **(b)** boundary alignment **(c)** isotropy inside the E and boundary alignment inside the G. Isotropy and anisotropy are optimized in terms of distribution of angles, while alignment is specified in a vector field (here parallel to the boundary). For the three curves, the number of transitions between the two letters of the logo is minimized. **(d)** Multi-material 3D printed cube, with isotropic layers optimized by our approach.*

Abstract

We explore the optimization of closed space-filling curves under orientation objectives. By solidifying material along the closed curve, solid layers of 3D prints 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 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;*

1. Introduction

Many additive manufacturing processes solidify material within each layer along trajectories [ALL*18]. The way these trajectories cover the layer has direct implications on the fabrication process itself and the object's final properties. In particular, the trajectories impact the quality of extrusion processes such as fused filament fabrication, silicon, ceramic, wire arc, and concrete extrusion. Thus, it is essential to minimize the occurrence of *travel moves*: motions where the material flow is interrupted while the extrusion device moves from one point to another. Such moves take time and introduce defects where the flow stops and resumes: the print head traveling above the part often leaves traces and spurious deposits. Therefore, a trend in Additive Manufacturing (AM) focuses on determining trajectories that fill a layer in a single continuous motion [ZGH*16, DSL18, PBS18, JM20]. In such cases, the trajectory is a cyclic space-filling curve. As layers are now printed in cycles, the start and end positions of the print head can be chosen advantageously (e.g., to minimize travel times), while the extrusion flow is only interrupted where necessary.

Another important criterion is the orientation of the trajectories. These directly impact the surface finish, as the trajectories remain visible and induce anisotropy in surface roughness. Several recent works also note that aligning the trajectories with the principal stress directions improves the mechanical robust-

ness [SIM16, TM17, LY17, KWW19a, XLM20, FZZ*20]. A distinct problem is to attempt to obtain *isotropic* behaviors, while the deposition tends to follow specific directions (e.g., zigzag and concentric patterns). This applies both to mechanical strength [LXM*19] and thermal differences during deposition [CLEC20].

Finally, another context where precise control over the trajectories is required is when switching materials during a print [PJYP14, XJ18, SMB*19, SSMVL19]. In such cases, the trajectories have to cover the layer while following a target field, ensuring correct mixtures are achieved despite relatively slow gradients of change, minimizing the number of switches. We refer to this property as trajectory *zoning*.

In this paper, we consider three different objectives: continuity, orientation, and zoning. 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 and minimizing crossing between zones forms a multi-objective optimization problem.

Interestingly, a similar challenge has been considered before in the context of Computer Graphics, primarily for interactive artistic creation of maze-like patterns [PS06, WT13]. However, a number of limitations make these techniques less suited for optimizing our objectives automatically across large numbers of slices. We provide further discussion and comparisons in Section 5.2.

We depart from existing methods and approach this challenge 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. 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 globally optimized for all objectives.

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 distinct zones. The trajectory can also be optimized to remain in a designated area (zoning) for as long as possible. This objective ensures that multi-material mixtures remain stable while in a specific area.

We consider that a *cyclic curve* is 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 [AHMS96, Tau03, GE04, GLLR11], or improving image compression rates [DCOM00].

Space-filling curves have unique aesthetic qualities, which led to their use for artistic creations. Bosch et al. [BH04, Bos10] and Kaplan et al. [KB*05] explore continuous line drawings using TSP solvers to find a Hamiltonian cycle. Wong and Takahashi [WT13] stylize images, growing an initial cycle in oriented grids obtained from quad meshing. A similar growth process is performed in [DCOM00]. The approach of Pedersen and Singh [PS06] 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 [SC18] 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 control over the final trajectories by locally influencing the process, as demonstrated in [PS06, WT13]. Thus, they are a possible option to approach the challenges we seek to address. We discuss the difficulties this presents in Section 5.2.

Graph theory approaches have also been proposed to find covering cycles. These combinatorial methods are very efficient since – on specific graphs – they are supported by a linear time algorithm [Tau03]. This is demonstrated in Akleman et al. [AXG*13] to generate Hamiltonian cycles along surfaces for various artistic effects. We use a similar approach to initialize our optimizer.

Space-filling curves have properties that make them attractive as deposition trajectories for Additive Manufacturing [JM20]. Chen et al. [CLEC20] enumerate all cycles within a regular square grid, evaluating each cycle for the resulting thermal properties during deposition. Lin et al. [LXM*19] 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. [PBS18] explore the use of Hilbert curves for continuous deposition, noting that uninterrupted extrusion reduces print time. Continuity is also an essential factor in final quality when extruding pastes or ceramics [HHLT19]. Giannatsis et al. [GVCD15] construct space-filling curves following a material gradient. Chen et al. [CSG*17] vary the local density of the trajectories following an input image. Zhao et al. [ZGH*16] generate contour parallel fill trajectories that form singly connected spirals within a contour, leading to a continuous fill with smooth trajectories. Soler et al. [SRJG17] create 3D trajectories as Hamiltonian cycles in grids for wireframe 3D printing. In the context of two-photon lithography, Dahaeck et al. [DSL18] 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., [KWW19b]); however, our focus in this work is on dense, solid infills. These techniques do not provide any direct control over the orientation of the generated trajectories.

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 U, a set of isotropic zones $\mathcal{I}_i \subset \mathcal{P}$, a set of anisotropic zones

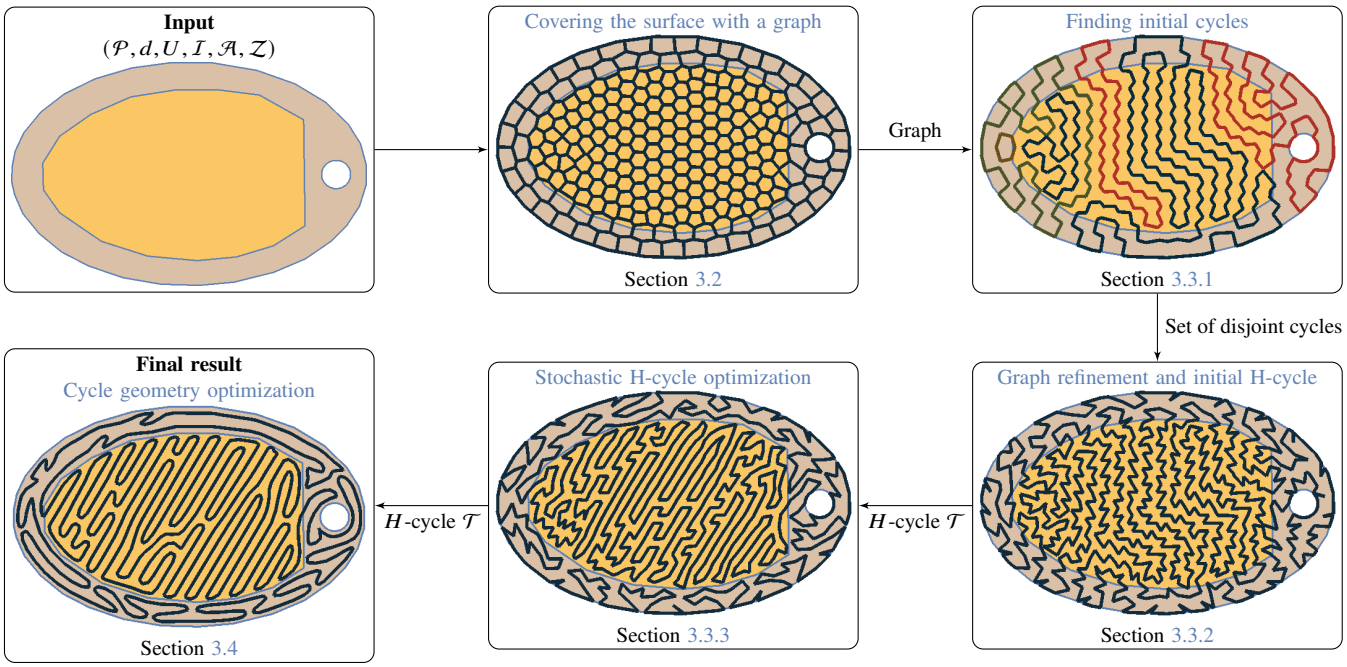


Figure 2: *General overview of our approach. We start by covering the input polygon $\mathcal{P}$ with a graph and finding an initial cycle. Then, we proceed to optimize the topology of the graph. Finally, we optimize its geometry. Each frame visually represents a step of the method, and is related to a different section.*

$\mathcal{A}_i \subset \mathcal{P}$, 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 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 U 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. 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 area around the segment. Multiplied by layer height, it gives the volume to extrude. Dividing $A(\boldsymbol{v}_i, \boldsymbol{v}_{i_+})$ area by the segment length gives a notion of the average spacing around.

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 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 U and encourages $\mathcal{T}$ 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. An open-source implementation of our approach can be found at `https://github.com/mfx-inria/controllable-space-filling-curve`.

Our approach operates in three main steps (see Figure 2):

- (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 Tessellation 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 U that specifies alignment vectors (vectors with the 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 its 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 [LN89]. 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. This leads to an optimal transport problem in the geometric solver context through the Wasserstein distance [Vil03]. 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 [GE04] 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 whose deletion would increase the number of connected components. Besides, we seek to obtain a graph with edges distributed evenly over $\mathcal{P}$.

From these two requirements, we choose to create the graph from the vertices and edges of a Centroidal Voronoi Tessellation (CVT). We generate initial seeds by intersecting a triangular tiling with our polygon $\mathcal{P}$ (see Figure 3).

The spacing d between adjacent vertices parameterizes the tiling. Each seed position is perturbed by a random offset, sampled in a disc of radius $\frac{d}{20}$, in order to get rid of co-circular cases. The value of 20 was determined empirically. 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 13). In practice, d is adjusted empirically by restarting the process with a lower d when the interspace is too high and higher d when the interspace is too low. After the Voronoi tessellation, 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 4. A CVT graph is extremely likely to have nodes of degree three everywhere, as co-circular cases are extremely rare. Actually, we have never observed a node with a degree greater than three. The graph is bridge-less by construction, as each edge belongs to a cyclic cell.

However, directly using a point CVT graph would 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 U, we can only hope to find a cycle with aligned edges if they exist in the graph. Therefore, the graph has to ideally contain some edges aligned along the non-zero vectors of U.

To align the edges, we introduce anisotropy in the Voronoi diagram. Instead of casting this as a metric distortion problem [LB13],

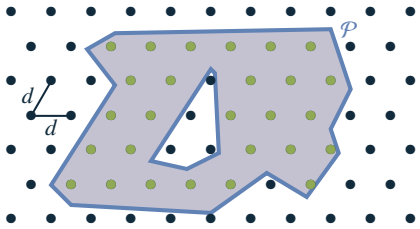


Figure 3: *Initial seeds (in green) are created by intersecting a triangular grid (side length d) with $\mathcal{P}$.*

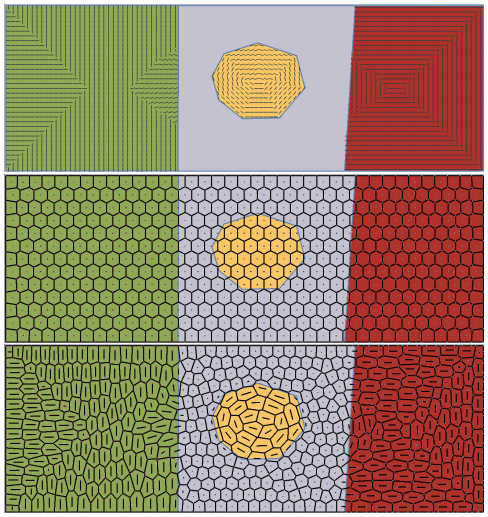


Figure 4: *Obtaining aligned edges with the needles.* ***Top:*** *vector field: the green, yellow, and red zones have alignment vectors specified.* ***Middle:*** *standard point Voronoi tessellation.* ***Bottom:*** *Voronoi tessellation obtained using needles. Many of the diagram edges now follow the input vector field.*

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 U, see Figure 4. 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.

In order to obtain cell edges aligned with the vector field U, we want the needle to be aligned with the average of the direction field U inside its own cell C_p. However, such a field has symmetry: a vector and its opposite represent the same direction. To break the symmetry, one can double the angle of U (denoted by $\angle U$), leading to a new vector field V. In that way, two opposite vectors in U become equal in V, and integration inside a cell is then possible.

$$V = \begin{cases} \begin{pmatrix} \cos(2\angle U) & \sin(2\angle U) \end{pmatrix} & \text{if } U \neq (0,0) \\ \begin{pmatrix} 0 & 0 \end{pmatrix} & \text{otherwise} \end{cases} \tag{4}$$

Now the angle of the needle can be expressed as $\frac{1}{2}\angle\mathcal{V}_p$, half the angle of $\mathcal{V}_p$, the integral of V inside C_p, as defined in the appendix equation (32).

The needle length is computed as $\mathcal{L}_p = \frac{||\mathcal{V}_p||}{\text{area}(C_p)} \cdot \sqrt{\frac{\text{area}(\mathcal{P})}{\pi S}}$, where the square root term is a global scaling factor based on the total area covered with S seeds and area(.) computes the area of a polygon. To ensure that needles never cross $\mathcal{L}_p$ is clamped so that the needle is fully contained in C_p.

We optimize for the seeds' positions under a *segment* CVT objective, using the L-BFGS algorithm [LN89]. We use the simplifying assumption that $\mathcal{V}_p$ undergoes only small changes at each iteration. Therefore, we only optimize seed positions, recomputing $\mathcal{V}_p$, $\mathcal{L}_p$ 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 $\mathcal{P}$. The above procedure gives us an initial graph satisfying our requirements (see Figure 4).

3.3. Combinatorial optimization

Once a bridge-less 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 [NW17, Nis20]. 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 crucial step that 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 [GE04]. We compute a *perfect matching* on the graph [Edm65]: a selection of edges such that every node is adjacent to exactly one selected edge. This can be done in linear time on bridge-less planar graphs of degree three [BBDL01]. Then, a set of cycles covering all graph nodes is obtained by inverting this edge selection. However, these cycles are not connected in a single Hamiltonian cycle (see Figure 5).

Gopi and Eppstein [GE04] 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.

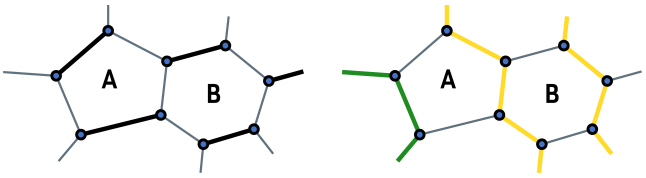


Figure 5: *Closeup on two cells A and B of a three connected, bridge-less planar graph.* ***Left:*** *a perfect matching of edges (thick black lines).* ***Right:*** *inverting the perfect matching creates cycles covering all graph nodes. Here two cycles (in green and orange color) are produced from which we see only a few edges.*

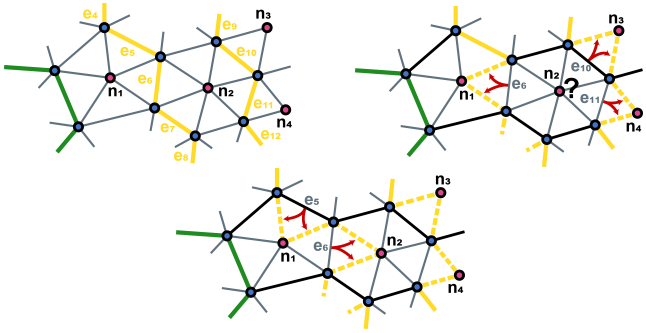


Figure 6: ***Top left:*** *the insertion of nodes n_1, n_2, n_3, and n_4 (in red color) produces new triangular faces. We seek to reconnect the new nodes with the two cycles passing around (in green and yellow color) while covering all nodes.* ***Top right:*** *we remove edges e_6, e_{10} and e_{11} and reconnect their endpoints with n_1, n_3 and n_4, respectively. However, we need to find an alternative since the resulting cycles do not cover the node n_2.* ***Bottom:*** *here we remove edges e_5 and e_6 and reconnect their endpoints with nodes n_1 and n_2 respectively. This leads to a satisfactory result covering all nodes.*

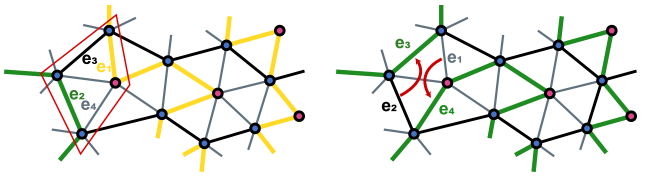


Figure 7: *Connecting cycles through split-and-reconnect. Here we have two different cycles (in green and yellow), and we identify an opportunity to split-and-reconnect using edges e_1, e_2, e_3 and e_4 (inside red quad).* ***Left:*** *initial configuration, the edge e_1 belongs to the yellow cycle, and the edge e_2 belongs to the green cycle.* ***Right:*** *the two cycles are joined by removing e_1, e_2 and adding e_3, e_4.*

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. The three-connected planar graph's low connectivity and mostly hexagonal faces make it challenging to perform simple modifications: any change impacts many edges at once.

Therefore, we propose refining the graph, increasing the density of edges, and 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 6), 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 6, top right.

Note that all cells have at least two cycle edges along their boundary by construction. Indeed, we started from a perfect matching (Section 3.3.1) and inverted the selection, ensuring that every node is connected to two-cycle edges and a third edge not in the cycle. Thus, 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 three 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 6, 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 6. 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 7: 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 8. The algorithm goes along the cycles again, searching for the first free edge connecting another cycle and doubling it. Typically, only very few double edges are necessary, as experimentally verified in Table 1. In practice, doubling an edge is not so problematic as the geometric optimizer will quickly recover the proper interspace.

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.

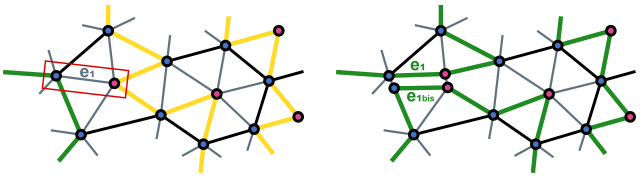
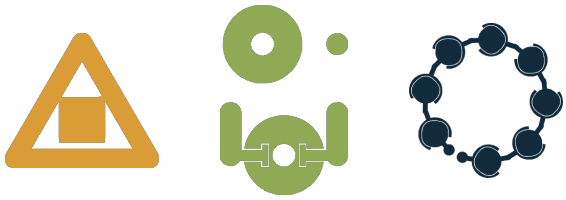


Figure 8: *Connecting cycles through edge doubling.* ***Left:*** *an unused edge e_1 (in red quad) between two different cycles (in green and yellow).* ***Right:*** *the two cycles are joined together after doubling the edge e_1 and its endpoints, adding an extra edge e_{1bis}.*



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 (db. edges) — when graph refinement is used or not. Models contain between* 100 *and* 200 *layers. Statistics are aggregated over all of them.*

3.3.3. Stochastic H-cycle optimization

Objectives. Given an 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 Appendix A 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 ($n = 100$ in our implementation). This histogram enables the computation of 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.

Evolving an H-cycle: local operations. The local operations we perform on the H-cycle are shown in Figure 9. There is a starting configuration and an updated configuration in each operation template. The starting configuration considers a set of nodes with a specific configuration of edges. The thick green edges are those that belong to the cycle; the thin black 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 9). A global connectivity check is required between two of the nodes, indicated in Figure 9 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 1.

Algorithm 1 The shuffle algorithm applies local operators to evolve and explore input H-cycle improvements with respect to our objectives.

```
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 ((1 + 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;
}
```

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 [HS85]) 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 1. 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 `ShuffleIter=ImproveIter=`$8N$, `ThreshHigh`=0.5, `ThreshLow`=1, `Mod`=400 and $C = 2, K = 12, I = 3$.

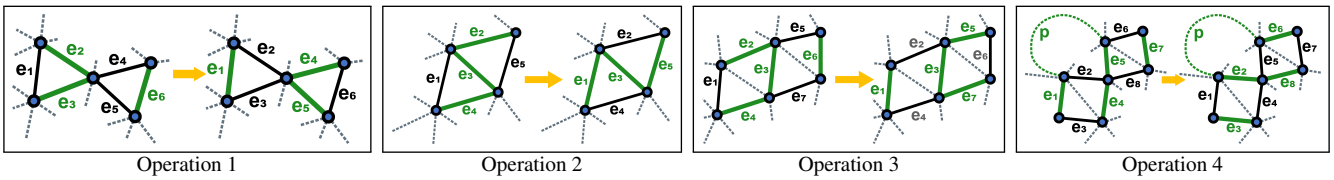


Figure 9: *Illustration of the four local operations evolving an H-cycle while preserving its Hamiltonian property. Thick green edges belong to the cycle; solid black edges do not belong to the cycle. The state of other edges (gray dashed) has no influence.* ***Operation 1:*** *two triangles given by the edges* (e_1,e_2,e_3) *and* (e_4,e_5,e_6) *share one node. We change the cycle edges from* (e_2,e_3,e_6) *to* (e_1,e_4,e_5). ***Operation 2:*** *two triangles given by the edges* (e_1,e_3,e_4) *and* (e_2,e_3,e_5) *share one edge* e_3. *We change the cycle edges from* (e_4,e_3,e_2) *to* (e_1,e_3,e_5). ***Operation 3:*** *two rhombuses enclosed by edges* (e_1,e_2,e_3,e_4) *and* (e_3,e_5,e_6,e_7) *share a common edge* e_3. *We change the cycle edges from* (e_2,e_3,e_4,e_6) *to* (e_1,e_3,e_5,e_7). ***Operation 4:*** *two rhombuses enclosed by edges* (e_1,e_2,e_3,e_4) *and* (e_5,e_6,e_7,e_8) *share a common node and there exists a path in the green cycle (indicated by the green dashed curve) that connects two nodes and that does not traverse any other node in the illustration. We change the cycle edges from* (e_1,e_4,e_5,e_7) *to* (e_2,e_3,e_6,e_8).

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. 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 tessellation. 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, we apply a few explicit smoothing steps before computing the target length. 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:

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

After applying (5), the target length L_0 becomes the length of $\mathcal{T}$. The geometric optimizer starts from this smoothed cycle.

Optimization We then minimize the geometric objective $\mathcal{E}_{\text{Geo}}$ defined in (2). We encourage the reader to have a look at the different terms of this objective in the appendix. Our geometric optimizer is supported by a quasi Newtonian solver L-BFGS [LN89]. 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. Figure 10 illustrates the impact of the initial smoothing (5) and the optimizer over the cycle geometry.

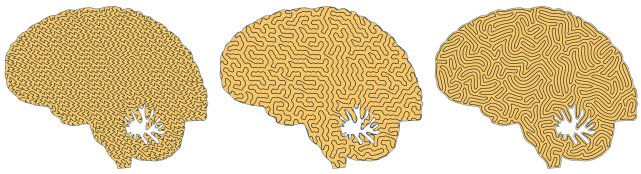


Figure 10: *Result of our geometry optimization over the brain model with an isotropy objective.*

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 enables 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 11 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 12 shows the effect of disabling the optimization of global terms ($\mathcal{E}_{\text{Ori}}, \mathcal{E}_{\text{Ali}}$). As can be seen, these terms are critical in obtaining a trajectory well aligned with the target field.

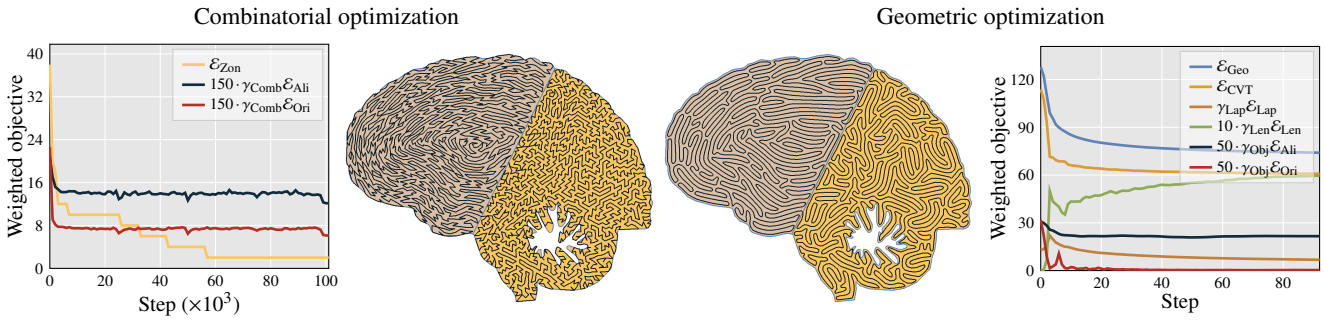


Figure 11: *Two-stage optimization of shape depicting a brain. The trajectory of the left part is under an alignment objective, while the trajectory in the right part follows an isotropy objective. The plots depict the evolution of the value of each objective function during the optimization steps. Note that some values are multiplied by a constant factor in order to improve the visualization.*

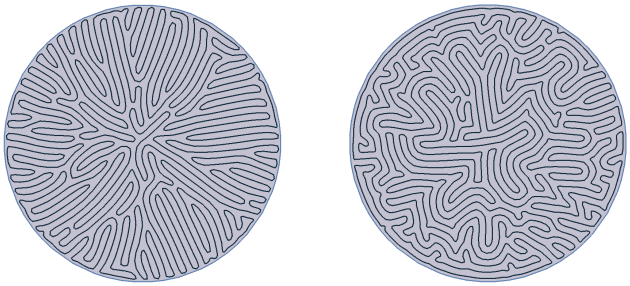


Figure 12: *Radial vector field in a disc.* ***Left:*** *Our result.* ***Right:*** *Without any global optimization of orientation (initial graph construction directly followed by geometric optimizer without $\mathcal{E}_{Ali}$, $\mathcal{E}_{Ori}$). The result no longer closely follows the desired orientation.*

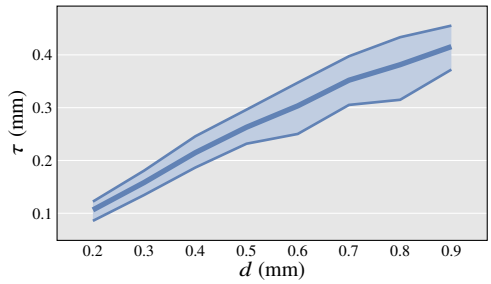


Figure 13: *Correlation between d (controlling the seed density – see Figure 3) and the interspace τ around the optimized cycle. The thick curve is the mean observed over 30 models using different objectives. The top and bottom curves are the reported min/max values of the mean across models.*

Figure 13 reveals 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 relationship between d and the final interspace behaves consistently across examples and can be predicted reasonably well using the curve of Figure 13. We can also observe that the variation across models is slight (within ±0.1 mm of each other's).

In addition, Figure 14 shows that the interspace values follow a (tilted) 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. One can also notice that the variance of the anisotropy is lower, as it is easier to obtain a constant interspace with zigzags.

Figure 15 demonstrates the effect of the Laplacian term γ_{Lap} (see Section A). While both results are compelling, using the Laplacian term avoids sharp turns, effectively smoothing the cycle.

Figure 16 shows some timing results. Both optimizers can execute iterations in a reasonable time, with a roughly polynomial correlation with respect to d.

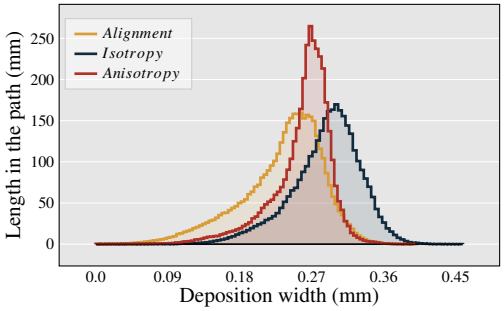


Figure 14: *Interspace distribution over the brain model with d = 0.5mm, for different objectives.*

4.2. 3D printed results

We carried out 3D printing of parts optimized with our method. We employed filament fused fabrication with PLA.

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

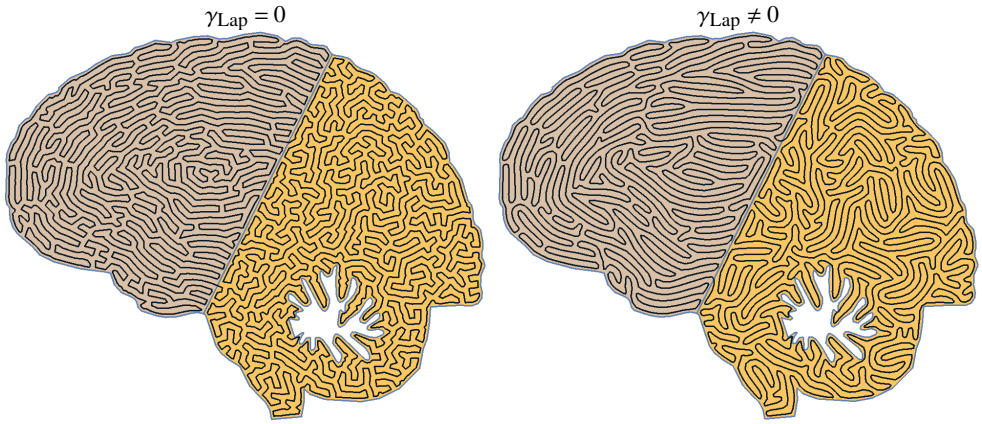


Figure 15: *Optimization results showing the effect of using or not the smoothing term γ_{Lap}.*

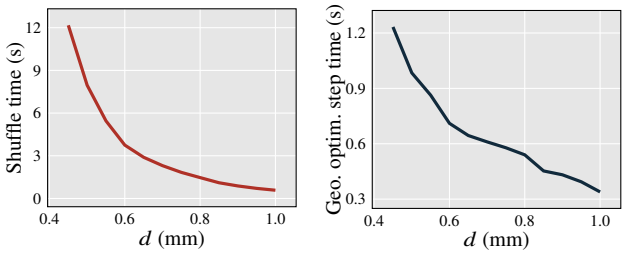


Figure 16: *Timings of the brain model (see Figure 11).* ***Left****: computing time of the function* `Shuffle` *(Fig. 1) with respect to d.* ***Right****: computing time of a single iteration of the L-BFGS solver in the geometric step.*

We use 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 toothpaste-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 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 17 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 material flow, leading to nearly optimal operational conditions for the extrusion process.

The *leaf* model shown printed in Figure 18 has many narrow regions, and an internal skeletal structure. This shows that despite being a single cycle, the trajectory can fill narrow regions orthogonally to their main directionality – this would be difficult to achieve with growth techniques that have to fit an even number of trajectories everywhere (see Section 2). The skeleton is well captured despite the multi-material transitions.

Figures 19 and 20 show other examples of shapes with narrow spaces and multi-material transitions. The *batman* logo uses an alignment objective parallel to the boundaries. The *zebra* uses anisotropy for the background and alignment along boundaries within the zebra stripes.

Dense plates We compared two methods on an isotropic cycle filling a square to evaluate how dense our prints can be. Isotropy results in a 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. The print speed is slightly impacted by the type of infill due to the acceleration limits. Comparing on a CR10S, printing a 40x40 mm square with a speed target of 20mm/sec, the isotropic infill reaches an average deposition speed of 19.75 mm/sec versus 19.94 mm/sec for an anistropic one.

As shown in Figure 21 the per-segment area achieves a denser result, leaving a few gaps. 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 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 around 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 must also be within reasonable margins of the nominal spacing (obtained as total area divided by cycle length).

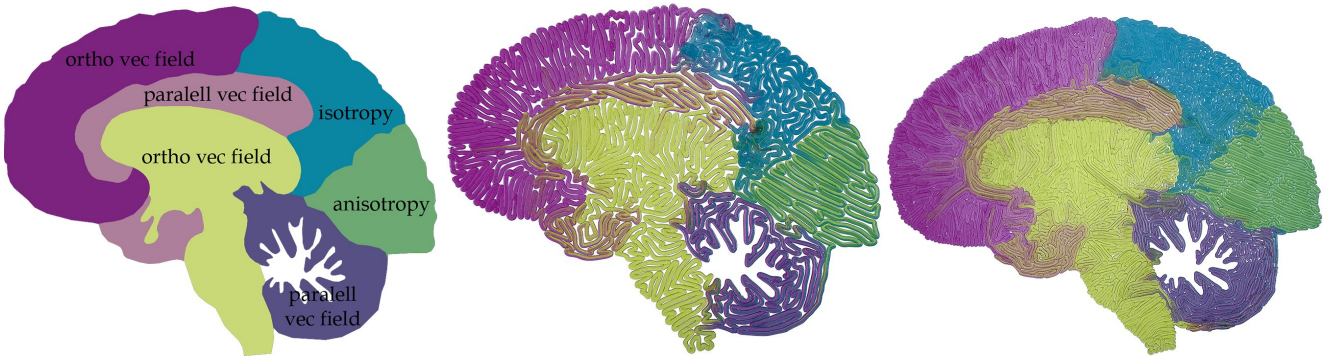


Figure 17: ***Left:*** *input specifying zones and objective selection in each. Parallel vec field means that we align the path with a vector field parallel to the zone border and ortho vec field means that we use alignment with a vector field orthogonal to the border.* ***Middle:*** *cycle optimized for a width of* 1.7*mm*. ***Right:*** *cycle optimized for a width of* 0.8*mm*.

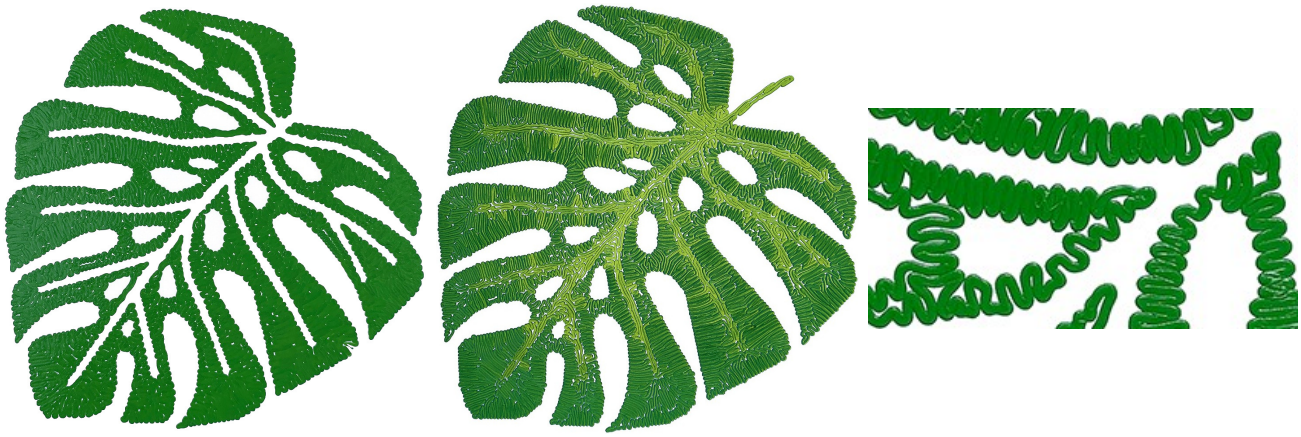


Figure 18: ***Left:*** *Single material leaf printed with a constant width smaller than nominal to appreciate the cycle folds. Note how the cycle goes around holes with a single folded trajectory in the center part.* ***Middle:*** *Multi-material version using a narrower spacing, with the inner skeleton in a different material.* ***Right:*** *Closeup. Note how the cycle locally has a single trajectory around the holes. 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.*

3D parts Beyond plates, we print 3D objects using our technique. Figure 22 compares three printed versions of the CuteOcto model, where our method is used in isolation and compared with a standard zigzag infill, also used in isolation. We print the CuteOcto first using our isotropic fill and next using a vector field that encourages the deposition to conform to the surface. In both cases, we obtain a solid object that prints with no travel moves beyond those strictly necessary (layers with two disjoint areas, e.g., front paw).

As we use the infill in isolation, we can see that the turns along the surface produce small depressions. However, even in the isotropic case, the surface quality remains better than that produced by a standard zigzag infill, creating turns everywhere along the boundary. Using the vector field (Figure 22, middle) further encourages the paths to circulate along the part boundary.

In standard practice, to obtain a smooth surface, an external contour (*perimeter*) is used, with the infill hidden behind. Since our infill is a cycle, and since external contours would also be cyclic, it would be simple to reconnect everything in a global cyclic trajectory, still producing a continuous extrusion.

Our technique is also useful on multi-material parts, as shown in Figure 23. Here, we use zoning to transition a cyclic isotropic infill between two material mixtures. The extrusion is continuous, and the transition between mixtures occurs in a user-specified zone (Figure 23, bottom left). In contrast, a two nozzle print on an Ultimaker 3 requires switching every layer and printing an extra tower to prime the nozzles, wasting material and time. Finally, an advantage of using mixtures is to print with none of the original colors: as long as the mixture is stable (e.g., 30% - 70% mix), a consistent color is produced [SMB*19].

4.3. Performance

We measure performance on a Core i7-10750H laptop (2.6GHz) with 16 GB of RAM. Our implementation is a non-optimized research prototype.

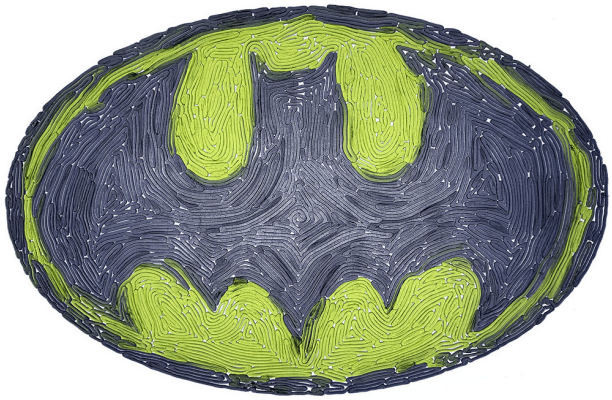


Figure 19: *Multi-material printing of the Batman logo following the parallel vector field.*

Figure 20: *Zebra printed with black and white filaments.*

We report in Table 2 some timings. The combinatorial optimizer runs several threads in parallel (12). On CuteOcto, timings are the total for all layers (noting that CuteOcto has a large base followed by smaller layers above). Init and geometric optimizers are parallelized when multiple slices are computed. In all cases, the total computation time is below fabrication time. The algorithm could be streamed as slices are processed and sent to the printer and multiple slices processed on different cores. The stopping criteria offer a time-quality tradeoff (see Figure 16). We use two settings: one for cases with zoning and one for single objective cases (the combinatorial optimizer typically converges faster on these).

In terms of complexity, initializing the graph is $O(N)$, where N is the number of cycle vertices, $N = O(\frac{area(P)}{d^2})$. Each iteration of the combinatorial optimizer is $O(N^2)$. The N^2 is due to the worst linear case of updating the connectivity check data structure of the transpose operator (Figure 9, rightmost). Each iteration of the geometric optimizer iteration is $O(Nlog(N))$.

Name	init	combinatorial	geometric	dimensions
Section	(3.2-3.3.2)	(3.3.3)	(3.4)	
Brain (Fig. 11)	4 sec	8 sec	12 sec	65x52mm
Leaf (Fig. 18, mid.)	54 sec	73sec	242 sec	146x160mm
CuteOcto (282 layers)	9 sec	462 sec	220 sec	82x81x56mm

Table 2: *Timings of different models (averaged, five runs).*

5. Discussion

In this section, we provide comparisons and additional discussion.

5.1. Comparisons

Figure 24 compares extrusion quality between a zigzag infill and our technique in a single layer, printed with flexible filament. The travel moves required by the standard zigzag produce artifacts, as the print head leaves marks and material oozes along with some travel moves. In contrast, our cyclic infill produces a similarly oriented deposition without travel moves. Other examples of near zigzag anisotropic fills are visible in Figure 1, **(b)**. Figure 22 also compares our infill to a standard zigzag on full 3D prints. Figure 25 provides a visual comparison to [ZGH*16] and [PS06], two techniques producing cyclic trajectories.

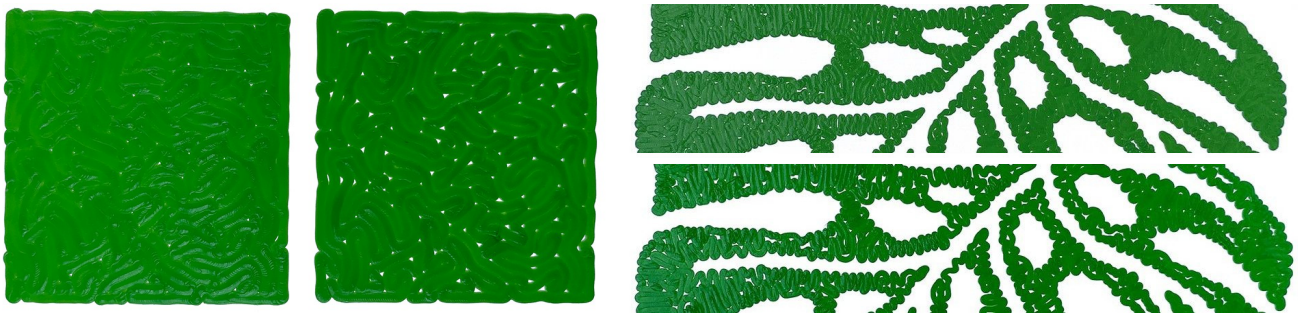
Fermat spiral. The method of [ZGH*16] constructs a concentric spiral explicitly. Our approach produces a trajectory that is well aligned with the concentric field and also exhibits a more even inter-spacing. In particular, the offsetting used during the Fermat spiral construction tends to accumulate gaps along the medial axis, a defect that would repeat and align throughout layers. Thanks to their randomized nature, our trajectories are less subject to defect accumulation across layers. We can also produce a cyclic trajectory following the direction orthogonal to the concentric field (the green region in Figure 25, see also Figure 12 left), which cannot be achieved with Fermat spirals.

Labyrinths and mazes. Our technique produces an outcome similar to that of Pedersen and Singh [PS06], following the field closely. It can also be seen that our result crosses between the three zones four times (two between green and orange – once around the arms, once around the feet – and two between orange and gray) which is the minimum possible. A growth approach will always cross at least six times (see also Figure **??**).

5.2. Limitations of growth approaches

Growth approaches offer multiple controls, some of which are similar to our goals, anisotropy, and zoning in particular. However, we are not aware of any method explicitly targeting isotropy.

In particular, the method of Pedersen and Singh [PS06] demonstrates impressive artistic results using these controls. This technique is primarily designed as an interactive authoring tool. Some of its properties do not translate to the automatic generation of trajectories in a large number of layers, as we discuss now.



(a) (b) (c)

Figure 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 18.* ***(c) bottom:*** *closeup on the same leaf printed without the per-segment area.*

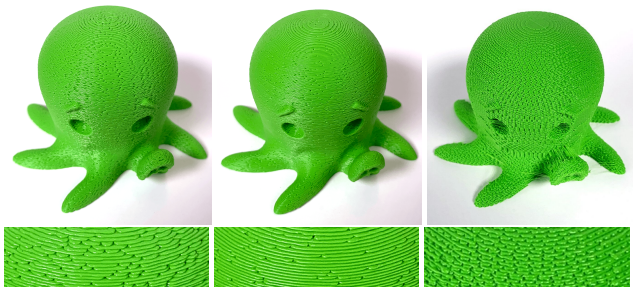


Figure 22: *CuteOcto model 3D printed with our isotropic (**left**), our vector concentric fills (**middle**), and standard zigzag for comparison (**right**). Please note that in standard prints, the infill 'sides' are usually hidden beneath an external contour parallel perimeter, but in this example we compared the infills in isolation.*

In the discussion below, we compare the number of iterations between methods. The growth process starts from a circle, and our method starts from our combinatorial initialization that takes $O(N)$ with N number of points along the trajectory. Our geometric optimizer and the growth process of Pedersen and Singh both have an $O(N\log(N))$ complexity per iteration. However, our approach performs a quasi-Newton optimization that converges towards a (local) minima in a few iterations. In contrast, the growth process does not converge: in our experience, the result keeps moving even after invading the entire domain. For the growth process, we used the implementation `https://github.com/twentylemon/organic-labyrinth` that we improved and extended.

A drawback of growth processes is to take significant amounts of time to evolve the curve, especially as the growth rate has to be limited to a fraction of the target inter-spacing to guarantee a stable growth [PS06]. To alleviate this issue, the growth is first performed with a larger inter-space that is then progressively reduced, as shown in Figure 26a. This is done manually in an interactive setting. In Figure 26a we decreased the interspace at manually selected points. In contrast, our technique immediately starts at the chosen inter-space. As a result, in Figure 26b our approach converges in 170 iterations versus 1700 for the multi-scale growth process.

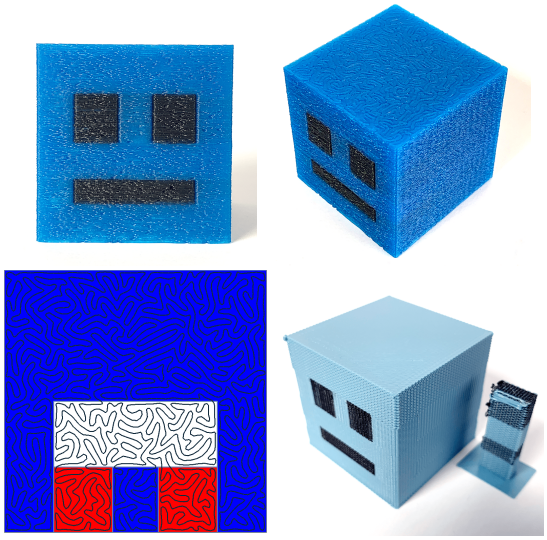


Figure 23: *Multi-material cube.* ***Top:*** *Views of the print obtained with a mixing extruder, the transition occurs within the part. Note the isotropic infill pattern on the top.* ***Bottom-left:*** *The white zone has to be traversed when going from blue (mixture A) to red (mixture B), and is where mixture transition occurs. The user can freely place it within each layer.* ***Bottom-right:*** *same model using a zigzag infill, printed on a two-extruders Ultimaker 3. Note the prime tower, requiring extra movements at each layer.*

This advantage is further increased in geometries with narrow passages. These hinder the use of a decreasing inter-space: with a large inter-space, the curve cannot invade past the narrow regions. This is illustrated in Figure 26c. Here the growth has to start with an inter-space smaller than the minimal feature size to reach every location. A more elaborate initialization – beyond a simple circle – would alleviate this issue. However, doing so in a general setting where the shape contains holes is not trivial and is what our combinatorial optimizer provides. As a result, in Figure 26d our approach converges in 260 iterations versus 110000 for the growth process.

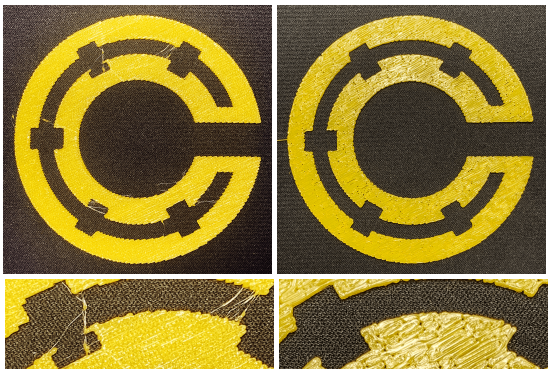


Figure 24: *Comparing a standard zigzag strategy and our anisotropic infill.* ***Left:*** *While the zigzag is perfectly regular, it does not allow for continuous extrusion. Travel moves create marks and oozing, that damage the final part.* ***Right:*** *None of these issues are present on our fully continuous 3D prints.*

Please also observe that continuously growing from a simple closed curve limits the cycles that can be achieved. For instance, this implies there will always be an even number of trajectories between two outer contours, making trajectories such as the closeup shown in Figure 18 unfeasible. This further limits following specific orientations in narrow features, such as the folds in Figure 18 which would not be possible with two trajectories side by side.

Finally, it is worth noting that after combinatorial initialization, we could use the optimizer of Pedersen and Singh [PS06]. However, the dynamic evolution process of the curve relies on a number of parameters that are difficult to tune to achieve specific properties (e.g., anisotropy, isotropy). This is not a major issue in an interactive setting where adjustments are performed live but proves difficult for trajectory computations where the parameters cannot be easily adjusted for every single situation. In contrast, our curves are globally optimized.

5.3. Influence of orientation on mechanical properties

Multiple factors affect the mechanical properties of the fabricated layers. Among the most important are the bonding strength, extrusion temperature, printing speed, extrusion rate, nozzle and filament diameter, viscosity, and thermal expansion coefficient of the deposition material [ATB*18, CdSC19]. A tightly controlled manufacturing environment and protocols are necessary to obtain consistent mechanical results.

Fortunately, several prior works studied this and indicate that the orientation of the deposition paths plays a role in the mechanical properties of each layer. For instance, Villacres et al. [VNA18] found the orientation of parallel paths with an infill density close to 100% has a monotonic relationship with stiffness: Young's modulus is highest for the parallel direction and lowest for the orthogonal direction. Recent studies have also been interested in producing paths that exhibit transversely isotropic mechanical properties, as they offer mechanical advantages compared to the traditional parallel paths [LXM*19]. Other examples of works using the mechanical influence of trajectory orientations are [SIM16, TM17, LY17, KWW19a, XLM20, FZZ*20].

A promising direction is to *increase* the influence of orientation, for instance, using on-purpose non-optimal deposition flow and temperature. This would potentially make bonding between neighboring paths weaker, increasing flexibility in these directions.

5.4. Limitations

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 [HKD*20]). However, some tiny porosities remain in challenging cases (see the leaf in Figure 21).

Our current graph construction tends to create edges that take many orientations, even when biased by the needles. This puts more pressure on the geometric optimizer. Other graph constructions, perhaps inspired by quad-meshing, could alleviate this issue.

6. Conclusions

Our technique offers control over material deposition orientation while manufacturing a solid layer along a strictly continuous extrusion trajectory. It *globally optimizes* a space-filling trajectory for isotropy, anisotropy, vector field alignment, and zoning. As we have demonstrated, this is of particular interest for multi-material prints and aesthetic purposes. Continuity is also an essential consideration for clay, ceramics, silicon, and concrete materials.

Our framework also supports a wide range of optimization objectives. We believe this will prove helpful for Additive Manufacturing processes using challenging materials, such as the fabrication of plates having isotropic elastic behavior [LXM*19].

Acknowledgements

This research was initiated at the 18th Bellairs Workshop on Computational Geometry, February 16-22, 2019, co-organized by S. Lazard and S. Whitesides. We thank the other workshop participants for stimulating conversations and Pierre-Alexandre Hugron for his help with 3D printing. This work was partly supported by Lorraine Université d'Excellence, ANR-15-IDEX-04-LUE, ANR-17-CE10-0002, and NSERC Discovery Grant.

References

[AHMS96] ARKIN E. M., HELD M., MITCHELL J. S., SKIENA S. S.: Hamiltonian triangulations for fast rendering. *The Visual Computer 12*, 9 (1996), 429–444. doi:10.1007/bf01782475. 2

[ALL*18] ATTENE M., LIVESU M., LEFEBVRE S., FUNKHOUSER T., RUSINKIEWICZ S., ELLERO S., MARTÍNEZ J., BERMANO A. H.: Design, representations, and processing for additive manufacturing. *Synthesis Lectures on Visual Computing: Computer Graphics, Animation, Computational Photography, and Imaging 10*, 2 (2018), 1–146. doi:10.2200/s00847ed1v01y201804vcp031. 1

[ATB*18] ABBOTT A., TANDON G., BRADFORD R., KOERNER H., BAUR J.: Process-structure-property effects on ABS bond strength in fused filament fabrication. *Additive Manufacturing 19* (2018), 29–38. doi:10.1016/j.addma.2017.11.002. 14

[AXG*13] AKLEMAN E., XING Q., GARIGIPATI P., TAUBIN G., CHEN J., HU S.: Hamiltonian cycle art: Surface covering wire sculptures and duotone surfaces. *Computers & graphics 37*, 5 (2013), 316–332. doi:10.1016/j.cag.2013.01.004. 2

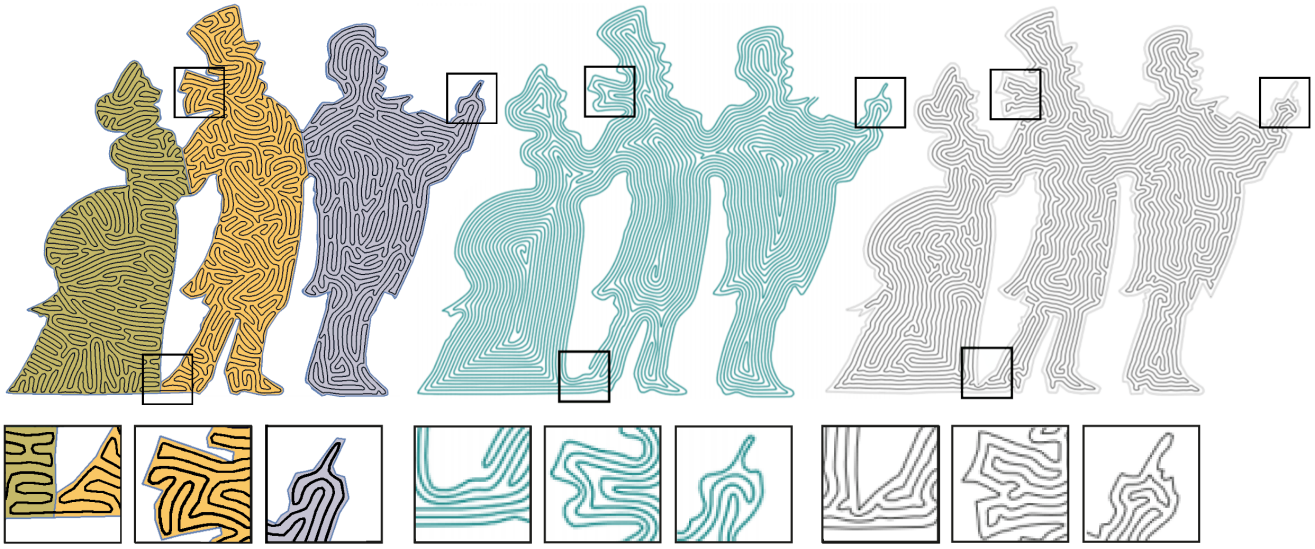
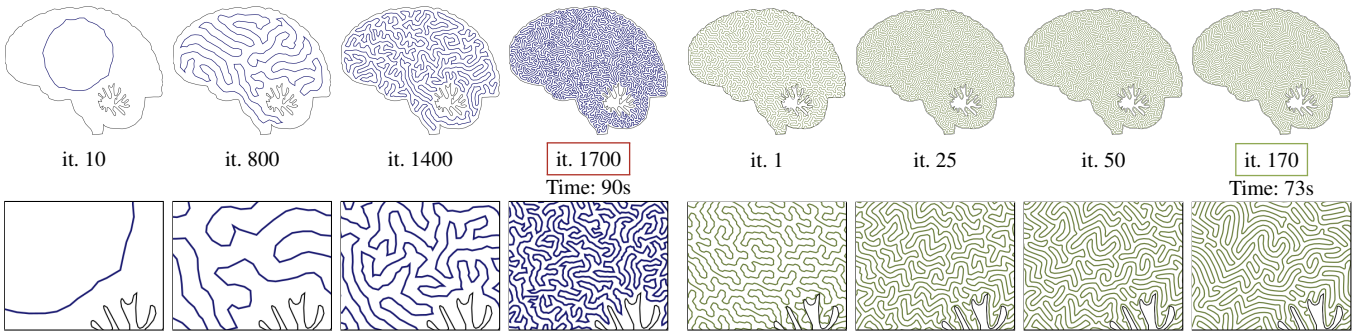
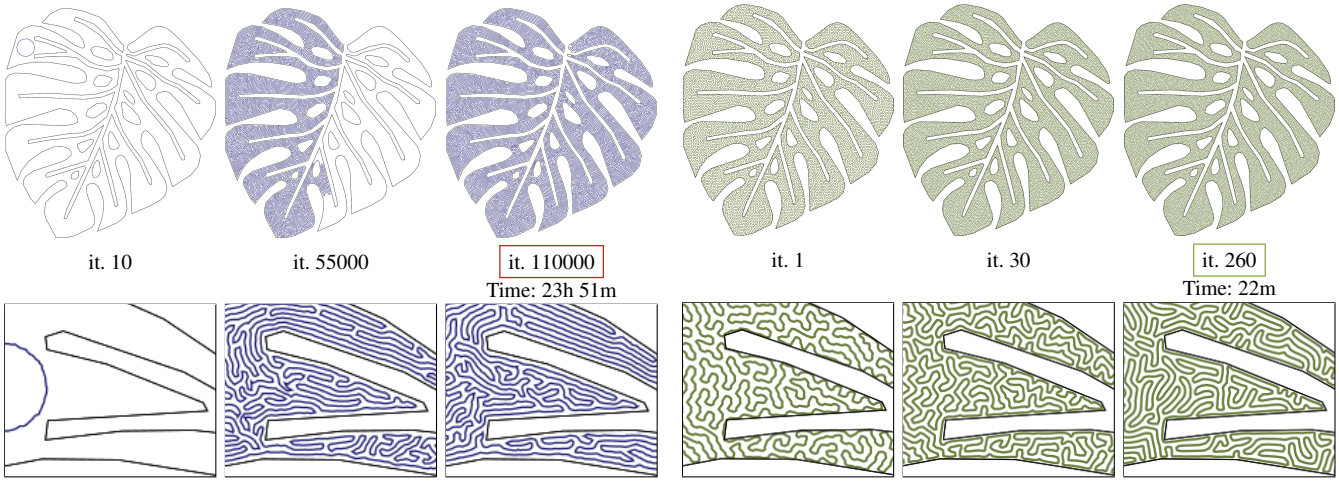


Figure 25: ***Left:*** *Result from our technique, with different objectives: vector field orthogonal to border (green), isotropic (orange) and concentric vector field (gray). Note the single crossing near the arms and the feet. The gray part is the one optimized to match the orientation of the two other methods.* ***Middle and right:*** *Results from respectively [ZGH*16] and [PS06] (images from papers).*



(a) *Multi-scale growth process* **(b)** *Our approach*



(c) *Growth process* **(d)** *Our approach*

Figure 26: *Comparison between a growth process (a,c) and our approach (b,d). Please refer to the text for details.*

[BBDL01] BIEDL T. C., BOSE P., DEMAINE E. D., LUBIW A.: Efficient algorithms for Petersen's matching theorem. *Journal of Algorithms 38*, 1 (2001), 110–134. doi:10.1006/jagm.2000.1132. 5

[BH04] BOSCH R., HERMAN A.: Continuous line drawings via the traveling salesman problem. *Operations research letters 32*, 4 (2004), 302–303. doi:10.1016/j.orl.2003.10.001. 2

[Bos10] BOSCH R.: Simple-closed-curve sculptures of knots and links. *Journal of Mathematics and the Arts 4*, 2 (2010), 57–71. doi:10.1080/17513470903459575. 2

[CdSC19] COSTA A. E., DA SILVA A. F., CARNEIRO O. S.: A study on extruded filament bonding in fused filament fabrication. *Rapid Prototyping Journal* (2019). doi:10.1108/rpj-03-2018-0062. 14

[CLEC20] CHENG P., LIU W. K., EHMANN K., CAO J.: Enumeration of additive manufacturing toolpaths using Hamiltonian paths. *Manufacturing Letters 26* (2020), 29–32. doi:10.1016/j.mfglet.2020.09.008. 2

[CSG*17] CHEN Z., SHEN Z., GUO J., CAO J., ZENG X.: Line drawing for 3D printing. *Computers & Graphics 66* (2017), 85–92. doi:10.1016/j.cag.2017.05.019. 2

[DCOM00] DAFNER R., COHEN-OR D., MATIAS Y.: Context-based space filling curves. *Computer Graphics Forum 19*, 3 (2000), 209–218. doi:10.1111/1467-8659.00413. 2

[DSL18] DEHAECK S., SCHEID B., LAMBERT P.: Adaptive stitching for meso-scale printing with two-photon lithography. *Additive Manufacturing 21* (2018), 589–597. doi:10.1016/j.addma.2018.03.026. 1, 2

[Edm65] EDMONDS J.: Paths, trees, and flowers. *Canadian Journal of mathematics 17* (1965), 449–467. doi:10.1007/978-0-8176-4842-8_26. 5

[FZZ*20] FANG G., ZHANG T., ZHONG S., CHEN X., ZHONG Z., WANG C. C.: Reinforced FDM: multi-axis filament alignment with controlled anisotropic strength. *ACM Transactions on Graphics 39*, 6 (2020), 1–15. doi:10.1145/3414685.3417834. 2, 14

[GE04] GOPI M., EPPSTEIN D.: Single-strip triangulation of manifolds with arbitrary topology. *Computer Graphics Forum 23*, 3 (2004), 371–379. doi:10.1145/997817.997888. 2, 4, 5

[GLLR11] GURUNG T., LUFFEL M., LINDSTROM P., ROSSIGNAC J.: LR: compact connectivity representation for triangle meshes. In *ACM SIGGRAPH 2011 Papers* (2011). doi:10.1145/1964921.1964962. 2

[GVCD15] GIANNATSIS J., VASSILAKOS A., CANELLIDIS V., DEDOUSSIS V.: Fabrication of graded structures by extrusion 3D printing. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (2015), pp. 175–179. doi:10.1109/ieem.2015.7385631. 2

[HHLT19] HERGEL J., HINZ K., LEFEBVRE S., THOMASZEWSKI B.: Extrusion-based ceramics printing with strictly-continuous deposition. *ACM Transactions on Graphics 38*, 6 (2019). doi:10.1145/3355089.3356509. 2

[HKD*20] HORNUS S., KUIPERS T., DEVILLERS O., TEILLAUD M., MARTÍNEZ J., GLISSE M., LAZARD S., LEFEBVRE S.: Variable-width contouring for additive manufacturing. *ACM Transactions on Graphics 39* (2020). doi:10.1145/3386569.3392448. 14

[HS85] HOCHBAUM D. S., SHMOYS D. B.: A best possible heuristic for the k-center problem. *Mathematics of Operations Research 10*, 2 (1985), 180–184. doi:10.1287/moor.10.2.180. 7

[JM20] JIANG J., MA Y.: Path planning strategies to optimize accuracy, quality, build time and material use in additive manufacturing: a review. *Micromachines 11*, 7 (2020), 633. doi:10.3390/mi11070633. 1, 2

[KB*05] KAPLAN C. S., BOSCH R., ET AL.: TSP art. In *Renaissance Banff: Mathematics, music, art, culture* (2005), Bridges Conference, pp. 301–308. doi:10.1515/9780691197036. 2

[KWW19a] KUBALAK J. R., WICKS A. L., WILLIAMS C. B.: Deposition path planning for material extrusion using specified orientation fields. *Procedia Manufacturing 34* (2019), 754–763. doi:10.1016/j.promfg.2019.06.209. 2, 14

[KWW19b] KUIPERS T., WU J., WANG C. C.: CrossFill: foam structures with graded density for continuous material extrusion. *Computer-Aided Design 114* (2019), 37–50. doi:10.1016/j.cad.2019.05.003. 2

[LB13] LÉVY B., BONNEEL N.: Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In *Proceedings of the 21st international meshing roundtable*. Springer, 2013, pp. 349–366. doi:10.1007/978-3-642-33573-0_21. 4

[LN89] LIU D. C., NOCEDAL J.: On the limited memory BFGS method for large scale optimization. *Mathematical programming 45*, 1-3 (1989), 503–528. doi:10.1007/bf01589116. 4, 5, 8

[LXM*19] LIN S., XIA L., MA G., ZHOU S., XIE Y. M.: A maze-like path generation scheme for fused deposition modeling. *The International Journal of Advanced Manufacturing Technology 104*, 1-4 (2019), 1509–1519. doi:10.1007/s00170-019-03986-7. 2, 14

[LY17] LIU J., YU H.: Concurrent deposition path planning and structural topology optimization for additive manufacturing. *Rapid Prototyping Journal 23*, 5 (2017), 930–942. doi:10.1108/rpj-05-2016-0087. 2, 14

[Nis20] NISHAT R. I.: *Reconfiguration of Hamiltonian cycles and paths in grid graphs*. PhD thesis, University of Victoria, 2020. 5

[NW17] NISHAT R. I., WHITESIDES S.: Bend complexity and Hamiltonian cycles in grid graphs. In *International Computing and Combinatorics Conference* (2017), Springer, pp. 445–456. doi:10.1007/978-3-319-62389-4_37. 5

[PBS18] PAPACHARALAMPOPOULOS A., BIKAS H., STAVROPOULOS P.: Path planning for the infill of 3D printed parts utilizing Hilbert curves. *Procedia Manufacturing 21* (2018), 757–764. doi:10.1016/j.promfg.2018.02.181. 1, 2

[PJYP14] P. M., J.-Y. H., P. M.: Toolpaths for additive manufacturing of functionally graded materials (FGM) parts. *Rapid Prototyping Journal 20* (2014), 511–522. doi:10.1108/rpj-01-2013-0011. 2

[PS06] PEDERSEN H., SINGH K.: Organic labyrinths and mazes. In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering* (2006), pp. 79–86. doi:10.1145/1124728.1124742. 2, 12, 13, 14, 15

[RDG11] RABIN J., DELON J., GOUSSEAU Y.: Transportation distances on the circle. *Journal of Mathematical Imaging and Vision 41*, 1-2 (2011), 147–167. doi:10.1007/s10851-011-0284-0. 19

[SC18] SHARP N., CRANE K.: Variational surface cutting. *ACM Transactions on Graphics 37*, 4 (2018). doi:10.1145/3197517.3201356. 2

[SIM16] STEUBEN J. C., ILIOPOULOS A. P., MICHOPOULOS J. G.: Implicit slicing for functionally tailored additive manufacturing. *Computer-Aided Design 77* (2016), 107–119. doi:10.1115/detc2016-59638. 2, 14

[SMB*19] SONG H., MARTÍNEZ J., BEDELL P., VENNIN N., LEFEBVRE S.: Colored fused filament fabrication. *ACM Transactions on Graphics 38*, 5 (2019), 1–11. doi:10.1145/3183793. 2, 11

[SRJG17] SOLER V., RETSIN G., JIMENEZ GARCIA M.: 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* (2017), pp. 562–571. 2

[SSMVL19] SKYLAR-SCOTT M. A., MUELLER J., VISSER C. W., LEWIS J. A.: Voxelated soft matter via multimaterial multinozzle 3D printing. *Nature 575* (2019). doi:10.1038/s41586-019-1736-8. 2

[Tau03] TAUBIN G.: Constructing hamiltonian triangle strips on quadrilateral meshes. In *Visualization and Mathematics III*. Springer, 2003, pp. 69–91. doi:10.1007/978-3-662-05105-4_4. 2

[TM17] TAM K.-M. M., MUELLER C. T.: Additive manufacturing along principal stress lines. *3D Printing and Additive Manufacturing 4*, 2 (2017), 63–81. doi:10.1089/3dp.2017.0001. 2, 14

[Vil03] VILLANI C.: *Topics in optimal transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. doi:10.1090/gsm/058. 4, 19

[VNA18] VILLACRES J., NOBES D., AYRANCI C.: Additive manufacturing of shape memory polymers: effects of print orientation and infill percentage on mechanical properties. *Rapid Prototyping Journal* (2018). doi:10.1108/rpj-03-2017-0043. 14

[WT13] WONG F. J., TAKAHASHI S.: Abstracting images into continuous-line artistic styles. *The Visual Computer 29*, 6-8 (2013), 729–738. doi:10.1007/s00371-013-0809-1. 2

[XJ18] XIAO X., JOSHI S.: Automatic toolpath generation for heterogeneous objects manufactured by directed energy deposition additive manufacturing process. *Journal of Manufacturing Science and Engineering 140* (2018), 1087–1357. doi:10.1115/1.4039491. 2

[XLM20] XIA L., LIN S., MA G.: Stress-based tool-path planning methodology for fused filament fabrication. *Additive Manufacturing 32* (2020), 101020. doi:10.1016/j.addma.2019.101020. 2, 14

[ZGH*16] ZHAO H., GU F., HUANG Q.-X., GARCIA J., CHEN Y., TU C., BENES B., ZHANG H., COHEN-OR D., CHEN B.: Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics 35*, 4 (2016), 1–10. doi:10.1145/2897824.2925958. 1, 2, 12, 15

Appendix A: Appendix

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:

$$\operatorname{dist}(x,S) = \inf_{y \in S} \|x-y\| \tag{6}$$

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

$$\int_{\mathcal{P}} \operatorname{dist}(x,\mathcal{T})^2 \, dx \tag{7}$$

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:

$$\mathcal{E}_{\text{CVT}} = \sum_i \int_{\mathcal{Z}_i} \operatorname{dist}(x, \mathcal{T} \cap \mathcal{Z}_i)^2 \, dx \tag{8}$$

Computing the intersection of $\mathcal{T}$ with $\mathcal{Z}_i$ leads to 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:

$$u = (1-t)\cdot \boldsymbol{v}_i + t \cdot \boldsymbol{v}_j \tag{9}$$

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

$$\mathcal{G}^{(i)}_{\text{CVT}} \leftarrow \mathcal{G}^{(i)}_{\text{CVT}} + (1-t)\cdot \mathcal{G}^{(u)}_{\text{CVT}} \qquad \mathcal{G}^{(j)}_{\text{CVT}} \leftarrow \mathcal{G}^{(i)}_{\text{CVT}} + t \cdot \mathcal{G}^{(u)}_{\text{CVT}} \tag{10}$$

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.

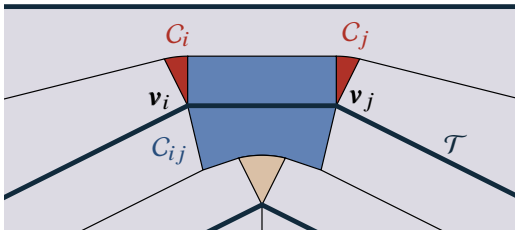


Figure 27: *A segment cell in blue and two point cells in red.*

We consider computing the objective and its gradient inside a single zone in the following. For simplicity and not introducing 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 $\mathcal{P}$ that are nearest to $\boldsymbol{v}_i$ than to any other point of $\mathcal{T}$:

$$C_i = \{x \in \mathcal{P} \mid \forall y \in \mathcal{T}, \|x-\boldsymbol{v}_i\| \leqslant \|x-y\|\} \tag{11}$$

- **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 $\mathcal{T}$:

$$C_{ij} = \left\{x \in \mathcal{P} \mid \forall y \in \mathcal{T}, \operatorname{dist}\left(x,]\boldsymbol{v}_i, \boldsymbol{v}_j[\right) \leqslant \|x-y\|\right\} \tag{12}$$

This partition is illustrated Figure 27. 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:

$$\begin{aligned}\mathcal{E}_{\text{CVT}} &= \sum_{i=1}^{N} \int_{C_i} \|x-\boldsymbol{v}_i\|^2 dx + \sum_{(ij)\in\mathcal{T}_E} \int_{C_{ij}} \operatorname{dist}\left(x,]\boldsymbol{v}_i, \boldsymbol{v}_j[\right)^2 dx \\ &= \sum_{i=1}^{N} \mathcal{E}_{\text{CVT},i} + \sum_{(ij)\in\mathcal{T}_E} \mathcal{E}_{\text{CVT},(ij)}\end{aligned} \tag{13}$$

Where $\mathcal{T}_E = \{(i,i_+) \mid i = 1,\ldots,N\}$ is the set of edges of the cycle. 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 tessellation.

We present how we derive the objective and gradient terms in the following. 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 $\mathcal{E}_{\mathrm{CVT},i}$ associated with one point cell C_i. First, we rewrite it as:

$$\begin{aligned}\mathcal{E}_{\mathrm{CVT},i} &= \int_{C_i} \|x-\boldsymbol{v}_i\|^2 dx \\ &= \int_{C_i} \|x\|^2 dx - 2\int_{C_i} (x.\boldsymbol{v}_i) dx + \int_{C_i} \|\boldsymbol{v}_i\|^2 dx \\ &= \int_{C_i} \left(x_x^2 + x_y^2\right) dx - 2\int_{C_i} \left(\boldsymbol{v}_{i,x} x_x + \boldsymbol{v}_{i,y} x_y\right) dx + \int_{C_i} \|\boldsymbol{v}_i\|^2 dx\end{aligned} \tag{14}$$

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 $\mathcal{E}_{\mathrm{CVT},i}$ with respect to $\boldsymbol{v_i}$ is:

$$\mathcal{G}_{CVT,i} = 2\left[\boldsymbol{v_i}\left(\int_{C_i} 1dx\right) - \left(\int_{C_i} x dx\right)\right] \tag{15}$$

New terms that appear in Equations 14 and 15 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. Shoelace formula gives us:

$$\int_{C_i} x_y^n dx = \frac{1}{(n+1)(n+2)} \sum_{(jk)\in C_i} (p_{k,x} - p_{j,x}) \sum_{l=0}^{n+1} p_{k,y}^l p_{j,y}^{n+1-l} \tag{16}$$

$$\int_{C_i} x_x^n dx = \frac{1}{(n+1)(n+2)} \sum_{(jk)\in C_i} (p_{j,y} - p_{k,y}) \sum_{l=0}^{n+1} p_{k,x}^l p_{j,x}^{n+1-l} \tag{17}$$

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 $\hat{w}$:

$$\hat{v} = \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\|\boldsymbol{v}_j - \boldsymbol{v}_i\|}, \qquad \hat{w} = R_{\pi/2}\hat{v} \tag{18}$$

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

$$m : x \mapsto \begin{pmatrix}(x-\boldsymbol{v}_i).\hat{v} & (x-\boldsymbol{v}_i).\hat{w}\end{pmatrix} \tag{19}$$

Two functions characterize 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 28. The support used for these two functions is the interval $]0,1[$. We then rewrite the segment objective as:

$$\begin{aligned}\mathcal{E}_{\mathrm{CVT},(ij)} &= \int_{C_{ij}} \mathrm{dist}\left(x,]\boldsymbol{v}_i, \boldsymbol{v}_j[\right)^2 dx \\ &= \|\boldsymbol{v}_j - \boldsymbol{v}_i\| \int_0^1 \int_{f(t)}^{g(t)} y^2 dy dt\end{aligned} \tag{20}$$

Gradients with respect to the two endpoints $\boldsymbol{v}_i, \boldsymbol{v}_j$ are given by:

$$\mathcal{G}^{(i)}_{\mathrm{CVT},(ij)} = -\|\boldsymbol{v}_j - \boldsymbol{v}_i\|\hat{w} \int_0^1 (1-t)\left(g(t)^2 - f(t)^2\right) dt \tag{21}$$

$$\mathcal{G}^{(j)}_{\mathrm{CVT},(ij)} = -\|\boldsymbol{v}_j - \boldsymbol{v}_i\|\hat{w} \int_0^1 t\left(g(t)^2 - f(t)^2\right) dt \tag{22}$$

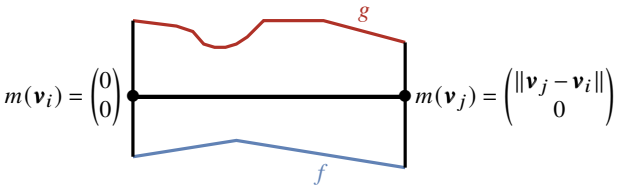


Figure 28: *The segment cell boundary is represented by two functions f, g.*

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 16), we get:

$$\mathcal{E}_{\mathrm{CVT},(ij)} = \frac{1}{12} \sum_{(kl)\in C_{ij}} (q_{l,x} - q_{k,x}) \sum_{m=0}^{3} q_{k,y}^m q_{l,y}^{3-m} \tag{23}$$

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

$$u_k = \frac{q_{k,x}}{\|\boldsymbol{v}_j - \boldsymbol{v}_i\|}, \qquad u_l = \frac{q_{l,x}}{\|\boldsymbol{v}_j - \boldsymbol{v}_i\|} \tag{24}$$

and we then have:

$$\mathcal{G}^{(i)}_{\mathrm{CVT},(ij)} = \hat{w} \sum_{(kl)\in C_{ij}} \frac{q_{k,x} - q_{l,x}}{12} \sum_{m=1}^{3} (4 - m u_k + (4-m)u_l)\, q_{k,y}^{m-1} q_{l,y}^{3-m} \tag{25}$$

$$\mathcal{G}^{(j)}_{\mathrm{CVT},(ij)} = \hat{w} \sum_{(kl)\in C_{ij}} \frac{q_{k,x} - q_{l,x}}{12} \sum_{m=1}^{3} (m u_k + (4-m)u_l)\, q_{k,y}^{m-1} q_{l,y}^{3-m} \tag{26}$$

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:

$$\mathcal{E}_{\mathrm{Len}} = (L - L_0)^2 = \left(\sum_{(ij)\in\mathcal{T}_E} \|\boldsymbol{v}_j - \boldsymbol{v}_i\| - L_0\right)^2 \tag{27}$$

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

$$\mathcal{G}^{(i)}_{\mathrm{Len}} = 2(L - L_0)\left(\frac{\boldsymbol{v}_i - \boldsymbol{v}_{i_+}}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_+}\|} + \frac{\boldsymbol{v}_i - \boldsymbol{v}_{i_-}}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_-}\|}\right) \tag{28}$$

Smoothing objective

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

$$\mathcal{E}_{\mathrm{Lap}} = \sum_{i=1}^{N} \left\| \boldsymbol{v}_i - \frac{1}{2}\left(\boldsymbol{v}_{i_+} + \boldsymbol{v}_{i_-}\right)\right\|^2 \tag{29}$$

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

$$\mathcal{G}^{(i)}_{\mathrm{Lap}} = 3\boldsymbol{v}_i + \frac{1}{2}\left(\boldsymbol{v}_{i_{++}} + \boldsymbol{v}_{i_{--}}\right) - 2\left(\boldsymbol{v}_{i_+} + \boldsymbol{v}_{i_-}\right) \tag{30}$$

Normalized weights used for the two previous objectives are:

$$\gamma_{\mathrm{Len}} = 0.4 \cdot \mathrm{area}(\mathcal{P})/N^2 \qquad \gamma_{\mathrm{Lap}} = 0.3 \cdot \mathrm{area}(\mathcal{P})/N \tag{31}$$

Alignment objective

Recall that we compute a vector field V by doubling the angle of U (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}:

$$\mathcal{V}_i = \int_{C_i} V(x)dx \qquad \mathcal{V}_{ij} = \int_{C_{ij}} V(x)dx \tag{32}$$

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

$$\begin{aligned}\mathcal{E}_{\text{Ali}} &= \sum_{i=1}^{N} \frac{\|\mathcal{V}_i\|}{Z}\left(\angle(\boldsymbol{v}_{i_+} - \boldsymbol{v}_{i_-}) - \frac{\angle\mathcal{V}_i}{2}\right)^2 \\ &+ \sum_{(ij)\in\mathcal{T}_E} \frac{\|\mathcal{V}_{ij}\|}{Z}\left(\angle(\boldsymbol{v}_j - \boldsymbol{v}_i) - \frac{\angle\mathcal{V}_{ij}}{2}\right)^2\end{aligned} \tag{33}$$

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

$$\angle v = \begin{cases} \arctan(v_y/v_x) & \text{if } v_x > 0 \\ \arctan(v_y/v_x) + \pi & \text{if } v_x < 0 \\ (\pi v_y)/(2|v_y|) & \text{if } v_x = 0 \end{cases} \qquad \nabla\angle v = \frac{R_{\pi/2}v}{\|v\|^2} \tag{34}$$

The values of $\angle\mathcal{V}_i$ and $\angle\mathcal{V}_{ij}$ in (33) 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:

$$Z = \left(\sum_{i=1}^{N}\|\mathcal{V}_i\| + \sum_{(ij)\in\mathcal{T}_E}\|\mathcal{V}_{ij}\|\right)\left(\int_{\mathcal{P}}\|V(x)\|dx\right)^{-1} \tag{35}$$

The integral of the norm over $\mathcal{P}$ is the total area in which the vector field is non-zero. This is useful for adequately weighing 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:

$$\begin{aligned}\mathcal{G}^{(i)}_{\text{Ali}} &= \frac{\|\mathcal{V}_{i_-}\|}{Z}\frac{R_{\pi/2}(\boldsymbol{v}_i - \boldsymbol{v}_{i_{--}})}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_{--}}\|^2}\left(\angle(\boldsymbol{v}_i - \boldsymbol{v}_{i_{--}}) - \frac{\angle\mathcal{V}_{i_-}}{2}\right) \\ &- \frac{\|\mathcal{V}_{i_+}\|}{Z}\frac{R_{\pi/2}(\boldsymbol{v}_{i_{++}} - \boldsymbol{v}_i)}{\|\boldsymbol{v}_{i_{++}} - \boldsymbol{v}_i\|^2}\left(\angle(\boldsymbol{v}_{i_{++}} - \boldsymbol{v}_i) - \frac{\angle\mathcal{V}_{i_+}}{2}\right) \\ &+ \frac{\|\mathcal{V}_{i_-i}\|}{Z}\frac{R_{\pi/2}(\boldsymbol{v}_i - \boldsymbol{v}_{i_-})}{\|\boldsymbol{v}_i - \boldsymbol{v}_{i_-}\|^2}\left(\angle(\boldsymbol{v}_i - \boldsymbol{v}_{i_-}) - \frac{\angle\mathcal{V}_{i_-i}}{2}\right) \\ &- \frac{\|\mathcal{V}_{ii_+}\|}{Z}\frac{R_{\pi/2}(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i)}{\|\boldsymbol{v}_{i_+} - \boldsymbol{v}_i\|^2}\left(\angle(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i) - \frac{\angle\mathcal{V}_{ii_+}}{2}\right)\end{aligned} \tag{36}$$

Vector field discretization Our field V is represented by a 2D array of size $W \times H$ whose elements are 2D vectors. 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:

$$\mathcal{V} = \sum_{(ij)\in C}(p_{j,x} - p_{i,x})\int_0^1\int_0^{p_{i,y}+t(p_{j,y}-p_{i,y})} V(p_{i,x}+t(p_{j,x}-p_{i,x}),y)dydt \tag{37}$$

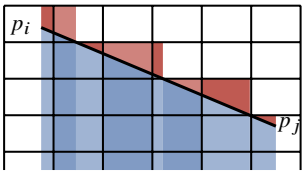


Figure 29: *The vector field's integral under a segment is obtained by decomposing it in smaller segments contained in single quads of the grid.*

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

$$SV[x][y] = \sum_{j=0}^{y} V[x][j] \tag{38}$$

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:

$$X\frac{\mathscr{W}}{W} \leqslant x_0, x_1 \leqslant (X+1)\frac{\mathscr{W}}{W} \tag{39}$$

$$Y\frac{\mathscr{H}}{H} \leqslant y_0, y_1 \leqslant (Y+1)\frac{\mathscr{H}}{H} \tag{40}$$

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

$$\begin{aligned}\int_0^1\int_0^{y_0+t(y_1-y_0)} &V(x_0+t(x_1-x_0),y)dydt \\ &= SV[X][Y] - \left(Y+1-\frac{y_0+y_1}{2\mathscr{H}}H\right)V[X][Y]\end{aligned} \tag{41}$$

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 29) 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).

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 [Vil03] 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 orientation distribution of $\mathcal{T}$ can be defined as:

$$\mathcal{D}_{\mathcal{T}} = \frac{1}{L}\sum_{(ij)\in\mathcal{T}_E}\|\boldsymbol{v}_j - \boldsymbol{v}_i\|\delta_{\angle(\boldsymbol{v}_j-\boldsymbol{v}_i)} \tag{42}$$

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 [RDG11] that the Wasserstein distance on a circle can be expressed as:

$$\mathscr{W}_2^2(\mathcal{U},\mathcal{D}_{\mathcal{T}}) = \inf_{\alpha}\int_0^1\left((F^{\alpha})^{(-1)}(t) - G_{\mathcal{T}}^{(-1)}(t)\right)^2 dt \tag{43}$$

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

$$F(\theta) = \theta/\pi, \quad G_{\mathcal{T}}(\theta) = \frac{1}{L} \sum_{(ij)\in\mathcal{T}_E} \|\boldsymbol{v}_j - \boldsymbol{v}_i\| \mathbb{1}_{\{\angle(\boldsymbol{v}_j - \boldsymbol{v}_i) < \theta\}} \quad (44)$$

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

$$(F^\alpha)^{(-1)}(t) = \pi(t+\alpha), \quad G_{\mathcal{T}}^{(-1)}(t) = \sum_{(ij)\in\mathcal{T}_E} \angle(\boldsymbol{v}_j - \boldsymbol{v}_i) \mathbb{1}_{\left\{t\in\left[T_{ij}^0, T_{ij}^1\right[\right\}} \quad (45)$$

where:

$$T_{ij}^0 = \frac{1}{L} \sum_{\substack{(kl)\in\mathcal{T}_E \\ \angle(\boldsymbol{v}_l - \boldsymbol{v}_k) < \angle(\boldsymbol{v}_j - \boldsymbol{v}_i)}} \|\boldsymbol{v}_l - \boldsymbol{v}_k\|, \qquad T_{ij}^1 = \frac{1}{L} \sum_{\substack{(kl)\in\mathcal{T}_E \\ \angle(\boldsymbol{v}_l - \boldsymbol{v}_k) \leqslant \angle(\boldsymbol{v}_j - \boldsymbol{v}_i)}} \|\boldsymbol{v}_l - \boldsymbol{v}_k\| \quad (46)$$

In the distance (43), 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:

$$\alpha_0 = \frac{1}{\pi}\mathbb{E}[\mathcal{D}_{\mathcal{T}}] - \frac{1}{2} \quad (47)$$

We then obtain:

$$\mathscr{W}_2^2(\mathcal{U}, \mathcal{D}_{\mathcal{T}}) = \sum_{(ij)\in\mathcal{T}_E} \int_{T_{ij}^0}^{T_{ij}^1} \left(\pi t + \pi\alpha_0 - \angle(\boldsymbol{v}_j - \boldsymbol{v}_i)\right)^2 dt \quad (48)$$

To simplify the following expressions, we let:

$$\Theta_{ij}^k = \pi T_{ij}^k + \pi\alpha_0 \quad (49)$$

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 (48), we get:

$$\mathcal{E}_{\text{Iso}}(\mathcal{T}) = \frac{1}{3\pi} \sum_{(ij)\in\mathcal{T}_E} \left(\Theta_{ij}^1 - \angle(\boldsymbol{v}_j - \boldsymbol{v}_i)\right)^3 - \left(\Theta_{ij}^0 - \angle(\boldsymbol{v}_j - \boldsymbol{v}_i)\right)^3 \quad (50)$$

As for the alignment objective, we only compute an approximation of the gradient by ignoring the variations of Θ_{ij}^k. Our approximation of the gradient with respect to $\boldsymbol{v}_i$ is as follows:

$$\mathcal{G}_{\text{Iso}}^{(i)} = \frac{R_{\pi/2}(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i)}{\pi\|\boldsymbol{v}_{i_+} - \boldsymbol{v}_i\|^2} \left[\left(\pi\Theta_{ii_+}^1 - \angle(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i)\right)^2 - \left(\pi\Theta_{ii_+}^0 - \angle(\boldsymbol{v}_{i_+} - \boldsymbol{v}_i)\right)^2\right] - \frac{R_{\pi/2}(\boldsymbol{v}_i - \boldsymbol{v}_{i_-})}{\pi\|\boldsymbol{v}_i - \boldsymbol{v}_{i_-}\|^2} \left[\left(\pi\Theta_{i_-i}^1 - \angle(\boldsymbol{v}_i - \boldsymbol{v}_{i_-})\right)^2 - \left(\pi\Theta_{i_-i}^0 - \angle(\boldsymbol{v}_i - \boldsymbol{v}_{i_-})\right)^2\right] \quad (51)$$

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:

$$\mathcal{E}_{\text{anIso}}(\mathcal{T}) = \frac{\pi^2}{12} - \mathcal{E}_{\text{Iso}}(\mathcal{T}) \quad (52)$$

The gradient of the anisotropic objective is simply the opposite of the isotropy gradient (51).

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

$$\mathcal{E}_{\text{Ori}} = \sum_i \text{area}(\mathcal{I}_i)\, \mathcal{E}_{\text{Iso}}(\mathcal{T} \cap \mathcal{I}_i) + \sum_i \text{area}(\mathcal{A}_i)\, \mathcal{E}_{\text{anIso}}(\mathcal{T} \cap \mathcal{A}_i) \quad (53)$$

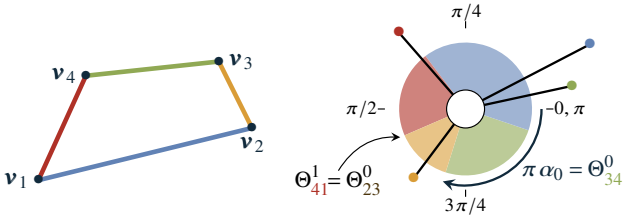


Figure 30: ***Left:*** *A path with 4 segments.* ***Right:*** *Transport map of the path orientation distribution to the uniform distribution. Sticks are Dirac located at the orientation of the segment having the same color. They are mapped to domains of* $[0,\pi[$ *represented by portions of the ring having an area proportional to the segment length.*

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 (10).

Finally, the 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:

$$\gamma_{\text{Obj}} = \frac{\text{area}(\mathcal{P})^3}{50 \cdot L_0^2} \left(\int_{\mathcal{P}} \|V(x)\| dx + \sum_i \text{area}(\mathcal{I}_i) + \sum_i \text{area}(\mathcal{A}_i)\right)^{-1} \quad (54)$$

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:

$$\mathcal{E}_{\text{Zon}} = \sum_{(i,j)\in\mathcal{T}_E} \sum_{k\neq l} \#\left([\boldsymbol{v}_i, \boldsymbol{v}_j] \cap \mathcal{Z}_k \cap \mathcal{Z}_l\right) \quad (55)$$

Where $\#S$ is the cardinal of a set S. 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:

$$\gamma_{\text{Comb}} = \frac{4}{\pi^2} \left(\int_{\mathcal{P}} \|V(x)\| dx + \sum_i \text{area}(\mathcal{I}_i) + \sum_i \text{area}(\mathcal{A}_i)\right)^{-1} \quad (56)$$