Orientable Dense Cyclic Infill for Anisotropic Appearance Fabrication

XAVIER CHERMAIN, CÉDRIC ZANNI, JONÀS MARTÍNEZ, PIERRE-ALEXANDRE HUGRON, and SYLVAIN LEFEBVRE, Université de Lorraine, CNRS, Inria, LORIA, France

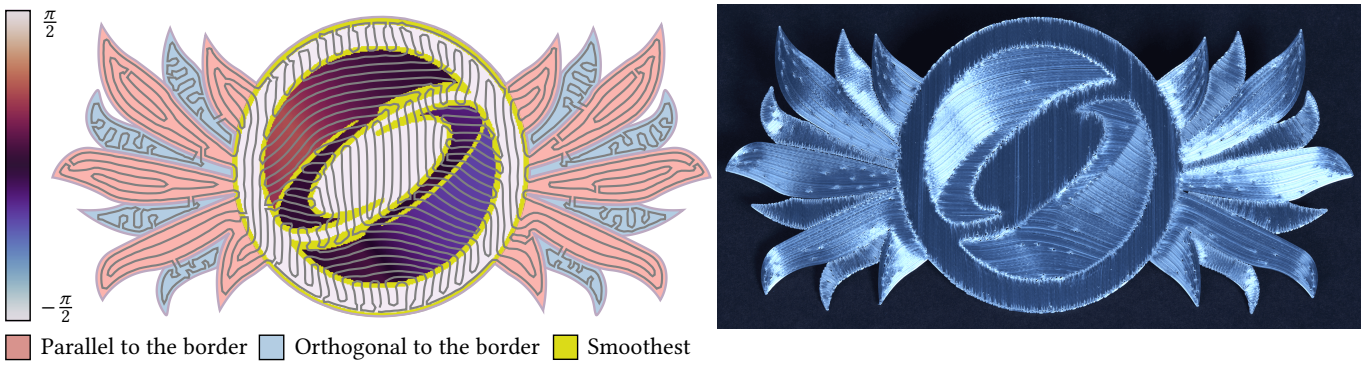


Fig. 1. We develop an efficient algorithm that produces an orientable dense cyclic infill by aligning a field of periodic functions, contouring it to obtain cycles, and connecting all cycles into one. We leverage this algorithm to print anisotropic appearances using fused filament fabrication. *Left*: the shape with purple boundaries is infilled with a cycle. The cycle's directions have four modes: parallel to the boundary (red area), orthogonal to the boundary (blue area), smoothest lines (yellow area), and constrained lines (color gradient area). Our algorithm is very flexible, allowing directions to be constrained everywhere (areas with a color gradient in the logo) or only within the vicinity of the boundary (blue, red, and yellow areas). Alignment with boundaries can also be constrained, as in this example. The grey cycle is the output of our algorithm (curve interspace objective: 2.5 mm). *Right*: Printed cycle with interspace set to 0.4 mm. The trajectory's directions determine the appearance, as extruded filaments exhibit anisotropic roughness.

We present a method to 3D print surfaces exhibiting a prescribed varying field of anisotropic appearance using only standard fused filament fabrication printers. This enables the fabrication of patterns triggering reflections similar to that of brushed metal with direct control over the directionality of the reflections. Our key insight, on which we ground the method, is that the direction of the deposition paths leads to a certain degree of surface roughness, which yields a visual anisotropic appearance. Therefore, generating dense cyclic infills aligned with a line field allows us to grade the anisotropic appearance of the printed surface. To achieve this, we introduce a highly parallelizable algorithm for optimizing oriented, cyclic paths. Our algorithm outperforms existing approaches regarding efficiency, robustness, and result quality. We demonstrate the effectiveness of our technique in conveying an anisotropic appearance on several challenging test cases, ranging from patterns to photographs reinterpreted as anisotropic appearances.

CCS Concepts: • **Applied computing** → **Computer-aided design**.

Additional Key Words and Phrases: fused filament fabrication, appearance fabrication, dense infill, shape optimization

ACM Reference Format:
Xavier Chermain, Cédric Zanni, Jonàs Martínez, Pierre-Alexandre Hugron, and Sylvain Lefebvre. 2023. Orientable Dense Cyclic Infill for Anisotropic Appearance Fabrication. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 13 pages. https://doi.org/10.1145/3592412

Authors' address: Xavier Chermain, xavier.chermain@inria.fr; Cédric Zanni, cedric.zanni@loria.fr; Jonàs Martínez, jonas.martinez-bayona@inria.fr; Pierre-Alexandre Hugron, pierre-alexandre.hugron@inria.fr; Sylvain Lefebvre, sylvain.lefebvre@inria.fr, Université de Lorraine, CNRS, Inria, LORIA, 615 Rue du Jardin-Botanique, Nancy, F-54000, France.

1 INTRODUCTION

Anisotropic appearances such as brushed metal offer a unique creative medium: as light reflects with a strong directional bias, patterns shimmer and move with the viewer's position, creating dynamic and lively imagery. Making brushed metal patterns is challenging, requiring mechanical abrasive tools to scratch surfaces and rare skills and expertise [Stango 2002].

We propose a novel method to fabricate plates visually resembling intricate brushed patterns on consumer-level 3D printers. We build upon a remarkable Fused Filament Fabrication (FFF) property: extruded plastic beads naturally exhibit a pronounced anisotropic surface roughness. We study and measure this optical property in Section 3. By orienting and aligning the deposition paths, we produce surfaces with precisely controlled anisotropic reflectance patterns according to user specifications.

Our patterns print as a single continuous extrusion path within a surface, leading to a high-quality and efficient deposition without transfer moves or interruptions of the deposition head. A continuous deposition is crucial for material extrusion quality [Hergel et al. 2019] and printing time [Papacharalampopoulos et al. 2018].

We encode the visual anisotropy with input maps akin to textures that define the orientation of the deposition paths. Such textures controlling the direction of anisotropy are commonly used in graphics software. We provide two complementary methods to define the input textures. The user can 1) use an existing design tool to create textures representing an anisotropic surface (e.g., Adobe Substance or Blender) and/or 2) request an anisotropy that is parallel/orthogonal to a shape boundary. From these inputs, our optimizer generates trajectories that precisely follow the requested anisotropy, are as

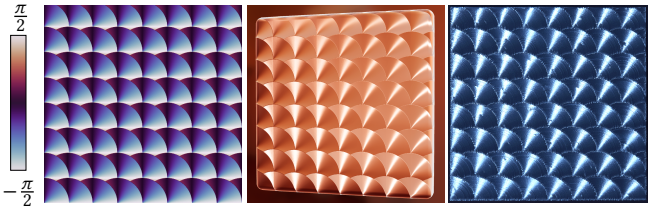


Fig. 2. Our method allows designers to print anisotropic appearances (right image) with an arbitrary direction of anisotropy (represented as an image texture on the left). This type of texture is used in physically based renderers to model complex anisotropic appearance (middle image obtained with the Filament renderer [Google 2023]).

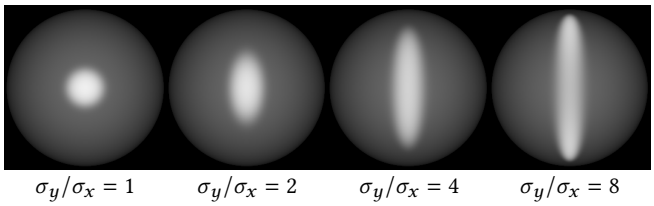


Fig. 3. Rendering of four plastic spheres with different roughness ratios, lighted by a point light at the camera position. The appearance is modeled by a physically based BSDF parameterized by roughnesses σ_x and σ_y (microfacet BSDF with anisotropic Beckmann distribution [Heitz 2014]). The highlight form depends on the anisotropy of the surface's roughness.

smooth as possible without constraints, and align with boundaries everywhere specified.

The computer graphics community has a rich literature devoted to the virtual design and rendering of complex anisotropic appearances, such as brushed metals [Westin et al. 1992], and in recent years has also developed studies revolving around additive manufacturing technologies. Our method tackles an interrelated problem: decorating flat surfaces with an anisotropic appearance using affordable and available FFF technologies (see Figure 2). By enabling the fabrication of anisotropic appearances, we empower texture artists with a method to materialize their virtual designs without incorporating additional features into the 3D printer.

Contributions. Our main contribution is an efficient and robust algorithm to compute a continuous deposition cycle that follows a given line field or other alignment conditions and confer a tailored anisotropic appearance to the result (Sections 4 and 5). The main algorithmic ingredients are a series of highly parallelizable optimizations that yield a cycle corresponding to the deposition path. Compared to the state-of-the-art [Bedel et al. 2022], we achieve a considerable speedup in execution time by up to three orders of magnitude. Our approach also aligns better with the orientation field: the number of turns is greatly reduced, improving the appearance (Section 6). In order to experimentally support our method, we empirically verify the correlation between the deposition path direction, surface roughness, and anisotropic appearance (Section 3).

We provide an open-source implementation of our method, the g-code file associated with each print, and the filament details at https://github.com/mfx-inria/anisotropic_appearance_fabrication.git.

2 RELATED WORK

This section reviews prior works on appearance fabrication and dense infills with orientable trajectories since our method covers these research topics.

2.1 Appearance fabrication

Surface geometry's microscopic details give rise to the visual appearance of many materials. Since the micro-scale geometry of a surface is usually highly complex, a statistical representation is more suitable to model the appearance. Traditionally, this representation has been the bidirectional scattering distribution function (BSDF), which represents how a surface reflects light [Bartell et al. 1981].

In computer graphics, anisotropic appearances are commonly modeled by microfacet BSDFs, which are simple and general-purpose scattering models based on geometric optics [Ribardière et al. 2019]. In addition, microfacet BSDFs are an excellent fit for real materials [Dong et al. 2016]. The surface's microscopic normal distribution mainly determines the shape of a microfacet BSDF. Simple analytical distributions parameterized by the surface roughness are usual. This surface statistic is the standard deviation of the slopes [Smith 1967] and can be measured on existing surfaces. The roughness is either equal (isotropic) or unequal (anisotropic) along the surface's tangent and bitangent directions [Heitz 2014]. This work focuses on anisotropic surfaces (brushed metals or plastics) where the roughness along the tangent direction is low, and the roughness along the bitangent is high (or inversely). This kind of surface gives elongated specular highlights, while materials with isotropic roughness give round highlights (see Figure 3).

Appearance fabrication using additive manufacturing is a growing research topic, where researchers target different technologies and appearances [Hullin et al. 2013].

Multi-material jetting [Sitthi-Amorn et al. 2015] (e.g., UV-sensitive resin droplets jetting) is the manufacturing technology that offers the most control over the appearance of an object, as many colors and materials are available and can be combined in the same part. One drawback of this fabrication process is that the inks are partially opaque, implying that the surface's colors are blurred. Several works [Babaei et al. 2017; Rittig et al. 2021; Sumin et al. 2019] optimize the choice of inks and strive to reproduce color faithfully by simulating light propagation in the object and modifying its volume composition. Beyond colors, Ansari et al. [2020] reproduced input spectral images using a data-driven model trained on a large dataset of prints made of a selected set of inks. Piovarči et al. [2020] proposed a method to control the roughness (gloss property) of a given 2.5 D object by introducing new printing hardware that sprays viscous varnishes on the object's surface. They achieved high-quality results, but the method is limited to isotropic appearance.

Unfortunately, multi-material jetting costs an order of magnitude higher than material extrusion (e.g., FFF) or layer solidification (e.g., stereolithography or SLA). Song et al. [2019] targeted an affordable fabrication technology and proposed a color mixing method using FFF. This enables color printing at a low cost, using commonly available hardware and materials.

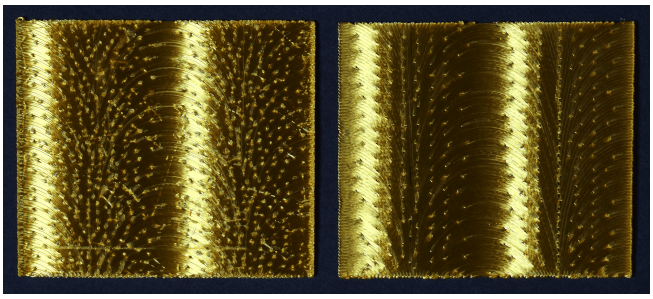


Fig. 4. Trajectories' directions angles increasing at a constant rate along the x directions. Contrary to continuous deposition (right, using our method), travels can leave unwanted but visible traces (left, using the method of Tricard et al. [2021]). This is worsened on the left by a much larger number of singularities.

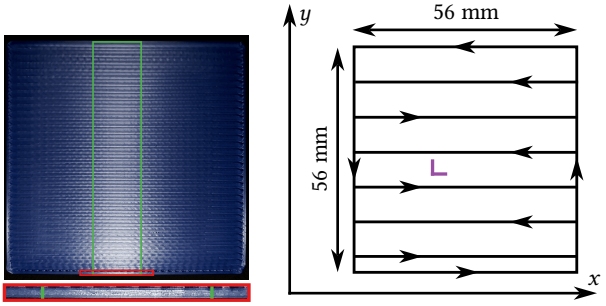


Fig. 5. Specifications of the printed samples for roughness measurements. *Left*: picture of blue PLA (0.8 mm bead width) taken with a flash, showing a typically elongated highlight (green rectangle) of glossy anisotropic material (Figure 3). In the red inset, we observe that a single filament exhibits an anisotropic appearance. *Right*: a schematic representation of our samples. The width of the printing trajectory black lines matches the width of a single bead in the photography. The printing directions are shown with arrows. We define the x direction as the zig-zag direction. The purple L represents the x and y evaluation lengths (five times the sampling length).

Regarding anisotropic light scattering, several works control the shape of the BSDF by generating and manufacturing micro-surfaces (microfacets) with user-specified distributions of normals (or heights). These works used different fabrication processes: micro-milling [Weyrich et al. 2009], material jetting [Lan et al. 2013; Rouiller et al. 2013], stereolithography [Luongo et al. 2020], and photolithography [Levin et al. 2013]. They demonstrate precise control of appearance, but one major limitation of these methods is the need for many microfacets in a microscopic spatial element (e.g., one square millimeter) to obtain spatially stable surface roughness statistics.

We observe that we can instead make the most of a specific property of FFF: the extruded material readily presents an anisotropic roughness. We can thus avoid the micro-surface generation step and the high spatial resolution constraint. Recent works [Bedel et al. 2022; Tricard et al. 2021] have also observed that property without precisely quantifying it. In our work, we experimentally verify this observation with roughness measurements in Section 3.

Tricard et al. [2021] demonstrated the possibility of producing a controlled anisotropic appearance. However, the proposed optimizer is heuristic and does not support alignment and sparse directional constraints. It also creates a high number of singularities – forks in trajectories – damaging the print quality, as shown in the comparison Figure 4.

2.2 Dense infills with orientable trajectories

Densely filling an area with material beads is a standard operation in additive manufacturing [Livesu et al. 2017]. However, in our setting, we need to tightly control the orientation of the deposition paths to influence the anisotropy direction. Dense and orientable infills can be obtained by optimizing a continuous scalar field, where trajectories are extracted as iso-values [Fang et al. 2020; Steuben et al. 2016; Tricard et al. 2021]. A uniform subdivision can also be used to recover a uniform or controlled spacing [Elber and Cohen 1990; Etienne and Lefebvre 2020; Ezair et al. 2018]. While orientation is a key ingredient to our method, the final surface quality should also be free of printing defects. However, defects arise when deposition paths are not continuous, especially at trajectories' endpoints (see Figure 4). These artifacts can be alleviated by filling a closed space with a single, continuous, and cyclic circuit, avoiding spurious deposits and unwanted traces.

Obtaining controllable, dense, and oriented *cyclic* infills is a challenging endeavor considered by few works. Zhao et al. [2016] generated connected Fermat spirals, but the cycle's orientations can only be parallel to the contour. The method of Pedersen and Singh [2006] offers controls similar to our goals by growing a circle in the shape's interior. The method of [Bedel et al. 2022] (Hamiltonian Cycle) is the closest to ours in terms of control and objective. However, the Hamiltonian Cycle approach is still far from ideal as it does not scale well (Table 4) and does not deliver satisfactory quality for our target application (Figure 12). In particular, the method cannot create long lines without turns. Our algorithm gains up to three orders of magnitude in execution time and better follows the input orientation field. From a technical point of view, the Hamiltonian Cycle approach and ours are fundamentally different: Bedel et al. [2022] start by covering the input space with a graph to obtain an initial Hamiltonian cycle, which is subsequently iteratively refined. In contrast, our approach works with a sum of sinusoidal functions representing the paths. Optimizing its parameters — directions and phases — leads to improvements in time and quality.

3 ANISOTROPIC APPEARANCE CHARACTERIZATION

To characterize the roughness anisotropy of an extruded filament, we selected five filament samples: clear PETG, red PETG, gold PLA, blue PLA, and purple PLA. We observed that filaments with minor surface roughness maximize the anisotropy, as the commercially available PETGs and glossy PLAs selected here. We printed two square-shaped samples (56^2 mm^2) for each filament sample with commercially available, brass coated, 0.4 and 0.8 mm nozzles. The bead width was set to nozzle extrusion hole width, and the layer height was set to bead width over two. Each layer was filled with a zig-zag (see Figure 5). We printed on a heating bed with a smooth PEI (Polyetherimide) sheet surface.

Table 1. Roughness (RMS slope) measurements characterize the anisotropic appearance. Measurements on the nozzle's and bed's sides are on each cell's top and bottom, *resp.* Each triplet $(\sigma_x, \sigma_y, \sigma_y/\sigma_x)$ has the following layout: the roughnesses along the x and y directions, then their ratio.

	Nozzle 0.4	Nozzle 0.8
Clear PETG	(0.88, 11.63, **13.2**)	(0.25, 14.20, **56.8**)
	(2.36, 11.78, **4.9**)	(1.90, 18.26, **9.6**)
Red PETG	(6.20, 12.98, **2.0**)	(1.36, 4.60, **3.3**)
	(2.34, 18.82, **8.0**)	(2.23, 21.29, **9.5**)
Gold PLA	(8.93, 15.60, **1.7**)	(6.36, 16.66, **2.6**)
	(2.43, 9.72, **4.0**)	(2.19, 19.34, **8.8**)
Blue PLA	(2.36, 17.32, **7.3**)	(2.31, 17.69, **7.6**)
	(2.56, 13.54, **5.2**)	(2.43, 20.90, **8.6**)
Grey PLA	(4.46, 15.81, **3.5**)	(4.38, 11.92, **2.7**)
	(2.57, 12.46, **4.8**)	(2.38, 19.16, **8.0**)

For PETG filaments, we printed at 235 °C, and for the PLA filaments, we printed at 210 °C. With the 0.8 mm nozzle, the speed was 20 mm/s for the first layer and then 30 mm/s. With the 0.4 mm nozzle, the speed was doubled.

We use a mechanical profilometer with a three microns radius tip to perform the roughness measurements (DektakXT from Bruker). This tool measures the heights along a given direction, given a primary height profile. To characterize the anisotropy, we acquired a profile along the tangent and bitangent of a printed trajectory (x and y directions in Figure 5). The primary height profile is filtered with a 25 μm cut-off length, given a roughness profile. Dong et al. [2016] show that high-frequency filtering of details is essential for characterizing appearance. From the roughness profile, the measurement tool computes the square average of the local slope (Rdq parameter [Nyhuis 2016], also called the root mean square (RMS) slope, and defined by the international standard ISO 4287) within a sampling length of 0.8 mm. We used the default five sampling lengths for evaluation (purple lines in Figure 5). According to ISO 4287, the roughness parameter is computed on sampling lengths — here, five disjoint successive segments — and then averaged automatically by the profilometer.

For the ten samples, we measured the Rdq roughnesses (σ_x, σ_y) along the x and y direction, respectively. We chose this particular roughness parameter among others because microfacet BSDFs [Heitz 2014] use it.

We performed each measure on both sides of the printed samples (nozzle and bedside) at a random position around the part's center. Roughness measurements are reported in Table 1. The ratios between σ_y and σ_x reveal that the FFF printer's nozzles extract material with anisotropic roughnesses (lowest roughness along the printing direction). These high ratios explain the typically elongated highlight visible in Figure 5 and predicted by physically based light scattering models (Figure 3).

In summary, we have experimentally verified that the direction of deposition paths strongly correlates with surface roughness and consequently affects the anisotropic appearance. Considering this observation, in the next section, we present how to 3D print an anisotropic appearance with an arbitrary direction of anisotropy by controlling the orientation of a dense infill cycle.

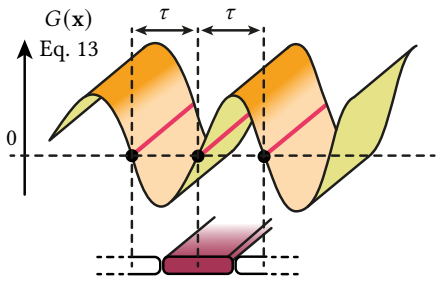


Fig. 6. We implicitly represent trajectories by an oscillating field whose target period is the input trajectory interspace parameter. The zero isocurves are the trajectories, and the field gradients are orthogonal to them.

4 OBJECTIVES, INPUTS, AND KEY IDEAS

This section gives our approach's objectives, inputs, and essential ideas. The main objective is to fabricate an anisotropic appearance with arbitrary anisotropy directions by capitalizing on the observation that extruded filament already exhibits an anisotropic appearance (see Section 3). To this end, the orientation of the deposition paths must follow the input anisotropy directions map as closely as possible. In addition, to avoid surface defects caused by nozzle travels, the deposition must ideally follow a continuous path.

The first key idea is to use an implicit representation of the trajectories, defined as the zero-level set of an oscillating scalar field with a target period of two times the trajectory interspace τ (see Figure 6). This has a twofold advantage. Firstly, the algorithm optimizes using an implicit representation of the paths, which provides intrinsic topology changes compared to an explicit representation (e.g., polygonal curves). Secondly, extracting the zero isolines with the marching squares algorithm (2D marching cubes [Lorensen and Cline 1987; Wyvill et al. 1986]) produces a finite set of cycles without self-intersections, providing proper handling of the domain boundary. This property is crucial because it allows us to exploit the second key idea: a finite set of disjoint cycles can be transformed into a single cycle without self-intersections [Kahng and Reda 2004].

As shown in Figure 5, going to the right or the left when extruding material gives the same anisotropic appearance. Hence, the nozzle can go in one direction or the other along the cycle. We use a line field to represent this property and to define user orientation constraints. A line field, also called a 2-directional field, specifies two directions $(\mathbf{d}, -\mathbf{d})$ for each domain point. This field type is rotationally-symmetric and is named a RoSy field in the literature [Ray et al. 2008]. Detailed information on directional fields is available in Vaxman et al. [2016].

It can also be helpful to only define vectorial shapes with boundaries that smoothly diffuse their normal inside their interior, similar to diffusion curves for smooth color gradients [Orzan et al. 2013]. For this reason, our input specifies explicit lines and three additional specific modes. In Figures 1 and 8, we use the red, blue, and yellow colors to represent them. They indicate areas where the trajectories' orientations must be the smoothest (yellow), the smoothest and

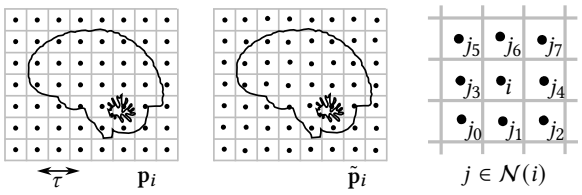


Fig. 7. The domain of the 2D shape is uniformly sampled with points $\mathbf{p}$ with spacing $\tau/2$ (left). Then a small perturbation is applied to them, giving perturbed points $\tilde{\mathbf{p}}$ (middle). The neighborhood $\mathcal{N}(i)$ of a vertex i is the set of eight vertices belonging to a cell next to the one of i (right).

parallel to the boundary (red), or the smoothest and orthogonal to the boundary (blue). We call the *direction mode map* the three direction modes combined with the anisotropy line map giving the line orientation in the range $(-\pi/2, \pi/2]$. This map is visible in Figures 1 and 8, which is one of our program inputs.

Shapes are transformed into a signed distance field (SDF) as it is a convenient representation for additive manufacturing: It naturally defines sides: interior with negative values, boundary with zero values, and exterior with positive values. Note that an explicit geometry can be simply and efficiently transformed into a SDF [Baerentzen and Aanaes 2005]. In the next section, we describe the cycle generation algorithm.

5 CYCLE GENERATION ALGORITHM

This section details each step of the cycle generation. See Figure 8 and Algorithm 1 to see an overview.

ALGORITHM 1: Cycle generation (SDF, directionModeMap, τ)

```
P, P̃ ← GeneratePoints(Bounds(SDF), τ)
L ← PointLineInit(P̃, SDF, directionModeMap, τ)
L ← SmoothLines(P̃, L)                          // Sec. 5.1
Φ ← PointPhaseInit(P̃, SDF)                     // Sec. 5.2
S ← (P̃, L, Φ)                                  // Create sine field
S ← AlignSines(S)                              // Sec. 5.2
C ← MarchingSquare(G(P − τ/4, S))              // Sec. 5.4
C_s ← StitchCycles(C)                          // Sec. 5.5, Alg. 2
```

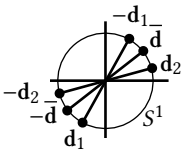
Domain sampling with points. A uniform grid partitions the domain with a cell width equal to $\tau/2$. Each cell contains a vertex $i \in \mathcal{V}$, where $|\mathcal{V}|$ denotes the total number of vertices. Each vertex i has an associated point $\mathbf{p}_i \in \mathbb{R}^2$ corresponding to the center of the cell and a line $\mathbf{d}_i \in S^1$ which is a representative normalized direction of a line.

Each point sample $\mathbf{p}_i$ is perturbed with a random translation of a small fraction of the target spacing τ, given perturbed points $\tilde{\mathbf{p}}_i$ (Figure 7). Even if the perturbation is not strictly necessary for the method to work, we observed that it helps for optimization convergence by breaking ties due to perfect symmetries (i.e., bringing the perturbed points into a general position).

The matrices $P \coloneqq (\mathbf{p}_i)_{i\in\mathcal{V}}$, $\tilde{P} \coloneqq (\tilde{\mathbf{p}}_i)_{i\in\mathcal{V}}$, and $L \coloneqq (\mathbf{d}_i)_{i\in\mathcal{V}}$ of shape $|\mathcal{V}|\times 2$ represent the points, the perturbed points, and the line field, *resp.* Finally, we define the neighborhood $\mathcal{N}(i)$ of a vertex i as the eight vertices j whose cell surrounds the i's cell.

Points' line initialization. A line d_i is initialized with a default value for each domain sampling point $\tilde{\mathbf{p}}_i$. For points inside the boundary area $\{\mathbf{x} \mid \mathrm{SDF}(\mathbf{x}) \in [-\tau; -\tau/2]\}$, the lines d_i are set to the normalized gradient of the SDF and are constrained, i.e., the optimizer will not modify them during the smoothing. In Figure 8.1, the default line orientation is set to $(1, 0)^T$.

5.1 Compute the smoothest line field



Suppose the user specifies zones where the smoothest lines must be computed, e.g., parallel or orthogonal to the border. In that case, the boundary's lines are diffused inside the shape by calculating the smoothest line field with respect to the following metric. The smoothness energy E_l of a line $\mathbf{d}$ with m other lines represented by the $m \times 2$ matrix $D \coloneqq (\mathbf{d}_j)_{1\le j\le m}$ is defined as [Paris et al. 2008]

$$E_l(\mathbf{d}, D) \coloneqq -\mathbf{d}^T D^T D \mathbf{d}. \tag{1}$$

The term $D^T D$ is the covariance matrix of the data representing the $2m$ endpoints $\mathbf{d}$ and $-\mathbf{d}$. The directions' symmetry implies that the mean is zero. The term $\mathbf{d}^T D^T D \mathbf{d}$ is the variance of the projected data vectors on $\mathbf{d}$. The line $\bar{\mathbf{d}}$ that aligns the most with the m other lines D is the direction of maximum variance, which is the eigenvector of the covariance matrix $D^T D$ with the highest eigenvalue

$$\bar{\mathbf{d}} \coloneqq \operatorname*{argmin}_{\mathbf{d}\in S^1} E_l(\mathbf{d}, D). \tag{2}$$

We define $\bar{\mathbf{d}}$ as the average of m lines D.

The smoothness of the line field L is the sum of each local smoothness energy E_l of the line $\mathbf{d}_i$ with its weighted neighborhood

$$E_L(L) \coloneqq \sum_{i\in\mathcal{V}} E_l\left(\mathbf{d}_i, (w_{ij}\mathbf{d}_j)_{j\in\mathcal{N}(i)}\right). \tag{3}$$

The neighboring lines $\mathbf{d}_j$ of a vertex i can be weighted by multiplying each adjacent direction $\mathbf{d}_j$ by a scalar w_{ij}. We chose to spatially weight each neighbor j with a 2D Gaussian evaluated at $\tilde{\mathbf{p}}_j$ with a mean set to $\tilde{\mathbf{p}}_i$ and a standard deviation set to $\tau/6$.

The line of a vertex i is set to the average of its neighbors to decrease the line field energy E_L:

$$\bar{\mathbf{d}}_i \coloneqq \operatorname*{argmin}_{\mathbf{d}_i\in S^1} E_l\left(\mathbf{d}_i, (w_{ij}\mathbf{d}_j)_{j\in\mathcal{N}(i)}\right). \tag{4}$$

Only the eight neighbors of the vertex i are implied in the covariance matrix computation.

This operation is done for all the lines to obtain an updated line field that further decreases this energy.

$$L \leftarrow \bar{L}, \quad \text{where} \quad \bar{L} \coloneqq (\bar{\mathbf{d}}_i)_{i\in\mathcal{V}}. \tag{5}$$

We iteratively decrease the energy by repeating the update operation several times, except for the constrained lines that are not updated. This approach is similar to the diffusion of a quantity inside an area: a process that has a slow convergence and tends to get stuck in local minima. As demonstrated in the context of quad and

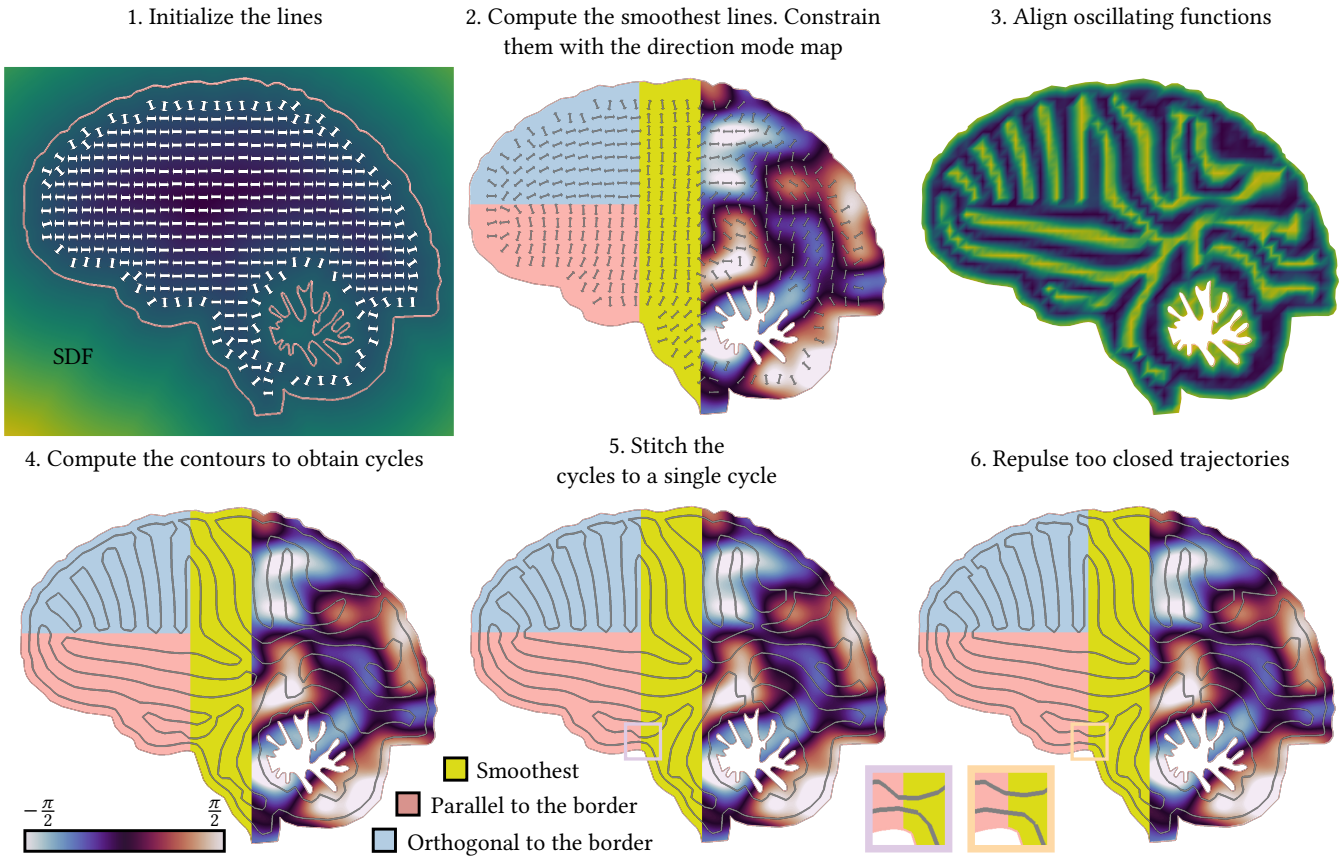


Fig. 8. Overview of the different steps of our cycle generation algorithm. A line field is initialized and smoothed (1, 2). It directs an aligned sine field (3) whose contours (4) are stitched to form a cycle (5). Trajectories that are too close are repulsed (6).

hexahedral meshing [Gao et al. 2017; Jakob et al. 2015], a multiresolution hierarchy can alleviate these drawbacks, so we use one to accelerate the computations (Section 5.3).

The smoothed lines are kept in the red areas. They are rotated by 90 degrees in the blue zones, given parallel and orthogonal trajectories to the boundary at the end (see Figure 8).

The lines inside the color gradient areas in the line mode map are set with the prescribed directions encoded as a polar angle in $(-\pi/2, \pi/2]$. A 90 degrees rotation is applied to them because each line indicates the orientation given the highest variation in the oscillating scalar field representing the trajectories.

All the lines are constrained at this point, except in the yellow areas, as these zones specify unconstrained lines. In Figure 8.2, we kept the smoothest lines as the default orientation in the yellow space.

5.2 Align periodic functions

Our method is based on an implicit representation of the trajectories, defined as the zero values of an oscillating scalar field (Figure 8.3). To obtain this scalar field, we associate a phase $\varphi_i \in (-\pi, \pi]$ to each vertex i, which allows us to define a local oscillating function around each point $\tilde{\mathbf{p}}_i$. The field of phases is denoted $\Phi \coloneqq (\varphi_i)_{i \in \mathcal{V}}$,

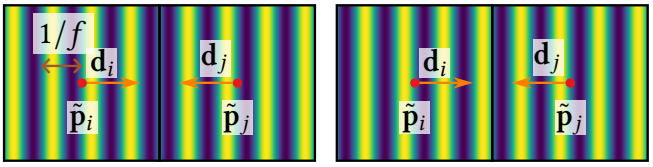


Fig. 9. The sine wave of the vertex i is not aligned with its neighbor j (left). By shifting the phase of i, the two sine waves can be aligned (right).

and the sine field S is represented by the perturbed points $\tilde{P}$, the line field L, and the phases Φ.

We must align the local oscillating functions locally to have trajectories at regular intervals. In different areas of computer graphics, oscillating function alignment has been used for surface parameterization [Ray et al. 2006] or synthesizing stripe patterns on triangulated surfaces [Knöppel et al. 2015].

The sine wave s_i is a local periodic function around the point $\tilde{\mathbf{p}}_i$

$$s_i(\mathbf{x}) \coloneqq \sin(2\pi f(\mathbf{x} - \tilde{\mathbf{p}}_i) \cdot \mathbf{d}_i + \varphi_i), \tag{6}$$

where f is the signal's frequency, fixed to $1/(2\tau)$. This function is plotted in Figure 9 with a higher frequency for illustration purposes.

We define the alignment energy $E_s(i,j)$ of the sine wave of vertex i with a vertex j as

$$\int_0^{\frac{1}{f}} \left(\sin\left(\operatorname{sgn}(\mathbf{d}_i \cdot \mathbf{d}_j) 2\pi f \left(x - p_i^{\perp j}\right) + \varphi_i\right) - \sin\left(2\pi f x + \varphi_j\right)\right)^2 \mathrm{d}x, \tag{7}$$

where $\operatorname{sgn}(x)$ is the sign function (1 if $x > 0$, -1 otherwise) and where $p_i^{\perp j} := (\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_j) \cdot \mathbf{d}_j$ is the scalar projection of the vector $(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_j)$ onto the direction $\mathbf{d}_j$. When the two sines are in opposite directions ($\mathbf{d}_i \cdot \mathbf{d}_j \leq 0$ and $\operatorname{sgn} = -1$), i's phase is rolling in j's inverse direction (Figure 9).

The energy term E_s measures the difference between the projected 1D sine of i onto j's line and the sine of j and is minimized when the two sines are equaled, i.e., when the phase of i is

$$\overline{\varphi}_{ij} := \underset{\varphi_i \in (-\pi, \pi]}{\operatorname{argmin}} \; E_s(i,j) = \begin{cases} 2\pi f p_i^{\perp j} + \varphi_j, & \text{if } \mathbf{d}_i \cdot \mathbf{d}_j > 0 \\ -(2\pi f p_i^{\perp j} + \varphi_j) + \pi, & \text{otherwise.} \end{cases} \tag{8}$$

With a measure of how a pair of sine waves are aligned, we can now define the alignment energy E_S of the sine field S. The energy E_S is the sum of each local alignment energy E_s of each sine i with its neighborhood

$$E_S(S) := \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}(i)} w_{ij} |\mathbf{d}_i \cdot \mathbf{d}_j| E_s(i,j). \tag{9}$$

We weigh each neighbor j with the 2D Gaussian used for the line optimization. In addition, each neighbor is weighted by the absolute value of the dot product $|\mathbf{d}_i \cdot \mathbf{d}_j|$ to encourage alignment between vertices having similar orientations. Sine waves with orthogonal directions cannot be aligned.

The same strategy used for the lines is employed here to decrease the sine field energy E_S and align the oscillating functions. We locally minimize the energy of vertex i with

$$\overline{\varphi}_i := \operatorname{Arg}\left(\sum_{j \in \mathcal{N}(i)} w_{ij} |\mathbf{d}_i \cdot \mathbf{d}_j| \exp(\imath\, \overline{\varphi}_{ij})\right), \tag{10}$$

and we update the phase field by reducing the energy of all the vertices

$$\Phi \leftarrow \overline{\Phi}, \quad \text{where} \quad \overline{\Phi} := (\overline{\varphi}_i)_{i \in \mathcal{V}}. \tag{11}$$

The phases inside the boundary area are initialized with the value $\pi(\mathrm{SDF}(\mathbf{x})/\tau + 1/2)$ to have trajectories that follow the shape's boundary, with an offset of half a nozzle inwards. These phases are constrained before the alignment process, i.e., they are not updated.

As it is a diffusion process, the convergence is slow, and we accelerate it again using a multiresolution approach.

5.3 Multiresolution grid

Multiresolution grids are standard in numerous domains for solving linear systems, image processing, fluid simulations, etc. In our situation, lines or phases must diffuse inside the shape (Sections 5.1 and 5.2), and a simple hierarchy quickly propagates information in the domain.

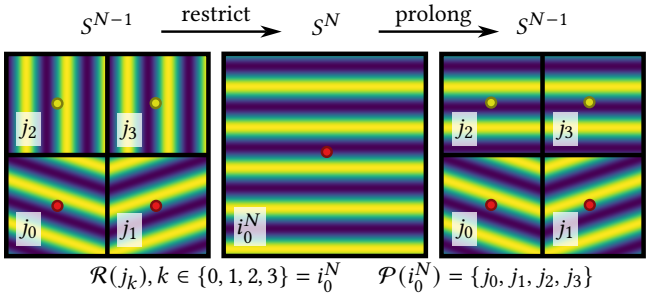


Fig. 10. Illustration of the restriction and prolonged operations used to accelerate the sines' alignment. A sine field S^{N-1} with two constrained (red) and unconstrained (yellow) vertices is restricted and then prolonged.

Restrict. The sine field $S^0 := \left(\tilde{P}^0, L^0, \Phi^0\right)$ is successively restricted into a series of sine fields $S^n := \left((\tilde{\mathbf{p}}_{i^n}), (\mathbf{d}_{i^n}), (\varphi_{i^n})\right), i^n \in \mathcal{V}^n$ with $0 \leq n \leq N$ until the coarsest sine field, S^N. Each level is defined recursively, starting from level zero. A sine field at level n has vertices/cells $i^n \in \mathcal{V}^n$: Its number of vertices/cells along the x and y axes is the power of two $B/2^n$, where B is also a power of two, giving the side's number of cells at level zero. A vertex i^{n+1} at level $n+1$ is prolonged to four adjacent vertices $\mathcal{P}\left(i^{n+1}\right)$ at level n. Inversely, a prolonged vertex $j \in \mathcal{P}\left(i^{n+1}\right)$ is restricted to its original vertex i^{n+1}, i.e., $i^{n+1} = \mathcal{R}(j)$. A point $\tilde{\mathbf{p}}_{i^{n+1}}$ at level $n+1$ is the barycenter of its four corresponding points at level n, i.e., $\tilde{\mathbf{p}}_{i^{n+1}} := 1/4 \sum_{j \in \mathcal{P}(i^{n+1})} \tilde{\mathbf{p}}_j$. A line $\mathbf{d}_{i^{n+1}}$ at level $n+1$ is the average of its four corresponding lines $(\mathbf{d}_j)_{j \in \mathcal{P}(i^{n+1})}$ at level n, i.e., $\mathbf{d}_{i^{n+1}} := \operatorname{argmin}_{\mathbf{d} \in S^1} E_l\left(\mathbf{d}, (\mathbf{d}_j)_{j \in \mathcal{P}(i^{n+1})}\right)$. A phase $\varphi_{i^{n+1}}$ at level $n+1$ is the average of its four corresponding phases at level n

$$\varphi_{i^{n+1}} := \operatorname{Arg}\left(\sum_{j \in \mathcal{P}(i^{n+1})} \exp\left(\imath\, \overline{\varphi}_{i^{n+1} j}\right)\right). \tag{12}$$

Only constrained lines and phases are averaged if any prolonged vertices $\mathcal{P}\left(i^{n+1}\right)$ have a constrained line or phase. A restricted vertex has a constrained quantity (line or phase) if at least one of its prolonged vertices has a constrained quantity (see Figure 10).

Prolong. Once the multiresolution sine field is built from the finest to the coarsest sine field S^N, the quantities return to the finest resolution. For each level n, except the coarsest one, we smooth the directions with Equation 5 and align the sines with Equation 11 several times. Then, the quantities are prolonged to level $n-1$, i.e., $\mathbf{d}_{i^{n-1}} := \mathbf{d}_{\mathcal{R}(i^{n-1})}$ and $\varphi_{i^{n-1}} := \overline{\varphi}_{i^{n-1}\mathcal{R}(i^{n-1})}$, except when constrained. Figure 10 illustrates the restriction and prolonged operations.

5.4 Sine field evaluation and contouring

Sine field evaluation. After the minimization process, the sine field S is evaluated. We use the following function inspired by the noise synthesizing domain [Lagae et al. 2009; Tricard et al. 2019] to evaluate the oscillating sine field S at an arbitrary position

$$G(\mathbf{x}, S) := \sum_{i \in \mathcal{N}(i_x) \cup i_x} w_i(\mathbf{x}) s_i(\mathbf{x}) / \sum_{i \in \mathcal{N}(i_x) \cup i_x} w_i(\mathbf{x}), \tag{13}$$

where w_i is a 2D Gaussian with mean $\tilde{\mathbf{p}}_i$ and standard deviation $\tau/6$. The term i_x is the vertex of the grid's cell of the evaluation point $\mathbf{x}$. The Gaussian has zero value after three standard deviations, allowing us only to evaluate the sine waves in the $\mathbf{x}$'s neighborhood. A plot of this function is visible in Figure 8.

Extraction of cycles. The marching square algorithm uses the grid defining the points (Figure 7) with a shift of $\tau/4$. The sine field S is evaluated for each grid vertex with the previous equation G. The scalar field's values are set to $2\text{SDF}(\mathbf{x})/\tau + 1$ in the zone defined as $\{\mathbf{x} \mid \text{SDF}(\mathbf{x}) \in [-\tau/2; 0]\}$ and one *at the shape's exterior*. These particular settings are essential to output a set of cycles and prevent trajectory extraction in the shape's exterior minus a margin of $\tau/2$.

We represent the collection of cycles as a graph $C = (\mathcal{V}_C, \mathcal{E}_C)$ with vertex points $\gamma : \mathcal{V}_C \to \mathbb{R}^2$. Its number of cycles, vertices, and edges are denoted $|C|$, $|\mathcal{V}_C|$, and $|\mathcal{E}_C|$, *resp.* Each edge $I \in \mathcal{E}_C$ connects two vertices, i_1 and i_2. We represent the k^{th} cycle as the subgraph $C_k = (\mathcal{V}_{C_k} \subset \mathcal{V}_C, \mathcal{E}_{C_k} \subset \mathcal{E}_C), k \in [1, |C|]$.

5.5 Stitch the cycles

The cycles C obtained with the contouring algorithm are stitched together to form a single cycle C_s. The algorithm is recursive: two cycles are matched and then stitched, and this operation is done $|C| - 1$ times, where $|C|$ is the number of cycles.

Matching two cycles is defined as the finding operation of a pair of edges belonging to each other, giving a stitch without intersection with any of the cycles' edges. The patching energy of an edge $I \in \mathcal{E}_C$ with $J \in \mathcal{E}_C$ is

$$E_p(I,J) \coloneqq \min(||\gamma_{i_1} - \gamma_{j_2}|| + ||\gamma_{i_2} - \gamma_{j_1}||, ||\gamma_{i_1} - \gamma_{j_1}|| + ||\gamma_{i_2} - \gamma_{j_2}||) - ||\gamma_{i_1} - \gamma_{i_2}|| - ||\gamma_{j_1} - \gamma_{j_2}|| \quad (14)$$

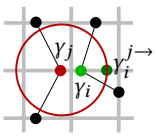
and gives a non-intersecting patch (I, J) for the least cost, as noted by Kahng and Redha [2004]. We use this energy in Algorithm 2 to recursively match and then stitch two edges while the number of cycles is not one. We first find the cycle C_k with the smallest number of edges. Then, for each of its edges ($I \in \mathcal{E}_{C_k}$), we compute the patching energy with the neighboring edges not belonging to $\mathcal{E}_{C_k}$. We note $\mathcal{N}(C_k)$ the neighboring edges of cycle C_k. A result of the algorithm is visible in Figure 8.5, where the input was the cycles in Figure 8.4.

ALGORITHM 2: Stitch the cycles C into a single cycle C_s.

```
while |C| > 1 do
    k ← argmin_{k∈[1,|C|]} |E_{C_k}|
    Ī, J̄_I = argmin_{I∈E_{C_k} and J∈N(C_k)} E_p(I, J)
    C ← StitchEdges(Ī, J̄_I)
end
C_s ← C
```

5.6 Repulse trajectories and compute variable widths

Repulse trajectories. In a few areas, the interspace between extracted contours can be small ($< \tau/2$), as shown in the purple inset in Figure 8. To avoid this issue, we repulse the cycle's points



within a $\tau/2$ distance. Each point of the curves is on a unique edge of the domain grid after the contouring algorithm and the stitching modifies the connectivity, not the positions. We leverage the grid configuration and particular vertex positioning to perform efficient neighboring requests. We also constrain the points to their grid edge to avoid self-intersection. To repulse the vertex position γ_i by point γ_j, we compute the intersection $\gamma_i^{j\rightarrow}$ of a $\tau/2$ radius circle centered in γ_j with i's grid edge. This computation is only done with neighboring vertices $j \in \mathcal{N}^{\rightarrow}(\omega_i)$, where $\mathcal{N}^{\rightarrow}(\omega_i)$ is the set of vertices respecting the condition $||\gamma_j - \gamma_i|| < \tau/2$ and where $|\mathcal{N}^{\rightarrow}(\omega_i)|$ is their number. To update the point of vertex i we use

$$\gamma_i \leftarrow \frac{1}{2}\left(\gamma_i + \frac{1}{|\mathcal{N}^{\rightarrow}(\omega_i)|} \sum_{j \in \mathcal{N}^{\rightarrow}(\omega_i)} \gamma_i^{j\rightarrow}\right). \quad (15)$$

A repulsion iteration is the independent update of all the vertices.

Compute the trajectory's variable width. The trajectory obtained has a variable distance to neighboring curves, so a variable width is computed to minimize overlaps and maximize space covering. For each cycle's vertex i, we define its width $\overline{\omega}_i$ as two times the smallest circle radius tangent to γ_i and passing through a cycle point γ_c as

$$\overline{\omega}_i \coloneqq \underset{\gamma_c \in \mathcal{N}(\gamma_i)}{\text{argmin}}\ \omega_i(\gamma_c), \quad \text{where} \quad \omega_i(\gamma_c) \coloneqq \frac{||\gamma_i - \gamma_c||^2}{|\det A|}. \quad (16)$$

The term A is the 2×2 matrix $\begin{bmatrix} T_i & \gamma_i - \gamma_c \end{bmatrix}^T$, where the first row is the tangent at point γ_i. The term $\mathcal{N}(\gamma_i)$ is the set of points on the edges (we use three point samples per cycle's segment) within a 2τ distance from γ_i. This formulation naturally ignores points close along the curve [Yu et al. 2021]. In our implementation, we compute ω_i with points sampling the neighboring cycle segments and keep the minimum width $\overline{\omega}_i$. This operation is done for all the cycle's vertices, and then the values are clamped in the printer's achievable bead widths range ($[3/4\tau, 2\tau]$ for our printer). Distributions of unclamped widths $\overline{\omega}_i$ computed from cycles generated with our method and the method of Bedel et al. [2022] are visible in Figure 14.

5.7 Implementation

We implemented the generation cycle algorithm with the JAX Python library [Bradbury et al. 2018].

Unified line and phase optimization. We chose to unify the line and the phase update (Equations 4 and 10) into a single optimization iteration. As a result, the SmoothLines() and AlignSines() functions in Algorithm 1 are the same in our implementation. The phases returned by SmoothLines() are ignored, and all the lines in AlignSines() are constrained except in the yellow areas (areas indicating smooth line field objective). The number of optimization iterations per level in the multiresolution grid is set to 32 in our implementation.

Point perturbation. Each vertex i is associated with a unique perturbed point $\tilde{\mathbf{p}}_i \coloneqq \mathbf{p}_i + \mathbf{r}_i$ where $\mathbf{r}_i$ is a random translation of width $\tau/10$, i.e., the i^{th} realization of the random variable following a bivariate uniform distribution $\mathcal{U}_2\left((-\tau/10, \tau/10)^2\right)$. The value $\tau/10$ is not a critical value; it is an epsilon chosen as a small fraction of the target spacing τ.

Table 2. Some properties of the results. From left to right: the size of the bounding box of the object, the number of cells partitioning the domain, the next power of two cell count, and the number of cycles stitched. The cell count is square because it is a requirement of our optimizer implementation.

	Size (mm)	#cells	#cells^2	#cycles
LOGO	200×90.5	1004×457	$1{,}024^2$	548
WAVE	170×114.8	854×579	$1{,}024^2$	495
BRAIN	63.75×51.40	328×266	512^2	121
PEOPLE	98.45×80.37	501×411	512^2	283
VARY	42.54×170	216×853	$1{,}024^2$	260

Parallelization. The update operations of two different lines d_i and d_j with Equation 4, or two different phases φ_i and φ_j with Equation 10, are independent. Consequently, we parallelize all line and phase updates (Equations 5 and 11) for each iteration when minimizing energies E_L and E_S. The construction of the multiresolution grid, the creation of contours with the marching square algorithm, the repulsion of paths, and the calculation of path widths are also parallelized in our implementation. The parallelization is done with automatic vectorization using the `vmap` operator of JAX.

Complexity. The time complexity of each optimization iteration is linear ($O(|\mathcal{V}|)$ time, see the #cells^2 column in Table 2 for typical values) with respect to the number of sampling domain points/cells. The stitching part has quadratic complexity $O(|C||\mathcal{E}_C|)$, but this operation is only done once. The time complexity is linear ($O(|\mathcal{V}|)$ time, see the #cells column in Table 2 for typical values) for the other steps of the algorithm, i.e., the sine field evaluation, the marching square, the point repulsion, and the width computation. The number of cycle edges $|\mathcal{E}_C|$ is proportional to the shape's area. In the worst case, e.g., when having only small isolated components, the number of cycles $|C|$ can increase linearly with the shape's area. We report the number of cycles $|C|$ of our examples in Table 2. In general, the timings heavily depend on the inputs, making it hard to draw a simple trend with respect to the grid size.

Repulsion of trajectory points. We perform a fixed number of eight iterations for the repulsion of trajectory points, so the process cannot loop indefinitely. Vertices move half the distance to the target, and the points are constrained to their grid edge.

Flow management. Flow management during deposition, especially in sharp turns, can produce uneven flow. We use the 3D printer firmware Klipper to mitigate this issue, as it is designed to consider accelerations and material pressure.

6 RESULTS

In this section, we provide numerical and experimental results.

We 3D printed five different results with a Creality CR-10 S Pro. We used a brass-plated nozzle with a 0.4 mm extrusion hole width. The nozzle temperature and velocity were set to 225 °C and 50 mm/s, *resp.* Additional properties of each result can be found in Table 2. The supplementary video is particularly helpful in assessing our method's dynamic visual effect on the surfaces when the light is moving or when the object is rotating.

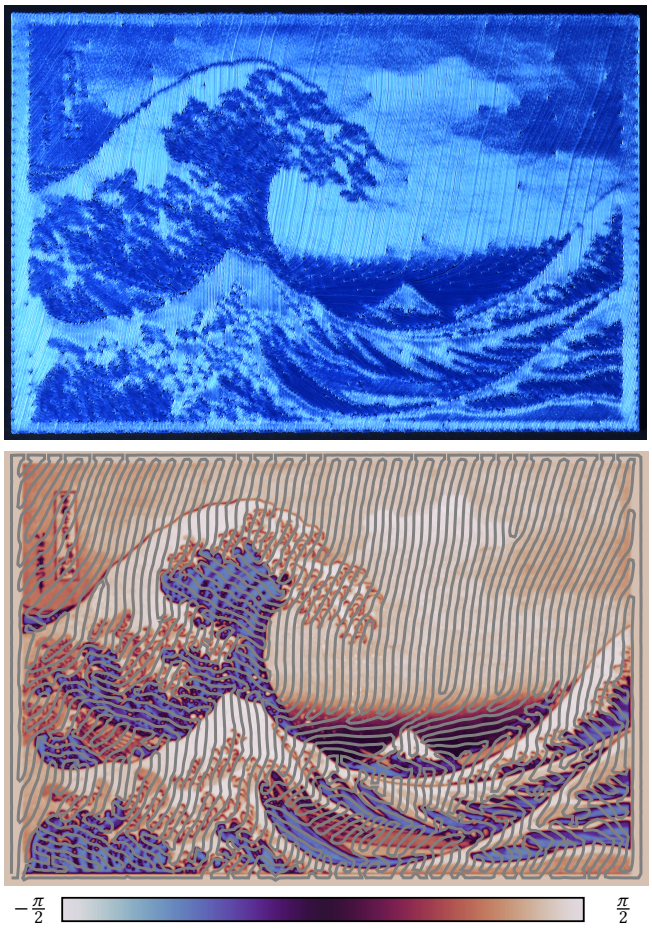


Fig. 11. The WAVE of Hokusai constrains the directions of the lines and gives a detailed anisotropic appearance once printed with our method. The small direction changes in the clouds have an impact on light reflections. Top: printed cycles with 0.4 mm interspace objective. Bottom: the lines' orientations map and cycles obtained with a 2.2 mm curve interspace objective.

Figure 1 (LOGO) was printed with a high gloss silver filament and depicts a decorated SIGGRAPH logo. Figure 11 (WAVE) was printed with silk blue PLA, and the line field was constrained with the artwork *The Great Wave off Kanagawa*. Note that the artistic details are finely developed and emerge naturally from the anisotropic appearance. Creating such an effect by an artist would be challenging using traditional mechanical brushing tools. Figure 12 shows the BRAIN, PEOPLE, and VARY results. BRAIN was printed with a high gloss silver filament and uses a line field to convey a dynamic appearance. PEOPLE was printed with a gold PLA and considers a highly discontinuous line field. We can appreciate the different reflectance of the patches corresponding to different values of orientation. VARY was printed with silk blue PLA and uses a noise field ranging from an isotropic to a fully anisotropic distribution of orientations.

Performance. We measure the execution times (Table 3) with an Nvidia GeForce RTX 2080 Ti GPU and an Intel Core i7-4770K CPU (3.50 GHz). All the algorithm runs on GPU, except for the stitching

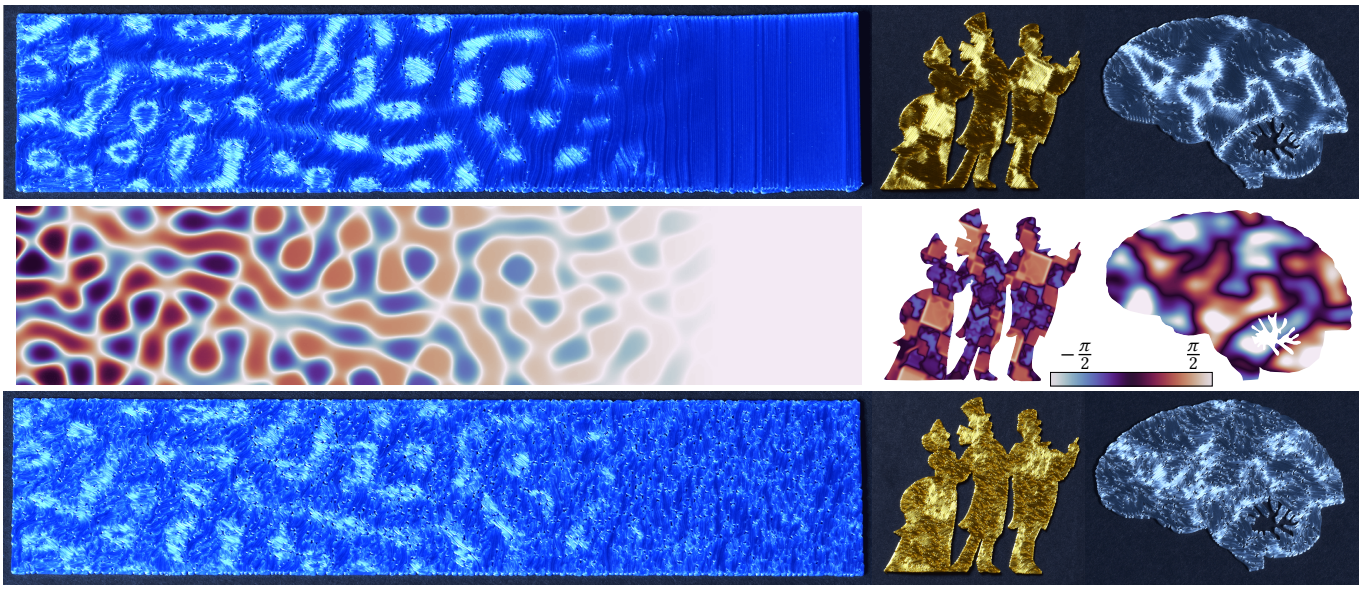


Fig. 12. The line fields of VARY, PEOPLE, and BRAIN (from left to right, middle line) orient the extrusion trajectories. Printed parts were obtained with our method (top) and Hamiltonian Cycle [Bedel et al. 2022] (bottom). Our trajectories better follow the prescribed line field, given smoother specular lobes.

Table 3. Detailed execution time for our dense cycle generation algorithm using the GPU, except for the stitching step that uses the CPU. From left to right: smoothing (only for LOGO) and sines alignment execution times, stitching execution times, and execution times for the other steps.

	Align	Stitch	Other steps	Total
LOGO	3.89 s (24%)	10.05 s (63%)	2.08 s (13%)	16.02 s
WAVE	2.03 s (15%)	10.48 s (75%)	1.49 s (10%)	14.00 s
BRAIN	0.70 s (21%)	1.02 s (31%)	1.54 s (48%)	3.26 s
PEOPLE	0.75 s (15%)	3.06 s (60%)	1.26 s (15%)	5.07 s
VARY	1.85 s (34%)	2.64 s (48%)	1.03 s (18%)	5.52 s

part, which uses the CPU. This step is the slowest because it has quadratic complexity. The trajectory alignment and stitching are as expensive as doing the other steps for small-size results like the brain. The line smoothing operation is only done for the LOGO result. Therefore its alignment execution time is almost twice that of the wave, which is a similar-sized result.

Stress tests. In Figure 13, we show results obtained with a spatially-varying frequency input: the frequency content of the line field and the boundary decreases from left to right. The direction mode map specifies four line modes: parallel to the border near the top and the bottom, horizontal and vertical lines in the middle, and the smoothest lines between these two areas. This configuration has many nearby points with orthogonal directions constraints. We stressed our algorithm by generating a cycle with a fixed nozzle width τ of 0.4 mm for smaller and smaller shape sizes. At some point, the sampling rate is insufficient to reconstruct all the details of the line field and shape's boundary, but our method still robustly generates an output. The algorithm's robustness stems from optimizing the parameters of the implicit representation, where the

Table 4. Absolute and relative execution times for both our method (multi-threaded GPU and mono-threaded CPU) and Hamiltonian Cycle (partially multi-threaded CPU).

	Ours (GPU)	Ours (CPU)	Hamiltonian (CPU)
BRAIN	3.26 s (×1)	12.67 s (×3.89)	275.44 s (×84.4)
PEOPLE	5.07 s (×1)	19.27 s (×3.80)	632.89 s (×124)
VARY	5.52 s (×1)	45.38 s (×8.22)	11017.7 s (×1995)

optimizer samples the constraints at the discrete evaluation points $\tilde{\mathbf{p}}$. Consequently, the resulting trajectories ignore the remaining features of a high-frequency control field. This is visible in Figures 11 and 13, which contain more details than the trajectory spacing τ could capture. Figure 8 also shows a case where the geometry along the boundary has a high frequency with respect to the target spacing. Nevertheless, the generated path remains well-behaved along these regions.

Comparison with [Bedel et al. 2022]. The BRAIN, PEOPLE and VARY results were generated with the method of Bedel et al. [2022] (Hamiltonian Cycle) by using their public code. We tried to generate the WAVE, but all our trials with different seeds failed as the algorithm did not terminate after 60 hours. In comparison, our method is robust: all the tests we performed always propose an oriented cycle infill. The LOGO was not generated with Hamiltonian Cycle as their method does not consider the line smoothing, as requested by the yellow area in this result. We performed an execution time comparison in Table 4. As the area of the shape grows, the execution time difference gets largely higher in our favor. This is due to the quadratic complexity of their combinatorial optimizer.

In Table 5, we give the coverage and overlap areas as a percentage of the shape's area. The alignment energy of the trajectories with

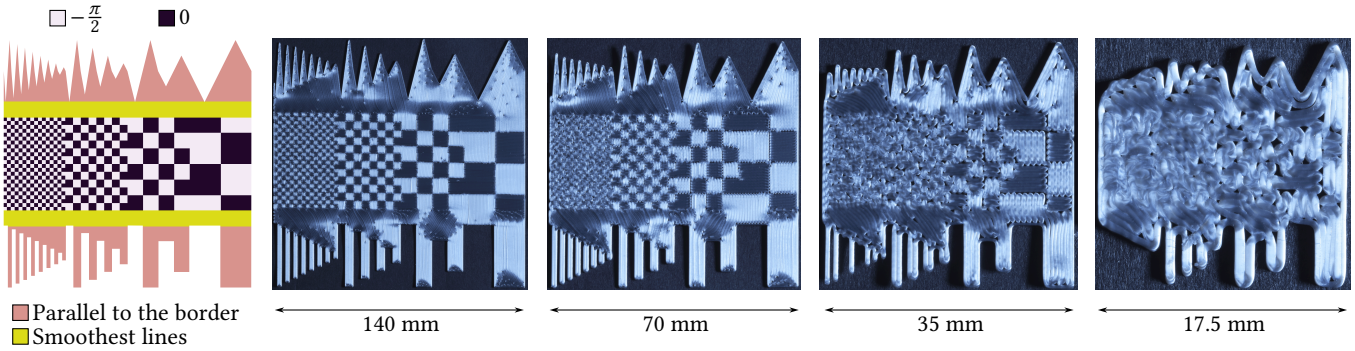


Fig. 13. Stress tests with a spatially-varying complexity. *Left*: The direction mode map and the boundary of the shape. *Other columns*: Photographs of the printed shape, with its size divided by increasing powers of two. Our algorithm robustly generates a cycle even if the resolution cannot capture the details.

Table 5. Coverage, overlap, and alignment energy (Ham. → Hamiltonian).

	Coverage		**Overlap**		**Alignment**	
	Ours	Ham.	Ours	Ham.	Ours	Ham.
LOGO	97.57 %	NA	0.78 %	NA	NA	NA
WAVE	96.75 %	NA	0.86 %	NA	-0.958	NA
BRAIN	95.91 %	94.43 %	1.13 %	3.30 %	-0.930	-0.863
PEOPLE	95.88 %	93.75 %	1.19 %	3.87 %	-0.924	-0.844
VARY	96.79 %	92.88 %	0.85 %	4.18 %	-0.946	-0.838

the prescribed line field $L_u : \mathbb{R}^2 \to S^1$ is also given. It is defined as

$$A(L_u, C_s) := \frac{1}{2\mathcal{L}(C_s)} \sum_{i \in \mathcal{V}_{C_s}} E_l(T_i, L_u(\gamma_i)^T)(\mathcal{L}(I_1) + \mathcal{L}(I_2)) \quad (17)$$

where $\mathcal{L}(C_s)$ is the length of the cycle and where $\mathcal{L}(I_1)$ and $\mathcal{L}(I_2)$ are the lengths of the first and second edges, *resp.* The term $L_u(\gamma_i)$ is the non-oriented direction at position γ_i and E_l is our measure of line alignment. As the lines' directions have scalar components between -1 and 1, the alignment energy A gives values between -1 (perfect alignment) and 0 (all the trajectories are orthogonal).

Pictures of the prints obtained with our method and the method of Bedel et al. are also visible (see Figure 12). Their approach has difficulty creating long trajectories following the input line field. As a consequence, the appearance is less glossy and anisotropic. This is partially explained by its lower coverage, higher overlap, and worse alignment compared to our method (see Table 5).

The distribution of bead width is visible in Figure 14 for both our method and Hamiltonian Cycle [Bedel et al. 2022]. Our algorithm gives bead width distributions with a peak centered or below the target spacing of $\tau = 0.4$. The distribution is skewed to the left, and the width density is concentrated above $\tau/2$.

Visual results. Figure 15 shows more photographs of planar printed results obtained with our method. We use the property that the trajectory's orientation is correlated to the amount of light scattered toward the observer to create fake bas-reliefs. A 3D shape lighted by a point light is rendered in grayscale mode, then the resulting image is used to constrain all the lines. We also demonstrate that mono-material QR codes can be manufactured with our method. In black areas, lines are horizontal; in white areas, lines are vertical. We used a nozzle with a 0.25 mm hole width for these results. Therefore, we put many details in relatively small areas. For example, the dragons' scales are visible while representing a small fraction of the model.

Visual flaws. There are some unintended visual flaws. Some points are darker or brighter than their surrounding area. They originate from sharp turns near singularities, as the turns locally produce all directions of anisotropy. The height field is also not perfectly flat due to possible under or overfilling. Consequently, areas with constant orientations do not have a uniform aspect.

7 LIMITATIONS AND FUTURE WORK

Our method comes with limitations and prospects for future work. The approach is inherently restricted to planar surfaces: a further step is to study the case where the input is a 3D surface, probably using a curved deposition method for printing [Etienne et al. 2019]. Moreover, the singularities of the implicit field can be placed in unwanted locations that may impact the visual appearance. This is apparent in Figure 1, where small marks are visible around the singularities. Thus, our method could be further enhanced with techniques that allow the user to place the singularities [Noma et al. 2022] to conceal them on the printed result.

Our approach is limited to high-contrast anisotropic appearances; it is unfeasible to control the amount of anisotropic roughness. Thus, it would be valuable to explore how further variations of the deposition paths influence the directional surface roughness and how this could be harnessed to reproduce different appearances. Eventually, we envision that surface roughness control through deposition trajectories could be used for the inverse design of optical [Cai and Shalaev 2010] or tactile metamaterials [Ion et al. 2018].

Our method could also be enhanced with FFF techniques that allow varying gradients of color [Song et al. 2019]. Apart from artistic and creative applications, our method could be applied in other sectors that are already adopting 3D printing, such as in construction, automobile, and whiteware sectors, where it is common to mechanically brush metal surfaces to give an aesthetic appeal to the manufactured components [1].

[1] https://www.imoa.info/download_files/stainless-steel/euroinox/Finishes.pdf

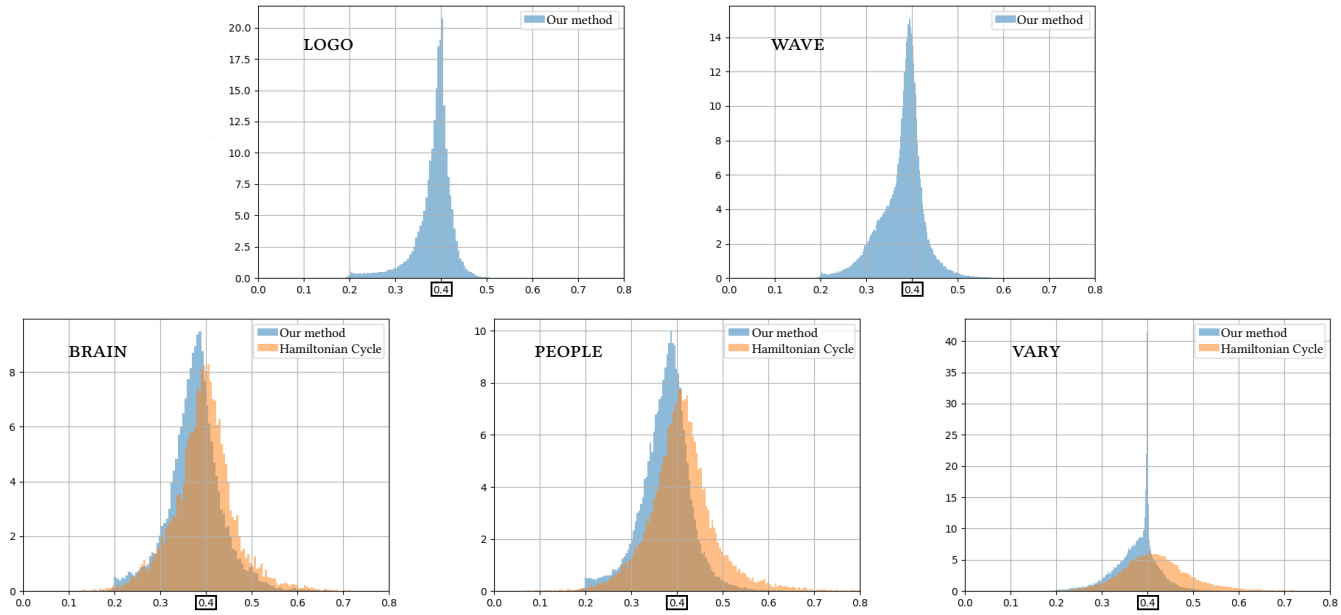


Fig. 14. The trajectory width distribution of each result, obtained with our method and Bedel et al. [2022] when applicable (bottom). Our distribution's peak is around the target width $\tau = 0.4$, and they are similar to the previous work's distributions, except for the VARY result where our trajectories are more aligned.

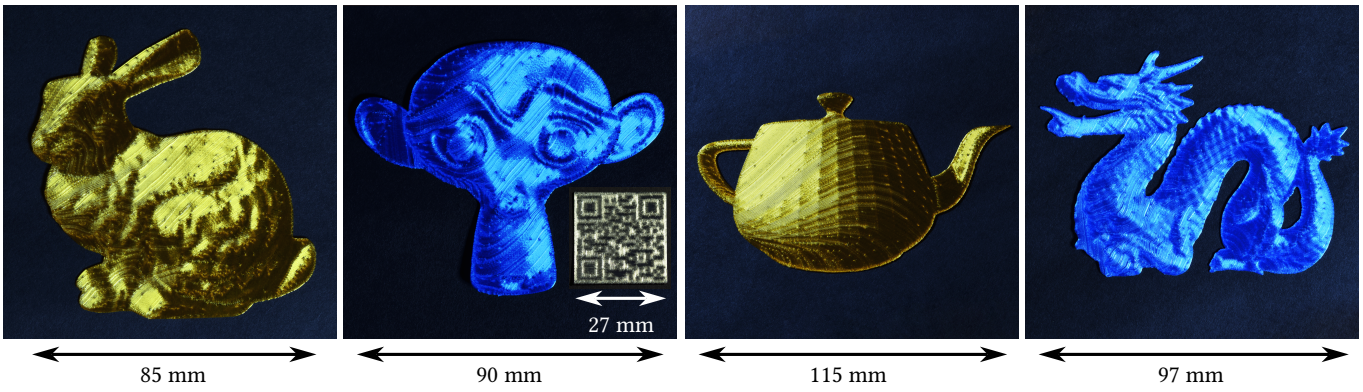


Fig. 15. Photographs of planar printed parts using FFF, our cycle generator, and a nozzle hole width of 0.25 mm. The trajectories' orientations are constrained with a rendering of the corresponding 3D object (Stanford bunny, Suzanne, coarse mesh of the Utah teapot, Chinese Dragon) or the SIGGRAPH 2023 QR code.

ACKNOWLEDGMENTS

We thank Régis Peignier of the Institut Jean Lamour for the roughness measurements, Marco Freire for mentioning the stitching of cycles, Nathaniel Seyler for the engineering support, and Nicolas Ray for providing good references on the optimization of periodic functions. The work is supported by the Agence nationale de la recherche of France under Grants No.: ANR-18-CE46-0004 and ANR-17-CE10-0002.

REFERENCES

Navid Ansari, Omid Alizadeh-Mousavi, Hans-Peter Seidel, and Vahid Babaei. 2020. Mixed Integer Ink Selection for Spectral Reproduction. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020). https://doi.org/10.1145/3414685.3417761

Vahid Babaei, Kiril Vidimče, Michael Foshey, Alexandre Kaspar, Piotr Didyk, and Wojciech Matusik. 2017. Color Contoning for 3D Printing. *ACM Trans. Graph. (Proc. SIGGRAPH)* 36, 4 (2017). https://doi.org/10.1145/3072959.3073605

J.A. Baerentzen and H. Aanaes. 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Trans. on Visualization and Comput. Graph.* 11, 3 (2005), 243–253. https://doi.org/10.1109/TVCG.2005.49

Frederick O Bartell, Eustace L Dereniak, and William L Wolfe. 1981. The theory and measurement of bidirectional reflectance distribution function (BRDF) and bidirectional transmittance distribution function (BTDF). In *Radiation scattering in optical systems*, Vol. 257. SPIE, 154–160.

A. Bedel, Y. Coudert-Osmont, J. Martínez, R. I. Nishat, S. Whitesides, and S. Lefebvre. 2022. Closed space-filling curves with controlled orientation for 3D printing. *Comput. Graph. Forum (Proc. of Eurographics)* 41, 2 (2022), 473–492.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.* http://github.com/google/jax

Wenshan Cai and Vladimir M Shalaev. 2010. *Optical metamaterials*. Vol. 10. Springer.

Zhao Dong, Bruce Walter, Steve Marschner, and Donald P. Greenberg. 2016. Predicting Appearance from Measured Microgeometry of Metal Surfaces. *ACM Trans. Graph.* 35, 1, Article 9 (2016), 13 pages.

Gershon Elber and Elaine Cohen. 1990. Hidden Curve Removal for Free Form Surfaces. In *Comput. Graph. (Proc. SIGGRAPH)*. 95–104. https://doi.org/10.1145/97879.97890

Jimmy Etienne and Sylvain Lefebvre. 2020. Procedural Band Patterns. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. Article 1, 7 pages. https://doi.org/10.1145/3384382.3384522

Jimmy Etienne, Nicolas Ray, Daniele Panozzo, Samuel Hornus, Charlie CL Wang, Jonàs Martínez, Sara McMains, Marc Alexa, Brian Wyvill, and Sylvain Lefebvre. 2019. CurviSlicer: Slightly curved slicing for 3-axis printers. *ACM Trans. Graph.* 38, 4 (2019).

Ben Ezair, Saul Fuhrmann, and Gershon Elber. 2018. Volumetric covering print-paths for additive manufacturing of 3D models. *Computer-Aided Design* 100 (2018), 1–13.

Guoxin Fang, Tianyu Zhang, Sikai Zhong, Xiangjia Chen, Zichun Zhong, and Charlie C. L. Wang. 2020. Reinforced FDM: Multi-Axis Filament Alignment with Controlled Anisotropic Strength. *Proc. ACM SIGGRAPH ASIA* 39, 6, Article 204 (2020), 15 pages. https://doi.org/10.1145/3414685.3417834

Xifeng Gao, Wenzel Jakob, Marco Tarini, and Daniele Panozzo. 2017. Robust Hex-Dominant Mesh Generation Using Field-Guided Polyhedral Agglomeration. *ACM Trans. Graph. (Proc. SIGGRAPH)* 36, 4 (2017).

Google. 2023. *Filament*. https://github.com/google/filament.

Eric Heitz. 2014. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. *Journal of Computer Graphics Techniques* 3, 2 (2014), 48–107.

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

Matthias B Hullin, Ivo Ihrke, Wolfgang Heidrich, Tim Weyrich, Gerwin Damberg, and Martin Fuchs. 2013. State of the art in computational fabrication and display of material appearance. In *Eurographics Annual Conference (STAR)*.

Alexandra Ion, Robert Kovacs, Oliver S Schneider, Pedro Lopes, and Patrick Baudisch. 2018. Metamaterial textures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

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

Andrew B Kahng and Sherief Reda. 2004. Match twice and stitch: a new TSP tour construction heuristic. *Operations Research Letters* 32, 6 (2004), 499–509. https://doi.org/10.1016/j.orl.2004.04.001

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe Patterns on Surfaces. *ACM Trans. Graph. (Proc. SIGGRAPH)* 34, 4, Article 39 (2015), 11 pages. https://doi.org/10.1145/2767000

Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural Noise Using Sparse Gabor Convolution. *ACM Trans. Graph. (Proc. SIGGRAPH)* 28, 3, Article 54 (2009), 10 pages. https://doi.org/10.1145/1531326.1531360

Yanxiang Lan, Yue Dong, Fabio Pellacini, and Xin Tong. 2013. Bi-Scale Appearance Fabrication. *ACM Trans. Graph. (Proc. SIGGRAPH)* 32, 4, Article 145 (2013), 12 pages. https://doi.org/10.1145/2461912.2461989

Anat Levin, Daniel Glasner, Ying Xiong, Frédo Durand, William Freeman, Wojciech Matusik, and Todd Zickler. 2013. Fabricating BRDFs at High Spatial Resolution Using Wave Optics. *ACM Trans. Graph. (Proc. SIGGRAPH)* 32, 4, Article 144 (2013), 14 pages. https://doi.org/10.1145/2461912.2461981

Marco Livesu, Stefano Ellero, Jonàs Martínez, Sylvain Lefebvre, and Marco Attene. 2017. From 3D models to 3D prints: an overview of the processing pipeline. *Comput. Graph. Forum (Proc. of Eurographics)* 36, 2 (2017), 537–564. https://doi.org/10.1111/cgf.13147

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Comput. Graph. (Proc. SIGGRAPH)* 21, 4 (1987), 163–169. https://doi.org/10.1145/37402.37422

A. Luongo, V. Falster, M. B. Doest, M. M. Ribo, E. R. Eiriksson, D. B. Pedersen, and J. R. Frisvad. 2020. Microstructure Control in 3D Printing with Digital Light Processing. *Comput. Graph. Forum* 39, 1 (2020), 347–359.

Yuta Noma, Nobuyuki Umetani, and Yoshihiro Kawahara. 2022. Fast Editing of Singularities in Field-Aligned Stripe Patterns. In *Proc. SIGGRAPH Asia*. Article 37, 8 pages. https://doi.org/10.1145/3550469.3555387

Peter Nyhuis. 2016. *Logistic Curves*. https://doi.org/10.1007/978-3-642-35950-7_7-3

Alexandrina Orzan, Adrien Bousseau, Pascal Barla, Holger Winnemöller, Joëlle Thollot, and David Salesin. 2013. Diffusion Curves: A Vector Representation for Smooth-Shaded Images. *ACM Trans. Graph. (Proc. SIGGRAPH)* 56, 7 (2013), 101–108. https://doi.org/10.1145/2483852.2483873

Alexios Papacharalampopoulos, Harry Bikas, and Panagiotis Stavropoulos. 2018. Path planning for the infill of 3D printed parts utilizing Hilbert curves. *Procedia Manufacturing (Proc. Global Conference on Sustainable Manufacturing)* 21 (2018), 757–764. https://doi.org/10.1016/j.promfg.2018.02.181

Sylvain Paris, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. 2008. Hair Photobooth: Geometric and Photometric Acquisition of Real Hairstyles. *ACM Trans. Graph.* 27, 3 (2008), 1–9.

Hans Pedersen and Karan Singh. 2006. Organic Labyrinths and Mazes. In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*. 79–86. https://doi.org/10.1145/1124728.1124742

Michal Piovarči, Michael Foshey, Vahid Babaei, Szymon Rusinkiewicz, Wojciech Matusik, and Piotr Didyk. 2020. Towards Spatially Varying Gloss Reproduction for 3D Printing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020).

Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic Global Parameterization. *ACM Trans. Graph.* 25, 4 (2006), 1460–1485. https://doi.org/10.1145/1183287.1183297

Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. N-Symmetry Direction Field Design. *ACM Trans. Graph. (Proc. SIGGRAPH)* 27, 2, Article 10 (2008), 13 pages. https://doi.org/10.1145/1356682.1356683

Mickaël Ribardière, Benjamin Bringier, Lionel Simonot, and Daniel Meneveaux. 2019. Microfacet BSDFs Generated from NDFs and Explicit Microgeometry. *ACM Trans. Graph.* 38, 5, Article 143 (June 2019), 15 pages. https://doi.org/10.1145/3338697

Tobias Rittig, Denis Sumin, Vahid Babaei, Piotr Didyk, Alexey Voloboy, Alexander Wilkie, Bernd Bickel, Karol Myszkowski, Tim Weyrich, and Jaroslav Křivánek. 2021. Neural Acceleration of Scattering-Aware Color 3D Printing. *Comput. Graph. Forum (Proc. of Eurographics)* 40, 2 (2021), 205–219. https://doi.org/10.1111/cgf.142626

Olivier Rouiller, Bernd Bickel, Jan Kautz, Wojciech Matusik, and Marc Alexa. 2013. 3D-Printing Spatially Varying BRDFs. *IEEE Computer Graphics and Applications* 33, 6 (2013), 48–57. https://doi.org/10.1109/MCG.2013.82

Pitchaya Sitthi-Amorn, Javier E. Ramos, Yuwang Wangy, Joyce Kwan, Justin Lan, Wenshou Wang, and Wojciech Matusik. 2015. MultiFab: A Machine Vision Assisted Platform for Multi-Material 3D Printing. *ACM Trans. Graph. (Proc. SIGGRAPH)* 34, 4, Article 129 (2015), 11 pages. https://doi.org/10.1145/2766962

B. Smith. 1967. Geometrical shadowing of a random rough surface. *IEEE Transactions on Antennas and Propagation* 15, 5 (1967), 668–671.

Haichuan Song, Jonàs Martínez, Pierre Bedell, Noémie Vennin, and Sylvain Lefebvre. 2019. Colored Fused Filament Fabrication. *ACM Trans. Graph.* 38, 5, Article 141 (2019), 11 pages. https://doi.org/10.1145/3183793

Robert J. Stango. 2002. Filamentary brushing tools for surface finishing applications. *Metal Finishing* 100 (2002), 82–91. https://doi.org/10.1016/S0026-0576(02)82007-4

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

Denis Sumin, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, Jaroslav Křivánek, Karol Myszkowski, and Tim Weyrich. 2019. Geometry-Aware Scattering Compensation for 3D Printing. *ACM Trans. Graph. (Proc. SIGGRAPH)* 38, 4 (2019), 111:1–111:14. https://doi.org/10.1145/3306346.3322992

Thibault Tricard, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, and Sylvain Lefebvre. 2019. Procedural Phasor Noise. *ACM Trans. Graph. (Proc. SIGGRAPH)* 38, 4, Article 57 (2019), 13 pages.

Thibault Tricard, Jimmy Etienne, Cedric Zanni, and Sylvain Lefebvre. 2021. A Brick in the Wall: Staggered Orientable Infills for Additive Manufacturing. In *Symposium on Computational Fabrication*. Article 7, 8 pages.

Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional Field Synthesis, Design, and Processing. *Comput. Graph. Forum (Proc. of Eurographics)* 35, 2 (2016), 545–572. https://doi.org/10.1111/cgf.12864

Stephen H Westin, James R Arvo, and Kenneth E Torrance. 1992. Predicting reflectance functions from complex surfaces. *ACM Siggraph Computer Graphics* 26, 2 (1992), 255–264.

Tim Weyrich, Pieter Peers, Wojciech Matusik, and Szymon Rusinkiewicz. 2009. Fabricating Microgeometry for Custom Surface Reflectance. *ACM Trans. Graph. (Proc. SIGGRAPH)* 28, 3, Article 32 (2009), 6 pages. https://doi.org/10.1145/1531326.1531338

Geoff Wyvill, Craig McPheeters, and Brian Wyvill. 1986. Data structure for soft objects. *The Visual Computer* 2, 4 (1986), 227–234. https://doi.org/10.1007/BF01900346

Chris Yu, Henrik Schumacher, and Keenan Crane. 2021. Repulsive Curves. *ACM Trans. Graph.* 40, 2 (2021).

Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2016. Connected Fermat Spirals for Layered Fabrication. *ACM Trans. Graph. (Proc. SIGGRAPH)* 35, 4, Article 100 (2016), 10 pages. https://doi.org/10.1145/2897824.2925958