Eurographics Symposium on Geometry Processing 2025
M. Attene and S. Sellán
(Guest Editors)

Volume 44 (2025), Number 5

Atomizer: Beyond Non-Planar Slicing for Fused Filament Fabrication

X. Chermain, G. Cocco, C. Zanni, E. Garner, P. A. Hugron, and S. Lefebvre

Université de Lorraine, CNRS, Inria, LORIA

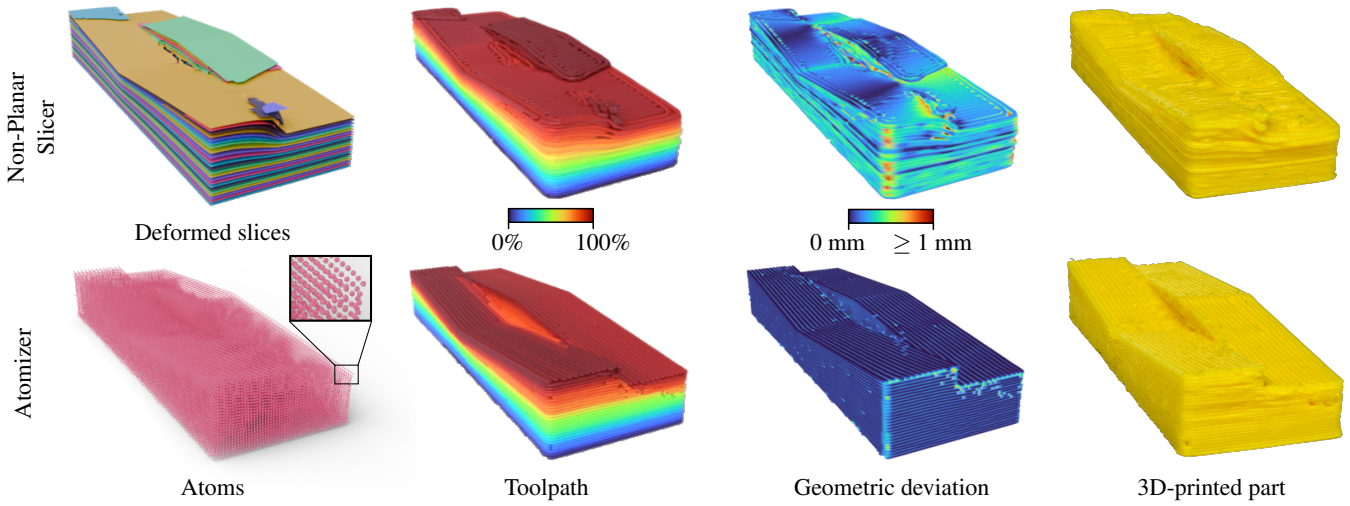


Figure 1: *Compared to non-planar slicers (top row), our Atomizer (bottom row) reduces geometric deviations from the input 3D model and improves the quality of the printed surface. This improvement is achieved by moving beyond the concept of slices and instead extracting the toolpath from evenly spaced atoms.*

Abstract

Fused filament fabrication (FFF) enables users to quickly design and fabricate parts with unprecedented geometric complexity, fine-tuning both the structural and aesthetic properties of each object. Nevertheless, the full potential of this technology has yet to be realized, as current slicing methods fail to fully exploit the deposition freedom offered by modern 3D printers. In this work, we introduce a novel approach to toolpath generation that moves beyond the traditional layer-based concept. We use frames, referred to as atoms*, as solid elements instead of slices. We optimize the distribution of atoms within the part volume to ensure even spacing and smooth orientation while accurately capturing the part's geometry. Although these atoms collectively represent the complete object, they do not inherently define a fabrication plan. To address this, we compute an extrusion toolpath as an ordered sequence of atoms that, when followed, provides a collision-free fabrication strategy. This general approach is robust, requires minimal user intervention compared to existing techniques, and integrates many of the best features into a unified framework: precise deposition conforming to non-planar surfaces, effective filling of narrow features – down to a single path – and the capability to locally print vertical structures before transitioning elsewhere. Additionally, it enables entirely new capabilities, such as anisotropic appearance fabrication on curved surfaces.*

CCS Concepts

• ***Applied computing*** → ***Computer-aided design;***

1. Introduction

Additive manufacturing is a transformative technology, offering unprecedented design freedom to both hobbyists and professionals. Extrusion-based methods, such as fused filament fabrication (FFF), in particular, provide tremendous flexibility in fabricating parts. By controlling extrusion trajectories, users can automatically optimize surface quality, dimensional accuracy, and even mechanical properties. However, limitations inherent to state-of-the-art techniques prevent designers from fully exploiting the true potential of this promising technology. Specifically, traditional methods for generating extrusion trajectories first partition the geometry into planar horizontal layers and then fill each layer with discrete trajectories. This decoupled process, while efficient, sacrifices much of the flexibility and design freedom that FFF can offer.

Recent advances in non-planar printing aim to overcome this limitation by relaxing the requirement that each layer be planar and horizontal. However, these methods still largely rely on a greedy approach that separates layer generation from trajectory planning. In the non-planar context, this does not guarantee collision-free paths or optimality.

This work extends beyond the notion of slices: we subdivide the object into a set of oriented solid elements, which we call *atoms*, and generate trajectories by optimizing their traversal in a fully three-dimensional (3D) context while considering fabricability constraints. By relying on atoms, our approach – termed the *Atomizer* – completely frees the generated toolpaths from layer ordering. Though challenging, this method enables a particularly robust implementation and supports unique features, such as producing an anisotropic appearance on curved surfaces.

Our core contributions are as follows:

- A method for distributing a set of oriented solid elements, referred to as *atoms*, evenly within a given part geometry while adhering to user-defined constraints. These constraints include conformity with geometric boundaries and alignment with user-specified orientations to achieve an anisotropic surface finish.
- A toolpath planner that optimizes for a *collision-free* traversal of the solid elements, generating a non-planar fabrication plan for extrusion processes.

We demonstrate our method on a 3D printer equipped with three independent Z-axis lead screws. We validate our technique by fabricating several parts on such a printer and provide measurements and comparisons in Section 7.

2. Related work

From its infancy, additive manufacturing has faced challenges stemming from the planar nature of fabricated layers [DM94]. The most recognizable is the so-called staircase defect, which degrades surface finish and part accuracy. Another issue is the weaker bonding between layers and across trajectories, particularly in extrusion-based processes (Fused Filament Fabrication), where structural strength is significantly higher *along* the deposition direction [RMFÖ17]. A third limitation is the strict layer-by-layer ordering, which is often suboptimal, especially for large builds. For instance, consider a vertical U-shaped part: at each layer, the extrusion device moves from one side of the U to the other. Instead, fabricating multiple layers on one side before switching to the other could be more efficient.

Different strategies have been employed to mitigate these issues. Part accuracy can be significantly improved by using *adaptive slicing* [KD96, SHB96], where the layer thickness and overall number of layers are optimized to minimize the difference between the 3D model and its physical counterpart [AHL17]. However, this never truly resolves the problem, as planar layers provide a poor discretization of sloped surfaces, and reducing layer thickness comes at the cost of significantly longer fabrication times. Layer-by-layer ordering can be re-optimized as a post-process, allowing for greater flexibility [KRBCS22, ZXZL23]. This is achieved by ordering connected components across layers to ensure that the extrusion device never collides with the part being built. Structural strength can be improved by optimizing the global orientation of the part [US13, UKY*15] and carefully selecting the infill pattern within each layer [WWZW16, TEZL21, LNAK*23]. However, the planar nature of the layers significantly limits the range of possible strategies. Consequently, several approaches have explored ways to exploit additional freedom of motion, moving beyond planar layers.

2.1. Thin shells and non-planar covers

A first set of techniques involves fabricating curved covers atop an otherwise standard planar fabrication [AWHZ19]. This approach works remarkably well for shapes with only top-most curved surfaces and shell-like parts [CARRC08, AT15]. However, for more complex parts, accessibility constraints quickly limit the ability to follow curved surfaces at different heights. A variant of this method adjusts the deposition thickness to align the fabricated layer tops with the actual surface tops [SRSL16]. However, this approach can lead to difficulties, as the nozzle may gouge into previously deposited material that is positioned higher. These techniques focus solely on top surfaces, without curving the interior or considering the orientation of deposition along curved layers.

2.2. Deformed planar slices

To fully curve a print without having to rebuild the entire process planning pipeline, Etienne et al. [ERP*19] proposed deforming the fabrication domain back and forth. The input part and its surrounding volume are deformed, sliced normally, and the resulting planar toolpaths are deformed back while adjusting for deposition parameters. While this initial technique focused on 3-axis printing, it was later extended to additional degrees of freedom [FZZ*20]. This led to a series of approaches aimed at increasingly complex optimizations for deformation, considering multiple objectives – reducing supports, increasing part strength, and improving surface accuracy [ZFH*22]. To mitigate the computational overhead of the required full-domain tetrahedralization, the latest techniques optimize deformation using a coarser tetrahedral deformation cage surrounding the object [LZC*24].

A major drawback of these techniques is the need to optimize for a continuous, smooth, and bijective deformation to ensure printable and non-intersecting deformed layers. Satisfying these constraints

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

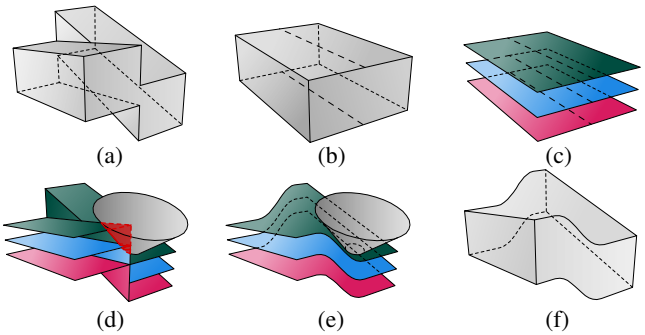


Figure 2: *Limitations of deformation-based slicing: The input model (a) undergoes deformation to generate conformal top and bottom slices (b). When the planar slices (c) are deformed back (d), collisions with the print head may occur. To prevent this, the layer slopes are smoothed (e); however, the final set of slices no longer precisely matches the original input model (f).*

requires a compromise on the curving objectives, potentially resulting in deformed layers that no longer conform to the top planar surfaces (Figure 2, (e)). The shape formed by the set of deformed slices may deviate from the input model (Figure 2, (f)), leading to large staircase defects and distortions (Figure 1).

The multi-axis case introduces further complications, as it is not possible to formalize the collision-free constraint during optimization. Additionally, constraining the maximum and minimum stretch in the general formulation becomes more challenging. Consequently, post-processing is applied to modify toolpaths by cutting and reordering them into a valid deposition plan, while overly thin and overly thick layers are either merged or split to achieve a roughly uniform inter-layer spacing [EFE18, ZFH*22]. Finally, these methods inherit from planar slicing the limitation of a layer-by-layer print ordering.

Instead of relying on the concept of slices or continuous deformation, our approach generates trajectories that achieve tight constraint alignment, even spacing, and a collision-free ordering, all without being restricted by a global slice sequence. We demonstrate its advantages in terms of part quality in Section 7.

2.3. Alternative techniques for non-planar fabrication

Several other methods have been explored that do not fit into the previous two categories. Gao et al. [GZN*15] explore fabrication of enclosures around a box-shaped cavity, printing in six different direction. Wu et al. [WDF*17] decompose a part into sub-regions which can print without support in the selected orientation. For both these methods, deposition is not truly curved, but rather multi-planar as each sub-region is fabricated along a single direction.

Song et al. [SRSL16] improve the accuracy and finish of top surface by slightly curving the top deposition layers. Dai et al. [DWW*18] create a non-planar deposition plan by peeling a voxel representation, under an accessibility and supportability constraint. This strongly reduces, and often eliminates the need for support structure. Ezair et al. [EFE18] extract layers as iso-surfaces and trajectories as iso-lines from a provided tri-variate parametrization, taking care of maintaining an equal spacing between deposition paths. The hierarchical subdivision however introduces a specific pattern of porosity that can be detrimental to surface finish and structural strength (see Figure 4 in [TEZL21]). Our method does not require an existing parametrization, and avoids this issue by optimizing for an even distribution of atoms filling the part volume. The method of Ezair et al. splits the initial curves and re-orders them into collision-free strands. Our algorithm follows a similar strategy but works from a soup of optimized atoms without prior knowledge of connectivity. Our heuristics are designed to further encourage continuity and penalize travel time.

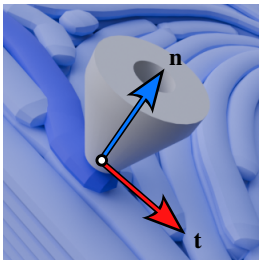
2.4. Beyond layer-per-layer ordering

A number of techniques consider improving the fabrication order, questioning the simpler approach of fabricating layers from bottom to top. This was first seen in slicer software for multiple part fabrication, where parts are fabricated one after the other rather than all at once layer-per-layer (e.g. Cura "one at a time" print mode).

Later techniques generalize the process to achieve reduced travel within the same part. This is done as a post-process, extracting a graph representing trajectories across layers and their accessibility [KRBCS22, ZXZL23]. A new plan is then produced by greedily printing nearby trajectories across layers, provided this does not render any other not-yet-printed trajectory inaccessible. Remarkably, our method automatically leads to such improved planning, without requiring any post-processing.

3. Method overview and setup

Our method generates the toolpaths required for fabricating a given 3D solid. A toolpath is a polyline where each vertex encodes a position, an orientation, and the quantity of material to be extruded. During fabrication, the extrusion device follows this path with the specified orientation, depositing material downward. The material extruded by the print head is locally characterized by a deposition height h and a deposition width τ, which our method aims to keep constant. Without loss of generality, we assume that the deposition height is half the deposition width, i.e., $h = \tau/2$.



The pipeline of our method is illustrated in Figure 3. The toolpaths are optimized so that extrusion occurs with an orientation $\mathbf{n}$ along a tangent $\mathbf{t}$. Both $\mathbf{n}$ and $\mathbf{t}$ are derived from automatically placed constraints and optimized accordingly, as described in Sections 4.1 and 4.2, respectively. The set of atoms is then optimized based on these target orientation and tangent directions (Section 4.3). Finally, the atoms are ordered according to fabrication objectives to form a set of collision-free toolpaths (Section 5). The toolpaths can be exported as GCode, depending on the targeted printer. In our method, the toolpath represents the centerline of the deposition trajectory. At the end of the pipeline,

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

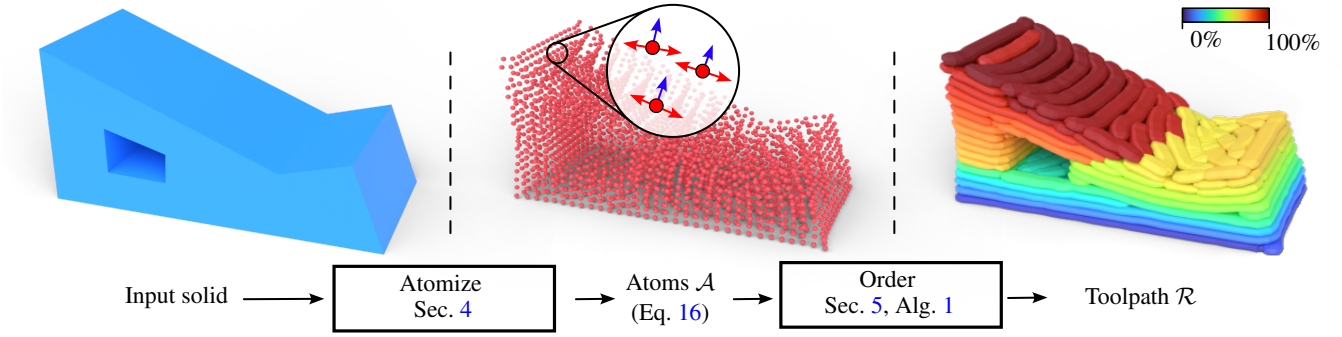


Figure 3: *Illustration of the Atomizer pipeline. An input solid is atomized, producing a set of unordered atoms $\mathcal{A}$. These atoms are then ordered to generate the toolpath $\mathcal{R}$. Blue arrows show tool head orientations; red arrows show unoriented toolpath tangents.*

it is adjusted to match the correct position of the extrusion nozzle (Section 6).

The three main inputs to our approach are the targeted deposition width, τ, the maximum allowable tool inclination, $\Theta \in [0, \pi/2)$, and a 3D solid. To achieve specific properties in the fabricated shape, constraints on $\mathbf{n}$ and $\mathbf{t}$ can be imposed. For example, we constrain the tangent using a direction texture to achieve an anisotropic appearance. Next, we detail the setup of our method, including the input solid, domain discretization, and necessary definitions.

Input solid. We consider the solid as a signed distance field (SDF), though other representations can be used if they define both the *sideness* and the distance to the boundary. The signed distance of the point $\mathbf{p}$ from the boundary is given by $\mathrm{SDF}(\mathbf{p}) \in \mathbb{R}$. SDFs are practical for manufacturing, as they encode information about the interior ($\mathrm{SDF}(\mathbf{p}) < 0$, Figure 4, left), boundary ($\mathrm{SDF}(\mathbf{p}) = 0$), and exterior ($\mathrm{SDF}(\mathbf{p}) > 0$). Notably, a watertight mesh can be efficiently converted into an SDF [BA05].

Domain discretization. The Atomizer requires discretizing the solid domain into volume elements (traditional voxels – not to be confused with atoms) to compute geometric data within the solid. In particular, as we detail below, the voxels store the target tool orientation and the target deposition tangents.

We use a set of voxels, denoted as $\mathcal{V}$, which are cubic cells of identical size, aligned with the three axes of the domain. The number of voxels is given by $|\mathcal{V}|$. Each voxel $i \in \mathcal{V}$ corresponds to a cell

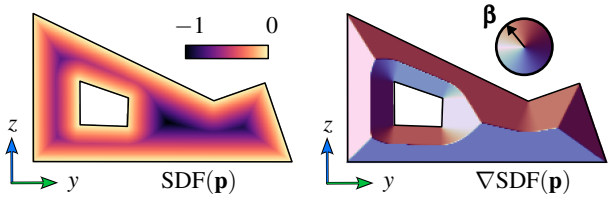


Figure 4: *The input solid is represented as an SDF. Left: A 2D SDF illustration. Right: Its gradient approximates the closest normal to the boundary.*

in a uniform grid and has a neighborhood $\mathcal{N}(i)$ consisting of the 26 surrounding cells. We denote by $\mathbf{p}_i \in \mathbb{R}^3$ the center of voxel i. In our geometric setting, the smallest geometric detail corresponds to the deposition height h. According to the Nyquist-Shannon sampling theorem, the spatial sampling period should be at most half the deposition height, i.e., $h/2$, to prevent geometric aliasing. We set the voxel side length r to three-quarters of this value, i.e., $r = \frac{3}{8}h$, ensuring a sufficiently high spatial sampling rate along all three axes of the domain. With this setting, the voxel diagonal length l is given by $l = \frac{3\sqrt{3}}{8}h$ indicating that the sampling rate along diagonals slightly exceeds the ideal sampling rate. Finally, we define a *masked voxel* as a voxel whose center lies outside or on the boundary of the solid, i.e., $\mathrm{SDF}(\mathbf{p}_i) \geq 0$.

Direction difference definition. As we manipulate orientations, we need to compare directions and normals. We define the difference $\Delta_{\mathbf{d}} \in [0, 1]$ between two directions $\mathbf{d}_1$ and $\mathbf{d}_2$ on the unit sphere S^2 as

$$\Delta_{\mathbf{d}}(\mathbf{d}_1, \mathbf{d}_2) := \frac{-\mathbf{d}_1 \cdot \mathbf{d}_2 + 1}{2}. \tag{1}$$

The difference $\Delta_{\mathbf{d}}$ is equal to one when the two directions are opposite and zero when they are identical.

Solid geometric properties. We assume that each voxel center, $\mathbf{p}_i$, has a unique closest point on the solid surface boundary. If multiple closest points exist, one is chosen arbitrarily. We denote by $\boldsymbol{\beta}_i \in S^2$ the normal at the closest point on the surface. The normal is defined as the gradient of the signed distance field, i.e., $\boldsymbol{\beta}_i := \nabla\mathrm{SDF}(\mathbf{p}_i)$ (Figure 4, right), computed using central finite differences. Furthermore, we define $\kappa_i \in [0, 1]$ as a scalar representing the local variation of the normal around voxel i. This quantity is computed via central finite differences by averaging, along the three coordinate axes, the local normal differences measured using the direction difference of Equation 1. These geometric properties are essential for defining the tool orientation and direction constraints in subsequent steps.

4. Atomizing process

We compute the atoms in three steps, each beginning with the setup of constraints in a few specific voxels, which are then propagated

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

through optimization to the rest of the domain. The three steps sequentially solve for the target tool orientation (Section 4.1), the target deposition tangents (Section 4.2), and finally the atom positions (Section 4.3). The combination of this information forms the final atoms.

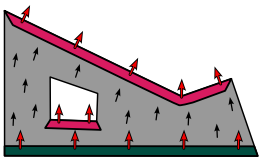
Decoupled optimization. The steps of atom computation are decoupled into three substeps. This separation is intentional and serves to simplify the complex problem of generating aligned frames within a volume. Simultaneously handling tool orientation, toolpath tangents, and equal spacing of toolpath vertices would require solving a highly complex optimization problem. By decoupling these aspects, each field can be optimized individually, making the process more manageable and computationally efficient. This design choice also enhances flexibility, as it allows for the use of different optimization techniques tailored to the specific requirements of each field. Finally, the decoupled approach improves scalability and performance, as each substep can be parallelized and scaled independently. Decoupled optimization has proven its efficiency in generating quad-dominant [JTPSH15] and hex-dominant [GJTP17] meshes, as well as in curved layer generation [FZZ*20], fiber printing [ZLD*25], and multi-axis machining [DZF*23].

4.1. Tool orientation computation

We associate each voxel i with a target tool orientation $\mathbf{n}_i \in \mathrm{H}^2$, where H^2 represents the upper hemisphere. This tool orientation is automatically constrained in voxels where it must follow curved surfaces and is then propagated throughout the domain under a smoothness objective.

The orientation $\mathbf{n}_i := (x_{\mathbf{n}_i}, y_{\mathbf{n}_i}, z_{\mathbf{n}_i})^T$ represents the normal of the deposition trajectory and can be described using spherical coordinates $(\theta_{\mathbf{n}_i}, \phi_{\mathbf{n}_i}) \in [0, \pi/2] \times [-\pi, \pi]$. Specifically, these spherical coordinates are defined as $\theta_{\mathbf{n}_i} := \arccos(z_{\mathbf{n}_i})$ and $\phi_{\mathbf{n}_i} := \operatorname{arctan2}(y_{\mathbf{n}_i}, x_{\mathbf{n}_i})$. To ensure a consistent fabrication process oriented from bottom to top, we constrain the polar angle $\theta_{\mathbf{n}_i}$ to the range $[0, \pi/2]$, ensuring that the tool orientation always points upward.

Tool orientation constraints. We constrain the tool orientation to align with the upward direction ($\mathbf{n}_i = (0,0,1)^T$) for the first deposition height ($z_{\mathbf{p}_i} < h$, green area in the figure), as this ensures proper adhesion to the printer bed. To achieve conformal top deposition trajectories and avoid staircase artifacts, we constrain the tool orientation to align with the closest boundary surface normal ($\mathbf{n}_i = \boldsymbol{\beta}_i$) when voxel i lies within a top surface region (highlighted in red in the Figure). A voxel is considered part of a top surface region if its distance to the boundary is less than the deposition height and the polar angle of its closest boundary normal, $\theta_{\boldsymbol{\beta}_i}$, lies within the interval $[0, \Theta]$, where $\Theta \in [0, \pi/2]$ represents the maximum allowable tool inclination. This set is defined as $\{i \in \mathcal{V} \mid \mathrm{SDF}(\mathbf{p}_i) \in [-h, 0) \text{ and } \theta_{\boldsymbol{\beta}_i} \in [0, \Theta]\}$. If the voxel normal deviation is high ($\kappa_i > 0.1$), the voxel is not constrained to avoid sudden tool orientation changes. In the next section, we explain how the non-masked and unconstrained voxels are filled from the constrained tool orientations.



Tool orientation propagation. The main objective for the propagation is to obtain a smooth field. Indeed, sudden changes in orientation require pauses during extrusion to allow the tool to complete its rotation. These pauses leave marks on the part because the plastic continues to flow during rotation. The goal is to minimize the tool's orientation changes and, consequently, the resulting defects.

We define the average of the tool orientation local differences, $\Delta_{\mathcal{O}} \in [0, 1]$, as

$$\Delta_{\mathcal{O}} := \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \left(\frac{\sum_{j \in \mathcal{N}(i)} w_{ij} \Delta_{\mathbf{d}} (\mathbf{d}_i, \mathbf{d}_j)}{W_i} \right), \tag{2}$$

where the weight w_{ij} decreases linearly with the distance between the voxel centers $\mathbf{p}_i$ and $\mathbf{p}_j$, assigning less weight to neighboring voxels on the diagonal. The term $W_i := \sum_{j \in \mathcal{N}(i)} w_{ij}$ represents the sum of the neighboring weights.

We minimize the orientation changes, $\Delta_{\mathcal{O}}$, using a local iterative method. This type of method relies on local smoothing operations that are applied iteratively. Specifically, each unconstrained orientation, $\mathbf{n}_i$, is updated to the normalized weighted average of its non-masked neighboring orientations:

$$\mathbf{n}_i \leftarrow \overline{\mathbf{n}}_i := \frac{\sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{n}_j}{\left|\left| \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{n}_j \right|\right|} \tag{3}$$

to reduce the global difference $\Delta_{\mathcal{O}}$. Masked neighboring voxels j have associated weight zeros ($w_{ij} = 0$). A multiresolution grid is employed to accelerate convergence (see Section 6).

4.2. Deposition tangent computation

We associate a deposition tangent $\mathbf{t}_i \in S^2$ with each non-masked voxel i and denote the field of tangents as $\mathcal{T}$. This field represents the desired alignment of deposition within the solid volume. At this stage of the pipeline, the tangents are considered non-oriented, as there is no need to constrain the nozzle to move in a specific direction. Consequently, the tangent field $\mathcal{T}$ is a *2-directional field*, specifying two possible directions, $(\mathbf{t}_i, -\mathbf{t}_i)$, for each voxel i. We aim for a smooth tangent field to minimize unnecessary tool movements and sharp turns. Since the tool orientation $\mathbf{n}_i$ is now fixed (Section 4.1), the set of possible tangents lies within the *tangent plane* defined by the voxel normal $\mathbf{n}_i$. As a result, each tangent can also be represented as a *2D Cartesian direction*, denoted by the unit vector $\hat{\mathbf{t}}_i \in S^1$. In the next section, we describe where the tangents are constrained and how these constraints are propagated to all voxels to obtain a smooth field.

Deposition tangent constraints. To achieve better surface quality, it is important for the toolpath tangent to remain parallel to the boundary of the solid, ensuring a contour-parallel outline. This is achieved by constraining the tangent to be the cross product of the tool orientation and the closest boundary normal in the wall region: $(\mathbf{t}_i = \boldsymbol{\beta}_i \times \mathbf{n}_i / \|\boldsymbol{\beta}_i \times \mathbf{n}_i\|)$. The wall region is defined as $\{i \in \mathcal{V} \mid \mathrm{SDF}(\mathbf{p}_i) \in [-\tau, 0) \text{ and } \theta_{\boldsymbol{\beta}_i} \in [\pi/4, 3\pi/4]\}$. The range

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

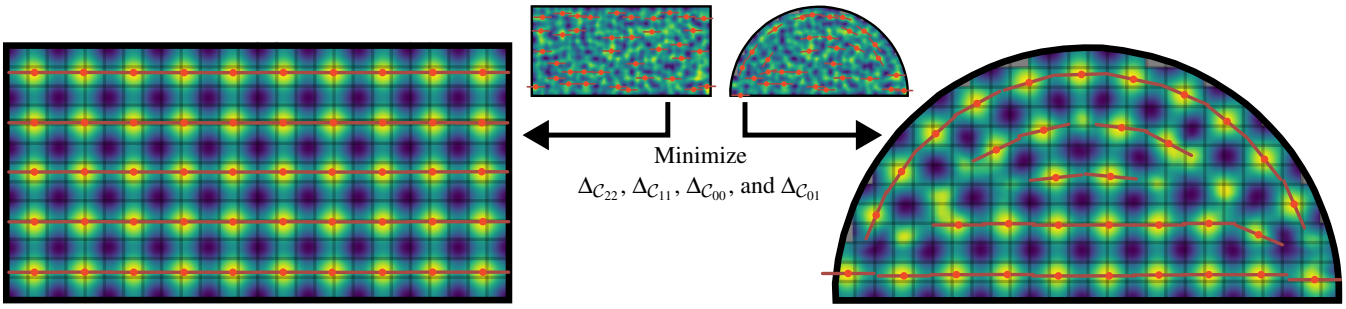


Figure 5: *Slices* $(\mathbf{t},\mathbf{n})$ *of tricosine fields* $\mathcal{C}$ *(Eq. 14) inside a box and a half-cylinder (black boundaries). The atoms are represented with their tangents* $(-\frac{h}{2}\mathbf{t}_i, \frac{h}{2}\mathbf{t}_i)$ *and correspond to the local maxima (Eq. 15) of the tricosine field* $\mathcal{C}$*. In the middle region, the cosines have random phases. Minimizing* $\Delta_{\mathcal{C}_{22}}$*,* $\Delta_{\mathcal{C}_{11}}$*,* $\Delta_{\mathcal{C}_{00}}$*, and* $\Delta_{\mathcal{C}_{01}}$ *(Eq. 9) encourages the toolpath to be correctly spaced. The atom position constraints at half a layer height promote alignment of the toolpath with the top and bottom surfaces.*

of angles $[\pi/4, 3\pi/4]$ implies that walls are considered to tilt up to a maximum of 45 degrees. The wall region corresponds to the blue area in Figure 6, left. If the voxel normal deviation exceeds a threshold ($\kappa_i > 0.1$), then the voxel boundary normal $\boldsymbol{\beta}_i$, computed using finite differences, may be inaccurate. In this case, we do not impose the constraint.

Deposition tangent propagation. To obtain a smooth, continuous deposition, we seek to minimize the global tangent deviation. Let M_i be the transformation matrix that maps a neighboring tangent $\mathbf{t}_j$ into the tangent space of voxel i. We define the neighboring tangent vector $\mathbf{t}_j$ in the tangent space of voxel i as $\tilde{\mathbf{t}}_j \coloneqq M_i\mathbf{t}_j$, and its projection onto the tangent plane as $\tilde{\mathbf{t}}_j^{\perp} \coloneqq (x_{\tilde{\mathbf{t}}_j}, y_{\tilde{\mathbf{t}}_j})^T$ (see Figure 6, right).

The set of weighted neighboring tangents is represented by a 26×2 matrix $T_i \coloneqq \left(\sqrt{w_{ij}}\tilde{\mathbf{t}}_j^{\perp}\right)^T_{j\in\mathcal{N}(i)}$. We define the local tangent deviation, $\Delta_{T_i} \in [0,1]$, as:

$$\Delta_{T_i} \coloneqq -\frac{\hat{\mathbf{t}}_i^T T_i^T T_i \hat{\mathbf{t}}_i}{W_i} + 1, \tag{4}$$

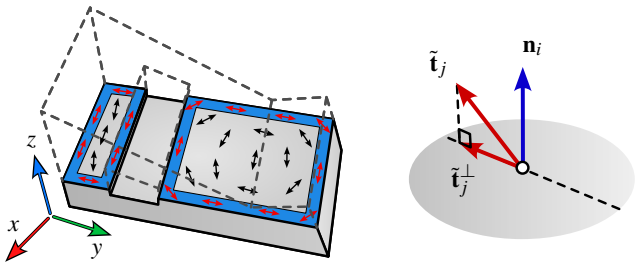


Figure 6: *Left: The desired deposition orientation is set parallel to the boundary in the wall region (blue region). Within the volume, the tangents are smoothed to ensure consistency. Right: To smooth the tangents in a voxel i, each neighboring tangent* $\mathbf{t}_j$ *is transformed into the tangent plane defined by* $\mathbf{n}_i$*. The transformed tangent* $\tilde{\mathbf{t}}_j$ *is then projected onto this plane, yielding* $\tilde{\mathbf{t}}_j^{\perp}$*.*

where $\hat{\mathbf{t}}_i \in S^1$ is the 2D Cartesian tangent of voxel i, and W_i is the sum of the neighboring weights. A deviation of zero indicates that all neighboring tangents are perfectly aligned with $\mathbf{t}_i$. The global tangent deviation $\Delta_T \in [0,1]$ is defined as the average of the local deviations:

$$\Delta_T \coloneqq \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \Delta_{T_i}. \tag{5}$$

To minimize the global tangent deviation, we use a local iterative method based on a multiresolution grid, similar to the one employed for the tool orientation field. At each grid level, during each iteration, each unconstrained toolpath tangent is updated to the average of the neighboring tangents, which corresponds to the eigenvector with the largest eigenvalue of the matrix $T_i^T T_i$. For further details on the multigrid method for aligning unoriented directions, please refer to Chermain et al. [CZM*23].

After the tangent alignment process, each voxel i is associated with an orthonormal basis $(\mathbf{t}_i, \mathbf{b}_i, \mathbf{n}_i)$, where the bitangent $\mathbf{b}_i \coloneqq \mathbf{n}_i \times \mathbf{t}_i$ defines the direction in which an adjacent parallel toolpath should lie. As for the toolpath tangent field, the bitangent field is a 2-directional field, specifying two possible directions, $(\mathbf{b}_i, -\mathbf{b}_i)$.

4.3. Atom positions computation

Given the now fully determined tool orientation and deposition tangent fields, we optimize the atom positions for regular spacing between them. The toolpath points, i.e., the atom positions, are defined as the local maxima of a 3D scalar field $\mathcal{C}$ (Eq. 14), called the *tricosine field*. This scalar field is the average of three periodic functions whose orientations are orthogonal. In this work, we use three cosine plane waves, $\mathcal{C}_0$, $\mathcal{C}_1$, and $\mathcal{C}_2$ (Eq. 12); their alignment and blending result in evenly spaced local maxima that represent the toolpath points (Figure 5). This implicit representation facilitates toolpath alignment, as there is no need to handle connectivity between toolpath vertices in this domain. Using periodic functions as an implicit representation of toolpaths has been successfully applied in fused filament fabrication: two periodic functions were used to represent a diamond-shaped microstructure [TTZ*20] or a staggered infill [TEZL21], and a single periodic function was used to represent orientable 2D toolpaths [CZM*23, JHS*23, ZLD*25].

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

Each voxel i is associated with three orthogonal cosine plane waves, denoted as (c_{i0}, c_{i1}, c_{i2}), whose directions are given by $(\mathbf{d}_{i0}, \mathbf{d}_{i1}, \mathbf{d}_{i2}) \coloneqq (\mathbf{t}_i, \mathbf{b}_i, \mathbf{n}_i)$. The corresponding periods are $(p_{i0}, p_{i1}, p_{i2}) \coloneqq (h, \tau, h)$, and their phases are $(\varphi_{i0}, \varphi_{i1}, \varphi_{i2})$. Each cosine plane wave c_{ik}, where $k \in \{0,1,2\}$, is defined as

$$c_{ik}(\mathbf{p}) \coloneqq \cos\left(\frac{2\pi}{p_{ik}}(\mathbf{p}-\mathbf{p}_i)\cdot\mathbf{d}_{ik}+\varphi_{ik}\right). \tag{6}$$

For a masked voxel, we set the value of $c_{ik}(\mathbf{p})$ to -1. These cosine plane waves, when averaged together, implicitly represent toolpath points as illustrated in Figure 7. The figure also details how the periods of c_{i0}, c_{i1} and c_{i2} are chosen with respect to the deposition height and width.

We denote the *tricosine* c_i as the set of three orthogonal cosine plane waves, (c_{i0}, c_{i1}, c_{i2}), and the tricosine field $\mathcal{C}$ as the field of tricosine voxels. We use the subscript k to refer to the field of the k^{th} cosine plane wave. For example, $\mathcal{C}_2$ represents the strata defined along the local tool orientation direction. The objective is to align the tricosines between adjacent cells to obtain a smooth tricosine field that represents the atom positions (Figure 5).

Atom positions constraints. To ensure that the deposition is aligned with the bottom and top surfaces of the solid, each phase φ_{i2} is constrained in these regions. The set of voxels within the top surface regions is defined as $\{i \in \mathcal{V} \mid \text{SDF}(\mathbf{p}_i) \in [-l, 0) \text{ and } \theta_{\boldsymbol{\beta}_i} \in [0, \Theta]\}$, where l is the voxel diagonal length. Similarly, the set of voxels within the bottom surface regions is given by $\{i \in \mathcal{V} \mid \text{SDF}(\mathbf{p}_i) \in [-l, 0) \text{ and } \theta_{\boldsymbol{\beta}_i} \in [\pi - \Theta_b, \pi]\}$, where $\Theta_b \in [0, \pi/4]$ is the bottom slope maximum angle, set to one degree in our implementation. The phase φ_{i2} of voxels in the top surface regions is constrained to $\varphi_T \coloneqq 2\pi(\text{SDF}(\mathbf{p}_i)/h + 1/2)$, while for voxels in the bottom surface regions is constrained to $-\varphi_T$. In this case, the cosine plane wave has to be shifted in the opposite direction.

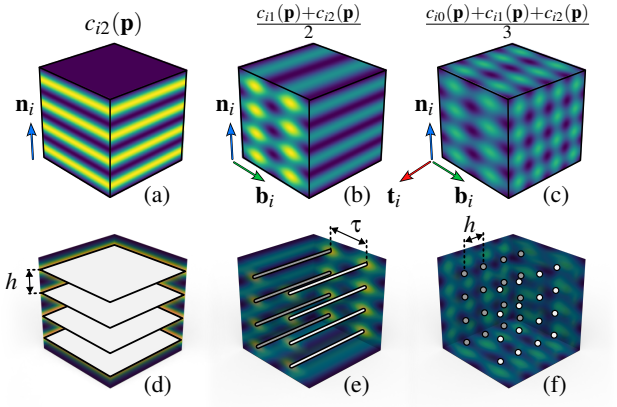


Figure 7: *Implicit representation of the toolpath points. In a voxel i, strata are implicitly defined as the ridge surfaces (d) of the cosine plane wave c_{i2}, defined along the normal $\mathbf{n}_i$ (a). Toolpaths are represented as the ridge curves (e) of the average of the two cosine plane waves, c_{i1} and c_{i2} (b). Toolpath points (f) are the maxima of the scalar field defined as the average of the three cosines, c_{i0}, c_{i1}, and c_{i2} (c). The spacing between strata is set to the deposition height h (d). Along the bitangent $\mathbf{b}_i$, the spacing between toolpaths is set to the deposition width τ (e). Finally, the spacing between toolpath points along the tangent $\mathbf{t}_i$ is set to the layer height h (f).*

The ability to match the alignment of the top and bottom surfaces with phase is a significant advantage over deformation-based approaches (Section 2.2), in which the alignment of slices with the actual surface tops is either heuristic [ERP*19] or relies on a carefully chosen manual parameter [ZFH*22].

To ensure that the deposition trajectories are aligned with the boundary, each phase φ_{i1} is constrained in the wall region, defined as $\{i \in \mathcal{V} \mid \text{SDF}(\mathbf{p}_i) \in [-l, 0) \text{ and } \theta_{\boldsymbol{\beta}_i} \in [\pi/4, 3\pi/4]\}$. Each phase φ_{i1} is set to $\varphi_W \coloneqq 2\pi(\text{SDF}(\mathbf{p}_i)/\tau + 1/2)$ if $\boldsymbol{\beta}_i \cdot \mathbf{b}_i \geq 0$, and to $-\varphi_W$ otherwise. There are two cases because the bitangent b_i is not always oriented along the gradient of the SDF since b_i is a non-oriented direction.

Atom positions optimization. We consider two adjacent cosine plane waves associated to voxel $i \in \mathcal{V}$ and $j \in \mathcal{N}(i)$ to be *aligned* when they have the same value at their midpoint $\mathbf{p}_{ij} \coloneqq (\mathbf{p}_i + \mathbf{p}_j)/2$. The deviation $\Delta_c \in [0,1]$ between two cosines c_{ik} and c_{jm}, where $(i,j) \in \mathcal{V}^2$ and $(k,m) \in \{0,1,2\}^2$, is defined as

$$\Delta_c(c_{ik}, c_{jm}) \coloneqq |c_{ik}(\mathbf{p}_{ij}) - c_{jm}(\mathbf{p}_{ij})|/2. \tag{7}$$

Given two cosine plane waves c_{ik} and c_{jm}, it is possible to align c_{ik} with c_{jm} to achieve zero deviation ($\Delta_c(c_{ik}, c_{jm}) = 0$) by setting the phase φ_{ik} of c_{ik} to

$$\overline{\varphi}_{ikjm} \coloneqq \underset{\varphi_{ik}}{\text{argmin}}\, \Delta_c(c_{ik}, c_{jm}) = \begin{cases} b-a, & \text{if } \mathbf{d}_{ik}\cdot\mathbf{d}_{jm} > 0 \\ -b-a, & \text{otherwise,} \end{cases} \tag{8}$$

where $a \coloneqq \frac{2\pi}{p_{ik}}(\mathbf{p}_{ij} - \mathbf{p}_i)\cdot\mathbf{d}_{ik}$ and $b \coloneqq \frac{2\pi}{p_{jm}}(\mathbf{p}_{ij} - \mathbf{p}_j)\cdot\mathbf{d}_{jm} + \varphi_{jm}$. The alignment is shown in Figure 8.

Given $(k,m) \in \{0,1,2\}^2$, we define the global cosine plane wave deviation $\Delta_{\mathcal{C}_{km}} \in [0,1]$ in terms of the local deviation Δ_c as

$$\Delta_{\mathcal{C}_{km}} \coloneqq \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \frac{\sum_{j\in\mathcal{N}(i)} w_{ij}|\mathbf{d}_{ik}\cdot\mathbf{d}_{jm}|\Delta_c(c_{ik}, c_{jm})}{\sum_{j\in\mathcal{N}(i)} w_{ij}|\mathbf{d}_{ik}\cdot\mathbf{d}_{jm}|}. \tag{9}$$

To obtain a smooth tricosine field with minimal gaps between

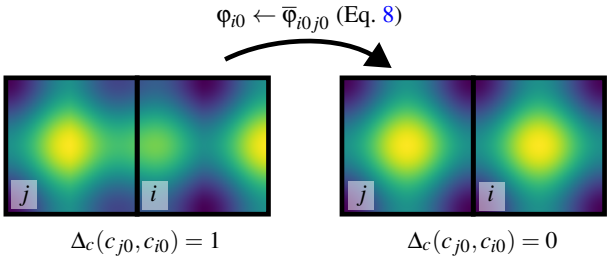


Figure 8: *Illustration of a slice $(\mathbf{t}, \mathbf{n})$ of a tricosine field $\mathcal{C}$ composed of two voxels i and j. Left: The two cosines, c_{j0} and c_{i0}, are out of phase and cancel each other when mixed at the boundary between the two voxels. Right: the phase φ_{i0} is now aligned with j's cosine c_{j0}, leading to phased signals. Here, the cosines' period is equal to the voxel width.*

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

adjacent local maxima – i.e., the trajectories – we minimize $\Delta_{\mathcal{C}_{22}}$, $\Delta_{\mathcal{C}_{11}}$, $\Delta_{\mathcal{C}_{00}}$, and $\Delta_{\mathcal{C}_{01}}$ (Fig. 5). A proper atom density is promoted by setting the cosine periods to the layer height h along the tangent and the normal directions, and to the deposition width τ along the bitangent direction. Minimizing $\Delta_{\mathcal{C}_{22}}$ aligns the strata, i.e., encouraging the local vertical spacing to be h. Similarly, minimizing $\Delta_{\mathcal{C}_{11}}$ aligns the atoms such that the spacing between parallel toolpaths will be τ. Minimizing $\Delta_{\mathcal{C}_{00}}$ aligns the atoms such that the spacing between points along the toolpath will be h. Minimizing $\Delta_{\mathcal{C}_{01}}$ aligns atoms that would be along the toolpath with toolpaths running orthogonally to them. In this case, the endpoints of the toolpaths are repelled from the orthogonal toolpaths ahead, promoting proper spacing.

To minimize these objectives, we use a multigrid method that relies on local iterative smoothing operations within a multiresolution grid. In each iteration, we align each cosine plane wave c_{ik} with its neighboring cosine plane waves c_{jm} using Equation 8. To account for all neighbors, the aligned phases are averaged:

$$\varphi_{i2} \leftarrow \arg(\eta_{i2|2}), \quad \varphi_{i1} \leftarrow \arg(\eta_{i1|1}), \quad \varphi_{i0} \leftarrow \arg(\eta_{i0|0} + \eta_{i0|1}), \tag{10}$$

where

$$\eta_{ik|m} \coloneqq \sum_{j \in \mathcal{N}(i)} w_{ij} |\mathbf{d}_{ik} \cdot \mathbf{d}_{jm}| e^{\imath\, \overline{\varphi}_{ikjm}}. \tag{11}$$

Tricosine field evaluation. To avoid discontinuities when evaluating the cosine field $\mathcal{C}_k(\mathbf{p})$, we interpolate the adjacent cosine plane wave values $c_{jk}(\mathbf{p})$ using

$$\mathcal{C}_k(\mathbf{p}) \coloneqq \frac{\sum_{j \in \mathcal{N}(i) \cup \{i\}} w_j(\mathbf{p}) c_{jk}(\mathbf{p})}{\sum_{j \in \mathcal{N}(i) \cup \{i\}} w_j(\mathbf{p})}. \tag{12}$$

Here, the voxel i is the one containing the evaluation point $\mathbf{p}$, and the spatial weighting function is defined as

$$w_j(\mathbf{p}) \coloneqq \max\left(0, r - ||\mathbf{p} - \mathbf{p}_j||\right)/r, \tag{13}$$

where the weight takes a value of one when $\mathbf{p} = \mathbf{p}_j$ and decreases linearly to zero as the distance between the evaluation point $\mathbf{p}$ and the voxel center $\mathbf{p}_j$ reaches the voxel side length r.

The evaluation of the tricosine field, $\mathcal{C}(\mathbf{p})$, is defined as the average of the values of three cosine fields:

$$\mathcal{C}(\mathbf{p}) \coloneqq \frac{\mathcal{C}_0(\mathbf{p}) + \mathcal{C}_1(\mathbf{p}) + \mathcal{C}_2(\mathbf{p})}{3}. \tag{14}$$

The tricosine field, $\mathcal{C}(\mathbf{p})$, is used to extract the atom positions (Figure 9). The proposed blending fades out when two wave functions are not in phase, e.g., when they have a phase shift of π (see Figure 8). This property prevents extractions when atoms are too close, but may create gaps with a diverging tangent field (Figure 5, right).

Implicit to explicit positions. The atom positions are defined as the local maxima of the tricosine scalar field $\mathcal{C}(\mathbf{p})$. We now describe the process of extracting these points. For each voxel i, we look for the point $\mathbf{p}_i^{\max}$ with the highest tricosine value: $\mathbf{p}_i^{\max} \coloneqq \operatorname{argmax}_{\mathbf{p} \in i} \mathcal{C}(\mathbf{p})$ where $\mathbf{p} \in i$ means that the point $\mathbf{p}$ is inside the voxel i. The search for the maximum point $\mathbf{p}_i^{\max}$ in each voxel i is performed using a brute-force approach, based on a 4^3 uniform space sampling of the voxel.

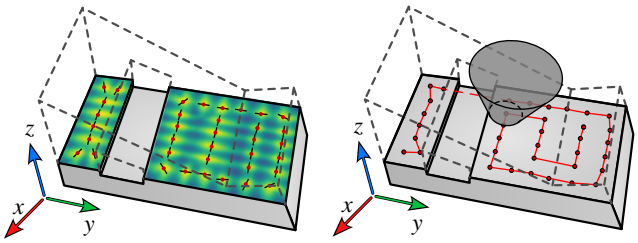


Figure 9: *Left: Illustration of the tricosine field $\mathcal{C}$ (Eq. 14) and its maxima, which correspond to the atom positions (Eq. 16). Each atom $i \in \mathcal{A}$ is associated with a tangent $\mathbf{t}_i$. Right: The atoms are ordered (Section 5) to construct a toolpath.*

The set of local maximum voxels $\mathcal{V}_A$ consists of voxels whose point $\mathbf{p}_i^{\max}$ has a tricosine value $\mathcal{C}(\mathbf{p}_i^{\max})$ greater than that of all its neighboring voxels:

$$\mathcal{V}_A \coloneqq \{i \in \mathcal{V} \mid \forall j \in \mathcal{N}(i),\ \mathcal{C}(\mathbf{p}_i^{\max}) > \mathcal{C}(\mathbf{p}_j^{\max})\}. \tag{15}$$

The atom positions correspond to the set of local maximum points $\mathbf{p}_i^{\max}, i \in \mathcal{V}_A$. Each local maximum point is associated with a tool orientation $\mathbf{n}_i$, a deposition tangent $\mathbf{t}_i$, and a bitangent $\mathbf{b}_i$. The set of frames $\mathcal{A}$, referred to as the set of *unordered atoms*, is defined as

$$\mathcal{A} \coloneqq \{(\mathbf{p}_i^{\max}, \mathbf{t}_i, \mathbf{b}_i, \mathbf{n}_i) \mid i \in \mathcal{V}_A\}. \tag{16}$$

In the next section, this set is ordered to construct the final toolpath (Figure 9, right).

5. Ordering atoms for fabrication

The objective is to create a fabrication plan in which the extrusion device visits the atoms in a valid and efficient sequence. The most critical fabrication constraint is the risk of collisions. Visiting atoms after others have already been fabricated can lead to situations where the 3D printer collides with the part. This issue is trivially resolved when printing with flat planar layers but becomes challenging with non-planar deposition and an orientable extruder. Overall, the ordering problem can be formulated as a single-vehicle task planning problem with precedence constraints [BCY*21], an NP-hard problem that we address using dedicated heuristics.

We propose an ordering algorithm that always generates a collision-free fabrication plan by integrating precedence constraints into the toolpath planner. To mitigate the risk of producing a low-continuity toolpath fragmented into numerous small segments, we employ a multistep heuristic ordering process that integrates both accessibility and supportability constraints (Section 5.1), with a backtracking mechanism designed to prioritize constraining atoms and promote longer continuous paths (Section 5.5). We also penalize fragmented toolpaths and favor deposition continuity through cost-based heuristics (Section 5.4). While these strategies are local, they are carefully designed to encourage global consistency and robust fabricability.

5.1. Fabricability constraints

Atom $i \in \mathcal{A}$ must be visited before atom $j \in \mathcal{A}$ if i supports j or if j blocks the 3D printer access to i. To check these constraints, each

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

atom i is associated with a *supporting cone* with apex $\mathbf{p}_i^{\max}$, axis $\mathbf{n}_i$, height $1.5h$, and an aperture of 130 degrees. Additionally, each atom has an *accessibility cone* α_i with the same apex and axis, but with infinite height and a user-defined aperture that should be large enough to encompass the extrusion device (80 degrees in our case).

Atom i supports atom j if j is inside its *supporting cone* and is beyond half the deposition height along $\mathbf{n}_i$, i.e., $\mathbf{n}_i \cdot (\mathbf{p}_j^{\max} - \mathbf{p}_i^{\max}) > 0.5h$. We denote this region as P_i^{above} (Figure 10, left). Atom j blocks access to atom i if j is inside the *accessibility cone* of i, i.e., $\mathbf{p}_j^{\max} \in \alpha_i$. In both cases, i must be visited before j, meaning that i is a predecessor of j and j is a successor of i.

5.2. Viable atoms

Inspired by Bai et al. [BCY*21], the ordering problem with *precedence constraints* can be solved by iteratively inserting *viable* atoms into a path until all atoms are inserted. A *viable* atom is one that can be inserted into the path without violating any precedence constraints. The toolpath can be constructed in either forward or backward mode. In forward mode, construction starts from the starting point, and viable atoms are those without predecessors. In backward mode, atoms are added starting from the endpoint, and viable atoms are those without successors.

We decided not to store the precedence constraint graph, as its memory complexity is $|\mathcal{A}|^2$. This decision requires evaluating the precedence constraints on the fly, thereby trading memory complexity for time complexity.

The choice between forward and backward ordering depends on the efficiency of determining whether an atom has any predecessors or successors. The supportability constraint is a local constraint and can be evaluated efficiently in both modes. In contrast, the accessibility constraint is easier to evaluate when moving backward than when moving forward. For the sake of the argument, let us consider only the collision constraints. When moving forward, an atom i is viable if it is not within the accessibility cone of any other atom. With N the number of atoms remaining to be inserted, this would require N intersection tests. In contrast, when moving backward, an atom i is viable if no other atom is within its accessibility cone. Determining whether at least one atom lies within a cone can be accelerated using a spatial data-structure. This possibility led us to solve the ordering problem in a backward manner. As a reminder, in backward mode, a viable atom is one that is not supporting and is accessible.

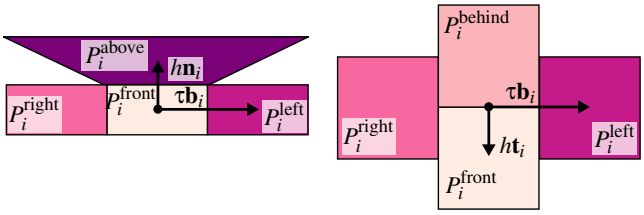


Figure 10: *Illustration of the neighborhood regions of atom i. Left: Front view. Right: Top view.*

5.3. Backward toolpath planning

Our toolpath planner is based on the cheapest insertion heuristic [RSL77]. Let $\mathcal{R}$ denote the ordered sequence of atoms currently inserted into the path, where $\mathcal{R}_{\text{last}}$ refers to the most recently inserted atom. We define $\overline{\mathcal{R}}$ as the unordered set of atoms that have not yet been included. Let $\mathcal{T}$ represent the set of viable atoms within $\overline{\mathcal{R}}$. The cost of moving from atom i to atom j, denoted as $C(i, j)$, is defined later.

The next cheapest atom $j^\star$ to be inserted in an iteration is defined as

$$j^\star := \operatorname*{argmin}_{j \in \mathcal{T}} C(\mathcal{R}_{\text{last}}, j). \tag{17}$$

except for the first iteration, where $\mathcal{R}_{\text{last}}$ is replaced by i_z^+, the index of the highest atom. Insertion updates the toolpath as $\mathcal{R} \leftarrow (\mathcal{R}, j^\star)$. After inserting $j^\star$ into the toolpath, we remove $j^\star$ from $\overline{\mathcal{R}}$, i.e., $\overline{\mathcal{R}} \leftarrow \overline{\mathcal{R}} \setminus j^\star$, and the set of viable atoms $\mathcal{T}$ is updated accordingly. The insertion procedure continues until all atoms are included in the toolpath, as summarized in Algorithm 1. Note that the final toolpath must be reversed before being used by the machine, as it is constructed backwards.

Assuming there are no cycles in the precedence graph, Algorithm 1 always terminates, as $\mathcal{T}$ is never empty. The resulting deposition toolpaths are collision-free, as atoms chosen in $\mathcal{T}$ are those that are freely accessible at each step, ensuring that the precedence constraints are never violated. Note, however, that travel moves between deposition toolpaths require special treatment. We defer this discussion to Section 6, where implementation details are provided.

Algorithm 1: Toolpath planner for ordering atoms $\mathcal{A}$.

Data: The unordered set of atoms $\mathcal{A}$
Result: The toolpath $\mathcal{R}$

$\mathcal{R} \leftarrow \varnothing$, $\overline{\mathcal{R}} \leftarrow \mathcal{A}$, $\mathcal{T} \leftarrow \text{viable}(\mathcal{A})$
$i \leftarrow i_z^+$
while $\overline{\mathcal{R}} \neq \varnothing$ **do**
  $j^\star \leftarrow \text{argmin}_{j \in \mathcal{T}} C(i, j)$ (Eq. 17)
  if $j^\star$ adjacent to a constraint
    // Backtrack to last deposition trajectory endpoint
    $j^\star, j_b, \mathcal{R}, \overline{\mathcal{R}}, \mathcal{T} \leftarrow \text{backtrack}(j_b)$ // see Sec. 5.5
  if travel required from i to $j^\star$
    // Backtracking checkpoint
    $j_b \leftarrow i$ // see Sec. 5.5
  $\mathcal{R} \leftarrow (\mathcal{R}, j^\star)$
  $\overline{\mathcal{R}} \leftarrow \overline{\mathcal{R}} \setminus j^\star$
  Update the set $\mathcal{T}$ of viable atoms
  $i \leftarrow \mathcal{R}_{\text{last}}$
$\mathcal{R} \leftarrow$ Reverse $\mathcal{R}$

5.4. Tool motion cost

The computation of the tool motion cost requires the relative position of an atom j with respect to an atom i. We define the regions

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

in front of, and behind atom i (Figure 10) as

$$P_i^{\text{front/behind}} := \left\{ \mathbf{p} \in \mathbb{R}^3 \;\middle|\; \begin{array}{l} |\mathbf{t}_i \cdot \mathbf{p}_{ij}^{\max}| < 2h \text{ and} \\ |\mathbf{b}_i \cdot \mathbf{p}_{ij}^{\max}| < 0.5\tau \text{ and} \\ |\mathbf{n}_i \cdot \mathbf{p}_{ij}^{\max}| < 0.5h, \end{array} \right\} \tag{18}$$

where $\mathbf{p}_{ij}^{\max}$ is the vector from atom i to atom j, i.e., $\mathbf{p}_{ij}^{\max} := \mathbf{p}_j^{\max} - \mathbf{p}_i^{\max}$. A point $\mathbf{p}$ belongs to the front region P_i^{front} if $\mathbf{t}_i \cdot \mathbf{p}_{ij}^{\max} > 0$, and to the behind region P_i^{behind} otherwise. In the latter dot product, $\mathbf{t}_i$ is oriented in the direction of the toolpath being grown to define the front and back.

Similarly, we define the regions to the left and right of atom i as

$$P_i^{\text{left/right}} := \left\{ \mathbf{p} \in \mathbb{R}^3 \;\middle|\; \begin{array}{l} |\mathbf{t}_i \cdot \mathbf{p}_{ij}^{\max}| < h \text{ and} \\ |\mathbf{b}_i \cdot \mathbf{p}_{ij}^{\max}| \in [0.5\tau, 1.5\tau] \text{ and} \\ |\mathbf{n}_i \cdot \mathbf{p}_{ij}^{\max}| < 0.5h, \end{array} \right\} \tag{19}$$

where a point $\mathbf{p}$ belongs to the left region P_i^{left} if $\mathbf{b}_i \cdot \mathbf{p}_{ij}^{\max} > 0$, and to the right region P_i^{right} otherwise. Here, the orientation of b_i is unimportant.

Our toolpath planner also considers the distance between atoms. Let $d_{ij} \in [0,1]$ be the Euclidean distance between two atoms, normalized by the length of the diagonal of the axis-aligned bounding box of the solid.

To determine the next cheapest atom to insert in an iteration, the cost $C(i,j)$ of moving from atom i to atom j is needed. The algorithm used to compute this cost is defined in Algorithm 2.

When searching for the next cheapest atom to insert into the toolpath $\mathcal{R}$, we first prioritize the closest atom in front of the last inserted atom $i := \mathcal{R}_{\text{last}}$ (Alg. 2, line 2). If no such atom is available, we then prioritize the closest atom behind the last inserted one (Alg. 2, line 4).

If no atom exists in $P_i^{\text{front/behind}}$, the deposition process is considered to stop, requiring the tool to travel to a new deposition trajectory starting point. In this case, atoms that are endpoints or part of a cycle are preferred, as we do not want to restart in the middle of an open path. Specifically, an atom j incurs an additional cost if: there is another atom $k \in \mathcal{T}$ in front of it ($\mathbf{p}_k^{\max} \in P_j^{\text{front}}$) and there is another atom $l \in \mathcal{T}$ behind it ($\mathbf{p}_l^{\max} \in P_j^{\text{behind}}$) and it is not part of a deposition trajectory cycle (defined by P_j^{front} adjacency). This penalty mechanism is implemented in Algorithm 2, line 7, and illustrated in Figure 11(a).

An additional penalty applies to atoms located between two adjacent deposition trajectories. Specifically, an atom j has its cost increased if: There is another atom $k \in \mathcal{T}$ to its left ($\mathbf{p}_k^{\max} \in P_j^{\text{left}}$) and there is another atom $l \in \mathcal{T}$ to its right ($\mathbf{p}_l^{\max} \in P_j^{\text{right}}$). This condition is implemented in Algorithm 2, line 9, and illustrated in Figure 11(b).

5.5. Removing constrainers first

To achieve a smooth deposition, it is important to encourage continuous toolpaths. Often, a toolpath will be followed until a constraint prevents further growth. This can happen multiple times due to the same constraint, leading to fragmentation of the toolpaths. To address this, we design the following backtracking process to remove such constraints first.

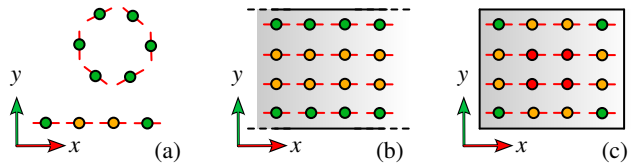


Figure 11: *To avoid segmenting deposition trajectories, we increase the moving cost for atoms that are neither part of a cycle and nor endpoints (a). Additionally, we penalize atoms that are surrounded by adjacent deposition trajectories on both sides (b). The combination of these two heuristics helps to start deposition trajectories from corners (c).*

Algorithm 2: Computation of the tool moving cost $C(i,j)$.

1 **if** $\mathbf{p}_j^{\max} \in P_i^{\text{front}}$
2 $\quad C(i,j) \leftarrow d_{ij}$ // If j is in front of i
3 **else if** $\mathbf{p}_j^{\max} \in P_i^{\text{behind}}$
4 $\quad C(i,j) \leftarrow d_{ij} + 1$ // If j is behind i
5 **else**
6 $\quad C(i,j) \leftarrow d_{ij} + 2$
7 $\quad$ **if** $\exists (k,l) \in \mathcal{T}^2 \mid \mathbf{p}_k^{\max} \in P_j^{\text{front}}$ and $\mathbf{p}_l^{\max} \in P_j^{\text{behind}}$, and $j \notin$ a cycle
8 $\quad\quad C(i,j) \leftarrow C(i,j) + 1$
9 $\quad$ **if** $\exists (k,l) \in \mathcal{T}^2 \mid \mathbf{p}_k^{\max} \in P_j^{\text{left}}$ and $\mathbf{p}_l^{\max} \in P_j^{\text{right}}$
10 $\quad\quad C(i,j) \leftarrow C(i,j) + 1$

If an atom i has an unviable atom $u \in \overline{\mathcal{R}} \setminus \mathcal{T}$ adjacent to it, i.e., if $\mathbf{p}_u^{\max} \in P_i^{\text{front/behind}}$ or $\mathbf{p}_u^{\max} \in P_i^{\text{left/right}}$, then we backtrack to the last deposition trajectory endpoint j_b. After backtracking, the last inserted atom $i := \mathcal{R}_{\text{last}}$ is the last atom in the sequence without any atoms in front of, or behind, in the set of remaining points.

Then, to find the next cheapest atom $j^\star$ to insert into the toolpath $\mathcal{R}$, we use a different cost function, $C_B(i,j)$, which prioritizes the constrainers of the unviable atom u. The set of constrainers of u is defined as $\mathcal{B} := \{b \in \mathcal{T} \mid \mathbf{p}_b^{\max} \in P_u^{\text{above}}$ or $\mathbf{p}_b^{\max} \in \alpha_u\}$. To prioritize the constrainers, we add five to the classic cost $C(i,j)$ when j is not a constrainer of u, with five being the maximum cost normally obtained. This results in the following:

$$C_B(i,j) := \begin{cases} C(i,j), & \text{if } j \in \mathcal{B}, \\ C(i,j) + 5, & \text{otherwise.} \end{cases} \tag{20}$$

Backtracking can occur sequentially (Figure 12, left). For example, after an initial backtracking step, we may initiate a deposition trajectory with a constrainer belonging to a partially inaccessible path. A backtracking sequence terminates if the deposition trajectory starts from an atom that has already served as a starting point in the current sequence. This approach prevents infinite backtracking to the same atom in cases where a deposition trajectory goes below itself (Figure 12, right).

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

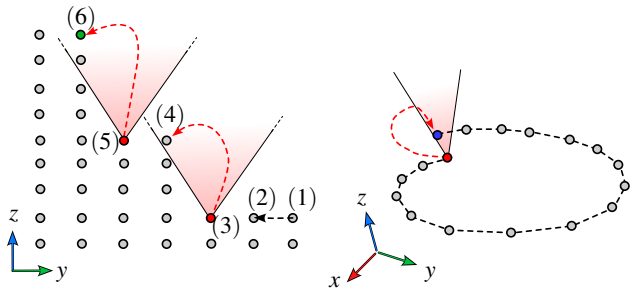


Figure 12: *Starting deposition trajectories with constrainers. Left: A sequence of two backtracking steps is shown. We assume that the last deposition trajectory endpoint has atom index (0) (not illustrated) and that each atom has* $\mathbf{t}_i \parallel y$ *and* $\mathbf{n}_i \parallel z$*. A first deposition trajectory starts at (1) and ends at (2), which is adjacent to the unviable atom (3), leading to a first backtrack to (0). The second deposition trajectory begins at (4), as it is a constrainer for (3). However, atom (4) is adjacent to another unviable atom (5), resulting in a second backtrack to (0). Finally, the new starting atom (6) does not belong to a partially inaccessible deposition trajectory. Right: Infinite backtracking can occur if a trajectory goes below itself. To prevent this, we terminate a backtracking sequence if a deposition trajectory starts with an atom that has previously been used as a starting point.*

6. Implementation

Atomizer. We implemented the Atomizer in Python 3.10 using the Taichi domain-specific language [HLA*19]. The source code is available at https://github.com/xavierchermain/atomizer. To minimize deviations in tool orientations, deposition tangents, and cosine plane waves (Equations 2, 5, and 9), we employed a custom multigrid method developed with Taichi's CUDA backend. The multigrid method is based on independent local averaging operations (Equations 3 and 11) using the classic V-cycle approach. First, we build the grid hierarchy by averaging only the constraints (restriction step), then perform N smoothing iterations per level before prolonging the smoothed values to the finer level. We use 64 iterations per level for tool orientation and toolpath tangent smoothing, and 128 iterations per level for aligning the cosine plane waves. The process consists of four V-cycles: first, we compute the tool orientation field, then the toolpath tangent fields, followed by the alignment of strata by minimizing $\Delta_{\mathcal{C}_{22}}$, and finally, we align the remaining cosine plane waves by minimizing $\Delta_{\mathcal{C}_{11}}, \Delta_{\mathcal{C}_{00}}$, and $\Delta_{\mathcal{C}_{01}}$ in the last V-cycle.

Toolpath planner. For the ordering (Section 5), we use the single-threaded CPU backend of Taichi. As explained earlier, we do not store the explicit precedence constraint matrix. To determine the set of viable atoms $\mathcal{T}$, we first identify the set of non-supporting atoms $\overline{\mathcal{S}}$ and then search for the accessible atoms within $\overline{\mathcal{S}}$. The determination of $\overline{\mathcal{S}}$ is based on a local neighborhood, accelerated by a bounding volume hierarchy (BVH). For the first iteration, $\overline{\mathcal{S}}$ is obtained by iterating through all atoms i and checking whether another atom exists in P_i^{above}. For subsequent iterations, $\overline{\mathcal{S}}$ is updated by iterating only within the neighborhood of the next cheapest atom, $j^\star$. The accessibility check using the accessibility cone α_i is also accelerated via a BVH. To further reduce computation time, we update the accessibility and in-cycle states only when the tool stops depositing material, i.e., when the cost $C(i,j) \geq 2$.

Travels without material deposition require special treatment to ensure collision-free navigation. We sample these paths at intervals of 0.2τ and check whether any of the sampled points collide with the remaining atoms to be ordered. The tool orientation at each sampled travel point is determined by interpolating the tool orientations at the endpoints of the travel. If any sampled point collides, it is iteratively moved away from the part by two deposition heights, $2h$, along its tool orientation until no collision remains.

Strata orthogonal to walls. We added the option to prioritize strata that are orthogonal to the walls, improving surface quality and local self-supportability. This impacts the optimization of the tool orientation described in Section 4.1. Since there are an infinite number of valid tool orientations for strata to be orthogonal to the walls, we perform the optimization in two steps. First, we optimize only accounting for strata conforming to top surfaces, as described previously. Then, we introduce additional constraints in the wall region (the wall region defined in Section 4.2): the constrained tool orientation is defined as the vector $\mathbf{n}$, orthogonal to the surface ($\boldsymbol{\beta}_i \cdot \mathbf{n} = 0$), and best aligned with the tool orientation $\mathbf{n}_i$ computed in the previous step, i.e., $\mathbf{n} \leftarrow (\boldsymbol{\beta}_i \times \mathbf{n}_i) \times \boldsymbol{\beta}_i$, normalized. The tool orientations $\mathbf{n}_i$ are then re-optimized with these additional constraints.

Implementation details. To enhance print quality by reducing the number of tool orientation changes, we apply 32 smoothing iterations – including constraints – during the first V-cycle. This process improves surface quality, as shown in Figure 13. Additionally, we filter out deposition trajectories that are shorter than two atoms in length. Finally, at the end of the toolpath generation, all points are lifted along the tool orientation by half of the deposition height ($0.5h$). This adjustment accounts for the fact that an atom represents the center of the trajectory, whereas the machine requires tool positions (GCode) corresponding to the top of the trajectory.

Figure 13: *Smoothing the constraints helps reduce sudden change in the tool orientation and improves surface quality (left to right).*

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

Table 1: *Computational statistics for the Atomizer pipeline.*

Model	Dim. (mm)	Figures	Θ	#Voxels	#Atoms	Memory (MB)	Computational time (min)		
							Atomizing	Ordering	Total
Cube	$20 \times 20 \times 21$	Fig. 13, 15, 20	7°	1,770k	43k	99	1.2	14.6	15.8
Tubes	$39 \times 41 \times 44$	Fig. 15, 17, 20	7°	14,872k	53k	833	1.5	15.5	17.0
Slopes	$24 \times 75 \times 12$	Fig. 1, 15, 20	7°	4,582k	98k	270	1.5	104.0	105.5
Pisa Tower	$53 \times 53 \times 95$	Fig. 15, 20	5.5°	54,318k	132k	3,554	1.8	136.7	138.5
Dome	$85 \times 85 \times 4$	Fig. 18	5.5°	6,605k	69k	370	1.5	137.6	139.1
Cubic	$91 \times 75 \times 27$	Fig. 18	5.5°	5,026k	77k	287	1.5	227.7	229.2
City	$65 \times 67 \times 64$	Fig. 16, 17, 20	5.5°	57,914k	186k	3,243	1.5	336.7	338.2
Car	$41 \times 84 \times 22$	Fig. 16	45°	14,992k	234k	840	1.6	417.0	418.6
Ankle	$63 \times 102 \times 31$	Fig. 16	45°	40,534k	243k	2,312	1.4	446.0	447.4

7. Results

We use nine 3D models, depicted in Figure 14, to evaluate our method and compare it to other techniques. The dimensions and relevant characteristics of the models are provided in Table 1, which also includes the maximum relative tool inclination angle Θ used to generate the toolpath for each model. Next, we briefly describe the models, followed by Sections 7.1 to 7.6, which are dedicated to analyzing various aspect of the results.

The Cube model is a variant of the classical calibration cube used by hobbyists to calibrate their 3D printers, with the top surface tilted by 6 degrees to test non-planar methods. The Tubes model consists of a thin-walled tube manifold that branches from a single tube into three smaller tubes oriented differently. The Slopes model features inverted slopes meeting in the middle, separated by a vertical wall. The Pisa Tower model includes several parallel, tilted floors. The Dome and Cubic models are designed to support an anisotropic appearance on their top surfaces. The City model is designed to challenge the orderability of non-planar methods, with ground and building rooftops also tilted. Finally, the Car and Ankle models provide additional comparisons in a virtual setup that allows for more inclination than our 3D printer can currently achieve.

7.1. Computational time and memory usage

Experiments were performed on a desktop with an Intel Core i7-4770K CPU @ 3.50GHz, 32 GB of RAM, and an NVIDIA GeForce RTX 2080 Ti with 11 GB of memory, running Windows 10. The memory usage and computational time are shown in Table 1 for each model. The number of atoms and voxels is related to the volume of each model. Both the voxel count and memory usage are linearly related to the volume of the axis-aligned bounding box of the model, whose dimensions are provided in the table. The atom count is also linearly correlated with the model volume.

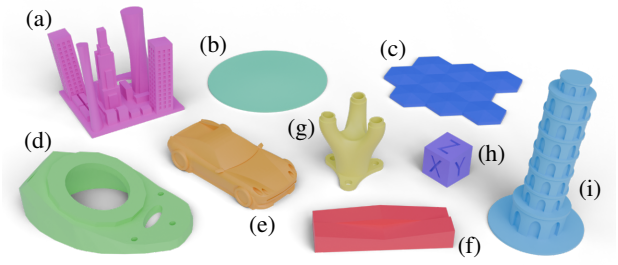


Figure 14: *The models used are: City (a), Dome* (b), Cubic* (c), Ankle (d), Car (e), Slopes (f), Tubes (g), Cube (h), and Pisa Tower (i), with * indicating models with anisotropic appearance.*

The significant memory usage arises from the atomization process, as it relies on volume discretization into voxels. Most of the computational time is spent on the ordering process because we do not explicitly store the precedence constraint graph. Consequently, we must evaluate whether an atom is viable at each iteration. Since our implementation iterates only over the set of non-supporting atoms $\overline{\mathcal{S}}$, the time complexity is primarily determined by the size of this set. For models with larger top surfaces, the computational time is therefore higher.

7.2. Geometric deviation

To validate our approach and compare it to the state of the art, we virtually compute the geometric deviation between the generated toolpath and the input model. The toolpath is represented as anisotropic tubes, where each tube's cross-section is a rounded box with a width and height corresponding to the deposition width τ and height h, respectively. These anisotropic tubes serve as a virtual representation of the toolpath and are shown in Figures 1, 3, 15, 16, 17, and 18.

In Figures 15 and 16, we use these tubes to represent the toolpaths and visualize the deviation of the toolpath from the input surface. More precisely, we depict on the input surface the Euclidean distance between each surface point and the closest point on a toolpath. Conversely, we also depict on the toolpaths the distance to the closest input surface point. We include results generated using two existing non-planar slicers: S^3 [ZFH*22] and CurviSlicer [ERP*19]. We used the reference implementation provided by the authors. Our results consistently show lower geometric deviation compared to those methods. Note that the adaptive slicing feature of the S^3 slicer is enabled for the Slopes and Ankle models. We also attempted to enable it for the Car model, but the algorithm failed to terminate, despite numerous trials with different parameters and tetrahedralizations. Consequently, this model is not shown in Figure 16.

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

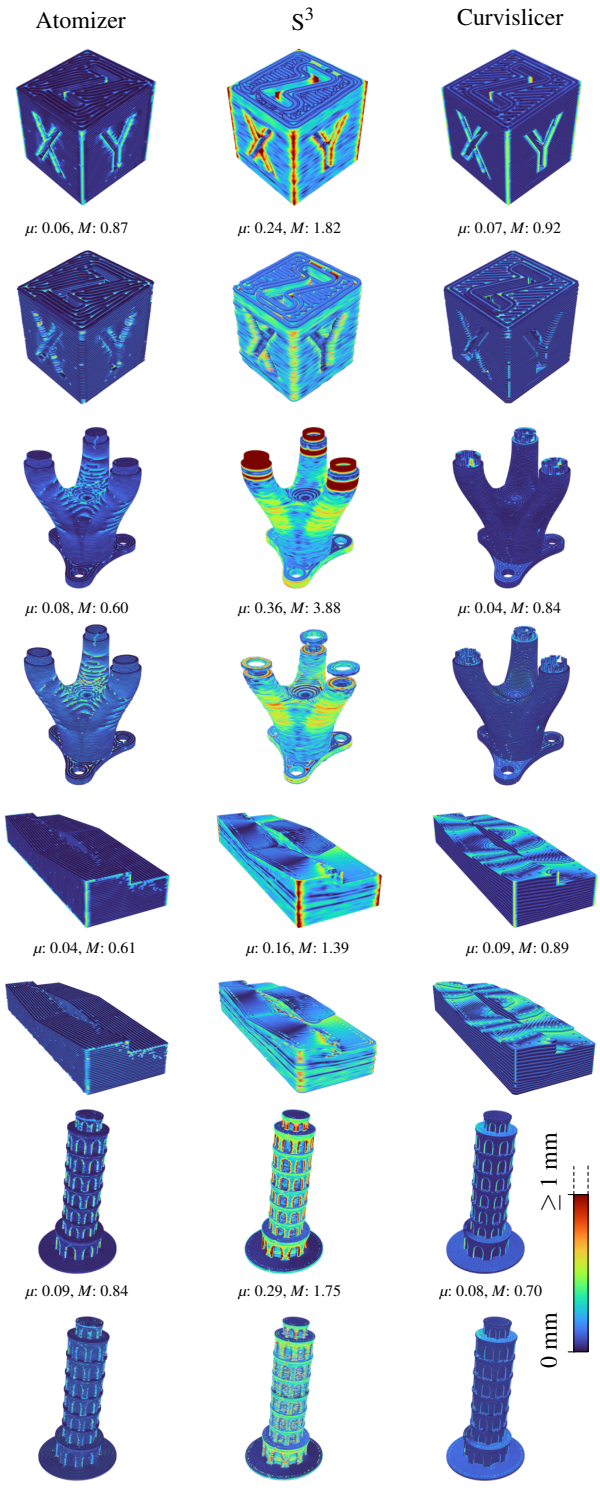


Figure 15: *Geometric deviations. For each model, the top row represents the distance from the input surface to the toolpath, displayed as color attributes on the input mesh. The bottom row represents the distance from the toolpath to the surface. In the first case, the mean (μ) and maximum (M) distances are shown.*

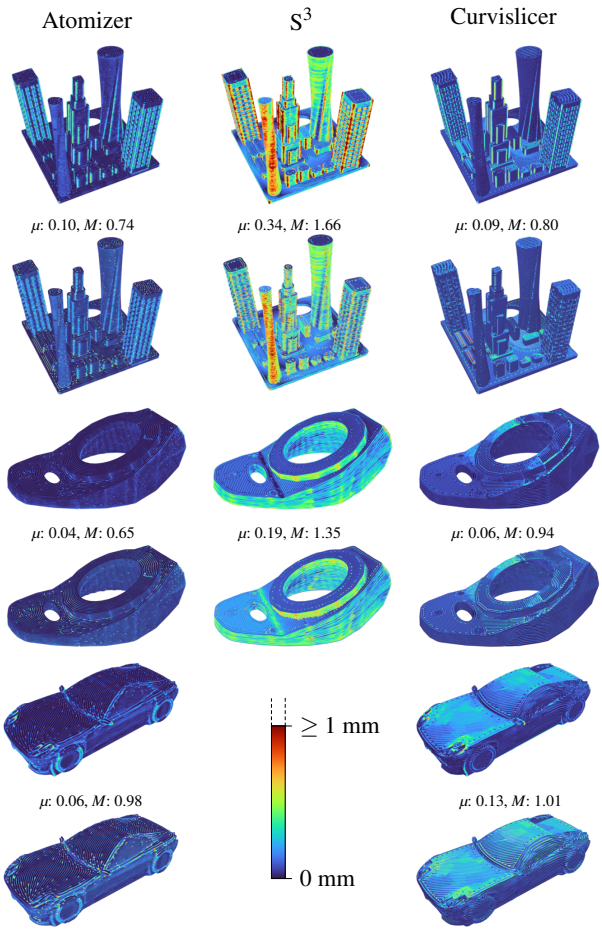


Figure 16: *Continuation of Figure 15.*

Specifically, we observe that the S^3 slicer tends to oversmooth the toolpaths, leading to significant deviations, which appear to be consistent with previously published results. In contrast, CurviSlicer does not exhibit oversmoothing and generally maintains a low geometric deviation. However, both methods can struggle to ensure conformal layers with the top surfaces. This occurs in two scenarios. First, even when the top layer is correctly oriented with respect to the top surfaces, it may still not align correctly with the surface. This issue is especially evident for models such as the Car and Ankle, shown in Figure 16. The apparent reason for this discrepancy in S^3 is that the optimization process used to achieve deformation does not explicitly enforce a distance constraint between the final layer and the object's surface. Rather, it considers only tool orientation. CurviSlicer attempts to perform this alignment, but this is only a heuristic post-process, and it fails on certain models such as the Car. Our method overcomes this limitation by integrating this constraint through precise phase alignment of the cosine plane wave representing the strata (Section 4.3). The other issue is evident in the Slopes model shown in Figure 15. Here, the vertical ridge in the center prevents the deformation-based techniques from achieving a smooth deformation that conforms with the slopes. As a result, a

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

worst-case scenario of staircases occurs, leading to significant distortion.

The Tubes model highlights the performance of each method on thin, slanted walls. These parts are poorly captured by both S^3 and CurviSlicer compared to our method, particularly near the top. These regions exhibit small geometric features of one deposition width ($\tau = 0.9$). Our method fits exactly one atom within such features, and reconstructs a clean toolpath without suffering from aliasing.

An additional issue observed with the reference methods is that the tetrahedralization process struggles to reconstruct fine geometric details. We also found other methods to be less robust and more sensitive to the choice of parameter values. For instance, the S^3 slicer failed to process the car model entirely, and in general, required careful selection of curved surfaces and significant parameter tuning to obtain the best results. Our method, by contrast, processes these models without manual intervention.

7.3. Beyond layer-by-layer ordering

By not constraining the toolpath planner to a strict layer-by-layer ordering, we can generate toolpaths that build vertically until access to another part of the model is obstructed. This behavior is illustrated in Figure 17. This approach helps reduce stringing and oozing artifacts, which commonly occur when the tool must move between adjacent vertical components of the model in each layer. To better appreciate our ordering strategy, please refer to the supplemental video, which shows a time-lapse of the fabrication process for the City and Tubes models.

7.4. Curved anisotropic appearance fabrication

We leverage the ability to constrain deposition tangents (Section 4.2) within any voxel to extend planar anisotropic appearance fabrication [CZM*23] to curved surfaces (Figure 18 and supplemental video). Within the top surface regions, defined as $\{i \in \mathcal{V} \mid \mathrm{SDF}(\mathbf{p}_i) \in [-h, 0) \text{ and } \theta_{\beta_i} \in [0, \Theta]\}$, each deposition tangent $\mathbf{t}_i$ is constrained to a direction determined using planar texture projection along the z-axis. The applied texture represents the Yin-Yang symbol and a cubic pattern.

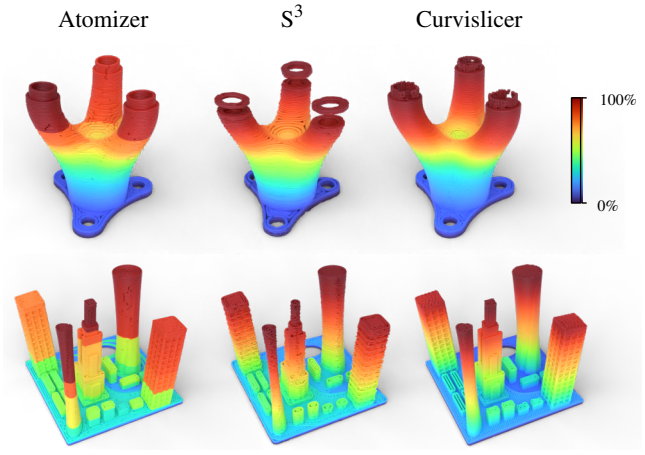


Figure 17: *The toolpaths and their color-mapped lengths. Note that our ordering does not necessarily follow the classic layer-by-layer approach seen in S^3 and CurviSlicer. Instead, each towering feature is built for as long as possible until an accessibility constraint is reached. This automatically stems from our ordering algorithm.*

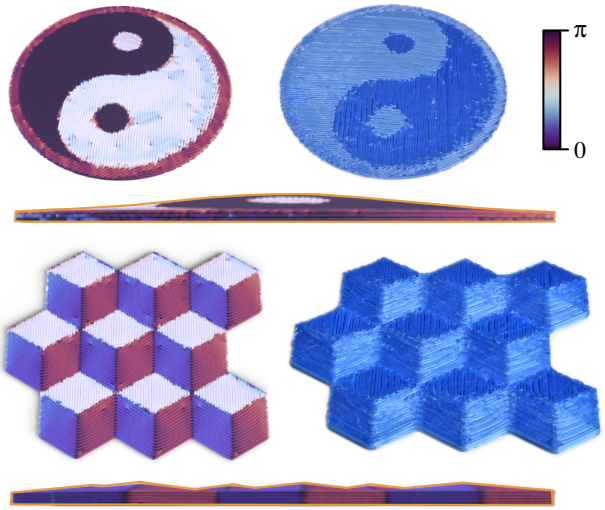


Figure 18: *Direction of the toolpaths (left and side views) and photographs of the anisotropic appearance (right).*

7.5. Statistics of deposition widths and heights

To validate that our toolpath points are correctly spaced, we measure the width and height around all points in all models. We now define the width and height around a toolpath point, $\mathbf{p}'_i$, and more specifically around the deposition trajectories (ignoring travel without material deposition). To achieve this, we define a distance metric that filters out points not in the direction of interest $\hat{\mathbf{d}}$. For the heights, this direction of interest is the tool orientation, $\mathbf{n}'_i$. For the widths, the direction of interest is the bitangent, $\mathbf{b}'_i := \mathbf{n}'_i \times \mathbf{t}'_i$, where $\mathbf{t}'_i$ represents the tangent direction of the toolpath at the point $\mathbf{p}'_i$.

Given two points $\mathbf{p}'_i$ and $\mathbf{p}'_j$, and a direction of interest $\hat{\mathbf{d}}$, the distance is defined as:

$$D(\mathbf{p}'_i, \mathbf{p}'_j, \hat{\mathbf{d}}) := \frac{\|\mathbf{p}'_i - \mathbf{p}'_j\|^2}{\left|(\mathbf{p}'_i - \mathbf{p}'_j) \cdot \hat{\mathbf{d}}\right|} \tag{21}$$

The deposition width and height of a point $\mathbf{p}'_i$ are then defined as: $\min_j \{D(\mathbf{p}'_i, \mathbf{p}'_j, \mathbf{b}'_i)\}$ and $\min_j \{D(\mathbf{p}'_i, \mathbf{p}'_j, \mathbf{n}'_i)\}$, where j ranges over all the points $\mathbf{p}'_j \neq \mathbf{p}'_i$ of the deposition trajectories.

In Figure 19, we compare the distribution of the deposition widths and heights for our method, S^3, and Curvislicer. This shows that our approach produces more consistent spacing. This is desirable, as managing the material flow under variations in deposition width and height is difficult, particularly due to pressure management in the nozzle. Generally, a constant deposition is preferred, as it facilitates control and process calibration.

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

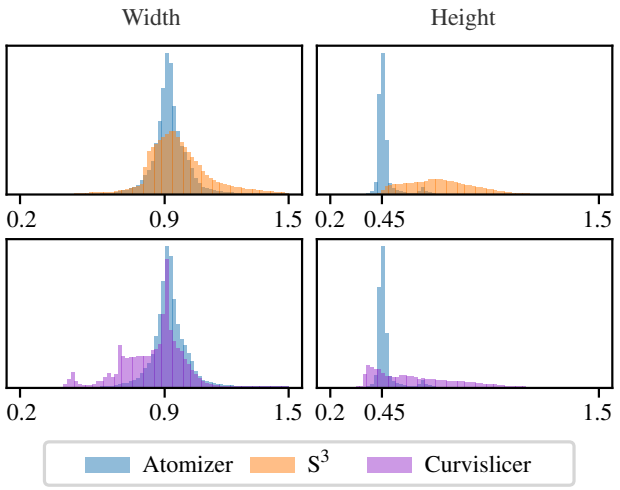


Figure 19: *Comparison of the deposition width and height distributions. The histograms show the aggregate results of the Cube, Slopes, Pisa Tower, City and, Ankle models.*

7.6. Physical experiments

Hardware. We use a commercial 3D printer equipped with three independent Z-axis lead screws: a *RatRig V-Core 3.1* (CoreXY kinematics) with RepRap firmware 3.6.0 on an Octopus Pro board. The extra degrees of freedom provided by the three Z axes were designed to automatically level the build plate. However, we repurpose them as a general bed-tilting mechanism with custom kinematics. The current specification of the printer limits the maximum tool inclination, Θ, to 7°.

Fabricated results. We compare fabrication using our method with fabrication using S^3 and CurviSlicer, based on the code available on the respective GitHub pages of these projects at the time of writing. All models are fabricated on the same printer with the same filament, using identical fabrication parameters: a deposition width of 0.9 mm, a deposition height of 0.45 mm, and a printing speed of approximately 20 mm/s during deposition. The bed never tilts when printing parts sliced with CurviSlicer, as this approach is a 3-axis-only method. The photographs of the fabricated parts are shown in Figure 20.

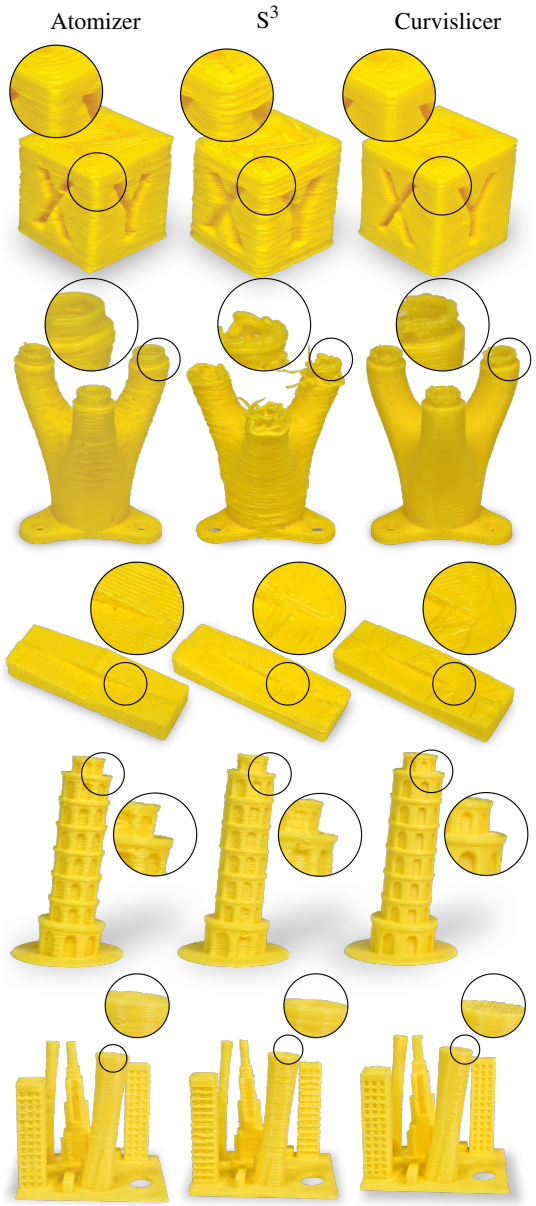


Figure 20: *Photographs of the fabricated parts.*

In general, it can be seen that our approach produces the most accurate and curved results, with only some artifacts along vertical sides compared to CurviSlicer – which never tilts the build plate. In the following, we discuss each print.

The Cube model shows that our technique produces an accurate, slanted top surface, with sides of good quality, only surpassed by CurviSlicer. The reason the sides of our prints are less regular is primarily explained by the rotating bed motions, which are currently slightly less precise on our printer due to uncalibrated kinematics, compared to the straightforward vertical motions in CurviSlicer. The model produced by S^3 suffers from excessive smoothing and uneven deposition.

The Tubes model shows that our technique is capable of curving and capturing the thin-walled tubes at the top, while other slicers fail. Our quality is overall superior to S^3, while CurviSlicer again shows better results on the sides. The main defects on the vertical sides of our print are due to strata that are not orthogonal to the wall and to the connectivity between strata for toolpath alignment. This limitation is discussed in Section 8.

The Slopes model is only properly captured by our technique, with both S^3 and CurviSlicer failing where the opposing slopes meet: significant geometric distortion and staircase defects appear (see also the teaser and Figure 2).

The Pisa Tower model shows how our approach properly repro-

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

duces curved features throughout the model, capturing the slope of individual floors, while CurviSlicer only curves the rooftop.

Finally, the City model demonstrates multiple combined effects: curved landscape and rooftops, detailed sides, as well as vertical features that can be printed one at a time (please also refer to the accompanying video). While CurviSlicer again achieves good print quality on the sides, it curves fewer features in comparison.

In conclusion, our technique produces accurate parts with curved features throughout, without loss of detail and with excellent print quality. The uneven sides are mostly explained by the more complex kinematics, which are not as optimized and calibrated as standard z-motions.

8. Limitations and future work

Connectivity between strata. One limitation of our approach is the connectivity between strata when aligning the cosine plane waves along the tangents and bitangents ($\mathcal{C}_0$ and $\mathcal{C}_1$). Aligning atoms between strata is overly restrictive and can lead to surface quality issues, particularly when the strata are not orthogonal to the wall, as observed in the Tubes model in Figure 15. In future work, we plan to improve surface quality by removing this connectivity and generating atoms that are not necessarily aligned between strata.

3-Z axes machine. The machine, with three Z axes, has a maximum bed inclination of 5° when the nozzle is at the level of the bed, due to the small difference between the bed supports and the tops of the axes at the start of the print. As the bed supports move to the middle of the Z axes, the maximum bed inclination increases to approximately 7°; however, this time, the limitation is caused by the length of the rails supporting the bed. In future work, we will explore hardware modifications to increase the maximum achievable bed slopes.

Infill, supports and bridges. Additional features, such as infills, supports, and bridges, have not yet been implemented in the Atomizer. For infills and supports, the input solid could be processed – such as by generating a microstructure inside it – before atomization. We experimented with basic microstructure generation, as shown in Figure 21; however, proper integration of these features requires further development and validation. The method may also be extended to feature sparse infill structures by using spatially varying periods for the cosine field associated with the bitangents. For example, doubling the period within the volume should lead to a 50% infill. For bridges, detecting the regions where deposition should be fully aligned and determining the bridging direction remains a challenge in the non-planar setting.

Strength reinforcement. We did not exploit control over the deposition direction for any purpose other than achieving anisotropic appearance. However, a potential advantage of this control – demonstrated in prior work – is strength reinforcement, achieved by aligning deposition trajectories with the directions of the mechanical stress field. Our technique should offer greater flexibility in pursuing this objective.

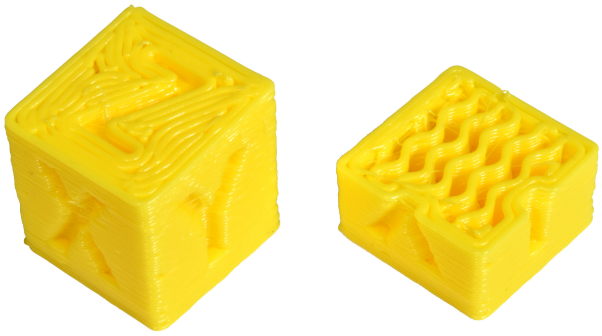


Figure 21: *Gyroid infill generated by processing the SDF prior to the atomization process.*

Toolpath endpoints. The endpoints of toolpaths influence printing quality, as they are often surrounded by under-filled areas, as shown in Figure 13. In our current implementation, the smoothest tangent field is encouraged (Equation 5), which can lead to configurations that are suboptimal for the placement of toolpath endpoints compared to the classical zigzag pattern. In future work, we plan to investigate improved objectives for toolpath tangent alignment. Additionally, toolpath continuity could be enhanced by integrating ideas from the *match twice and stitch* heuristic [KR04, CZM*23] into the proposed method.

Acknowledgments

The authors thank Vincent Belle for the modeling of the Tubes, Pisa Tower, and City models, as well as for his expertise in 3D printing. They also thank Nicolas Ray and Dmitry Sokolov for the inspiring discussions on atom alignment. This work was supported by the French National Research Agency (ANR) under grant numbers ANR-24-CE10-6403 and ANR-22-CE33-0018, and was partially funded by the Inria Challenge DORNELL. The tilted Cube is a remix of the XYZ 20mm Calibration Cube by iDig3Dprinting.

References

[AHL17] ALEXA M., HILDEBRAND K., LEFEBVRE S.: Optimal discrete slicing. *ACM Trans. Graph. (Proc. SIGGRAPH) 36*, 1 (2017), 1 – 16. 2

[AT15] ALLEN R. J., TRASK R. S.: An experimental demonstration of effective Curved Layer Fused Filament Fabrication utilising a parallel deposition robot. *Additive Manufacturing 8* (2015), 78–87. doi:https://doi.org/10.1016/j.addma.2015.09.001. 2

[AWHZ19] AHLERS D., WASSERFALL F., HENDRICH N., ZHANG J.: 3D Printing of Nonplanar Layers for Smooth Surface Generation. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)* (2019), pp. 1737–1743. doi:10.1109/COASE.2019.8843116. 2

[BA05] BAERENTZEN J., AANAES H.: Signed distance computation using the angle weighted pseudonormal. *IEEE Trans. on Visualization and Comput. Graph. 11*, 3 (2005), 243–253. doi:10.1109/TVCG.2005.49. 4

[BCY*21] BAI X., CAO M., YAN W., GE S. S., ZHANG X.: Efficient heuristic algorithms for single-vehicle task planning with precedence

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189

constraints. *IEEE Transactions on Cybernetics 51*, 12 (2021), 6274 – 6283. doi:10.1109/TCYB.2020.2974832. 8, 9

[CARRC08] CHAKRABORTY D., ANEESH REDDY B., ROY CHOUDHURY A.: Extruder path generation for curved layer fused deposition modeling. *Comput. Aided Des. 40*, 2 (2008), 235–243. doi:10.1016/j.cad.2007.10.014. 2

[CZM*23] CHERMAIN X., ZANNI C., MARTÍNEZ J., HUGRON P.-A., LEFEBVRE S.: Orientable Dense Cyclic Infill for Anisotropic Appearance Fabrication. *ACM Trans. Graph. (Proc. SIGGRAPH) 42*, 4 (2023). doi:10.1145/3592412. 6, 14, 16

[DM94] DOLENC A., MÄKELÄ I.: Slicing procedures for layered manufacturing techniques. *Comput. Aided Des. 26*, 2 (1994), 119–126. 2

[DWW*18] DAI C., WANG C. C. L., WU C., LEFEBVRE S., FANG G., LIU Y.-J.: Support-free volume printing by multi-axis motion. *ACM Trans. Graph. (Proc. SIGGRAPH) 37*, 4 (2018). doi:10.1145/3197517.3201342. 3

[DZF*23] DUTTA N., ZHANG T., FANG G., YIGIT I. E., WANG C. C. L.: Vector Field-Based Volume Peeling for Multi-Axis Machining. *Journal of Computing and Information Science in Engineering 24*, 5 (2023), 051001. doi:10.1115/1.4063861. 5

[EFE18] EZAIR B., FUHRMANN S., ELBER G.: Volumetric covering print-paths for additive manufacturing of 3D models. *Computer-Aided Design 100* (2018), 1 – 13. 3

[ERP*19] ETIENNE J., RAY N., PANOZZO D., HORNUS S., WANG C. C., MARTÍNEZ J., MCMAINS S., ALEXA M., WYVILL B., LEFEBVRE S.: CurviSlicer: Slightly curved slicing for 3-axis printers. *ACM Trans. Graph. (Proc. SIGGRAPH) 38*, 4 (2019). doi:10.1145/3306346.3323022. 2, 7, 12

[FZZ*20] FANG G., ZHANG T., ZHONG S., CHEN X., ZHONG Z., WANG C. C. L.: Reinforced FDM: Multi-Axis Filament Alignment with Controlled Anisotropic Strength. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 39*, 6 (2020). doi:10.1145/3414685.3417834. 2, 5

[GJTP17] GAO X., JAKOB W., TARINI M., PANOZZO D.: Robust Hex-Dominant Mesh Generation Using Field-Guided Polyhedral Agglomeration. *ACM Trans. Graph. (Proc. SIGGRAPH) 36*, 4 (2017). 5

[GZN*15] GAO W., ZHANG Y., NAZZETTA D. C., RAMANI K., CIPRA R. J.: RevoMaker: Enabling Multi-directional and Functionally-embedded 3D printing using a Rotational Cuboidal Platform. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (2015), pp. 437–446. doi:10.1145/2807442.2807476. 3

[HLA*19] HU Y., LI T.-M., ANDERSON L., RAGAN-KELLEY J., DURAND F.: Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 38*, 6 (2019). doi:10.1145/3355089.3356506. 11

[JHS*23] JOURDAN D., HUGRON P.-A., SCHRECK C., MARTÍNEZ J., LEFEBVRE S.: Shrink & Morph: 3D-Printed Self-Shaping Shells Actuated by a Shape Memory Effect. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 42*, 6 (2023). doi:10.1145/3618386. 6

[JTPSH15] JAKOB W., TARINI M., PANOZZO D., SORKINE-HORNUNG O.: Instant Field-Aligned Meshes. *ACM Trans. Graph. (Proc. SIGGRAPH) 34*, 6 (2015). 5

[KD96] KULKARNI P., DUTTA D.: An accurate slicing procedure for layered manufacturing. *Comput. Aided Des. 28*, 9 (1996), 683 – 697. 2

[KR04] KAHNG A. B., REDA S.: Match twice and stitch: a new TSP tour construction heuristic. *Operations Research Letters 32*, 6 (2004), 499–509. doi:https://doi.org/10.1016/j.orl.2004.04.001. 16

[KRBCS22] KAPLAN D., RORBERG S., BEN CHEN M., STERMAN Y.: NozMod: Nozzle Modification for Efficient FDM 3D Printing. In *Symposium on Computational Fabrication* (2022). doi:10.1145/3559400.3561999. 2, 3

[LNAK*23] LIU J., NAEEM M. A., AL KOUZBARY M., AL KOUZBARY H., SHASMIN H. N., ARIFIN N., ABD RAZAK N. A., ABU OSMAN N. A.: Effect of Infill Parameters on the Compressive Strength of 3D-Printed Nylon-Based Material. *Polymers 15*, 2 (2023). doi:10.3390/polym15020255. 2

[LZC*24] LIU T., ZHANG T., CHEN Y., HUANG Y., WANG C. C. L.: Neural Slicer for Multi-Axis 3D Printing. *ACM Trans. Graph. (Proc. SIGGRAPH) 43*, 4 (2024). doi:10.1145/3658212. 2

[RMFÖ17] RYBACHUK M., MAUGER C. A., FIEDLER T., ÖCHSNER A.: Anisotropic mechanical properties of fused deposition modeled parts fabricated by using acrylonitrile butadiene styrene polymer. *Journal of Polymer Engineering 37*, 7 (2017), 699–706. doi:doi:10.1515/polyeng-2016-0263. 2

[RSL77] ROSENKRANTZ D. J., STEARNS R. E., LEWIS II P. M.: An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing 6*, 3 (1977), 563–581. 9

[SHB96] SABOURIN E., HOUSER S. A., BØHN J. H.: Adaptive slicing using stepwise uniform refinement. *Rapid Prototyp J. 2*, 4 (1996), 20–26. 2

[SRSL16] SONG H.-C., RAY N., SOKOLOV D., LEFEBVRE S.: Anti-aliasing for fused filament deposition. *Computer-Aided Design 89* (2016), 25 – 34. doi:10.1016/j.cad.2017.04.001. 2, 3

[TEZL21] TRICARD T., ETIENNE J., ZANNI C., LEFEBVRE S.: A brick in the wall: Staggered orientable infills for additive manufacturing. In *Symposium on Computational Fabrication* (2021). 2, 3, 6

[TTZ*20] TRICARD T., TAVERNIER V., ZANNI C., MARTÍNEZ J., HUGRON P.-A., NEYRET F., LEFEBVRE S.: Freely Orientable Microstructures for Designing Deformable 3D Prints. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 39*, 6 (2020). doi:10.1145/3414685.3417790. 6

[UKY*15] ULU E., KORKMAZ E., YAY K., BURAK OZDOGANLAR O., BURAK KARA L.: Enhancing the Structural Performance of Additively Manufactured Objects Through Build Orientation Optimization. *Journal of Mechanical Design 137*, 11 (10 2015), 111410. doi:10.1115/1.4030998. 2

[US13] UMETANI N., SCHMIDT R.: Cross-sectional Structural Analysis for 3D Printing Optimization. In *SIGGRAPH Asia 2013 Technical Briefs* (2013), pp. 5:1–5:4. 2

[WDF*17] WU C., DAI C., FANG G., LIU Y.-J., WANG C. C.: RoboFDM: A robotic system for support-free fabrication using FDM. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), pp. 1175–1180. doi:10.1109/ICRA.2017.7989140. 3

[WWZW16] WU J., WANG C. C., ZHANG X., WESTERMANN R.: Self-supporting rhombic infill structures for additive manufacturing. *Computer-Aided Design 80* (2016), 32–42. doi:https://doi.org/10.1016/j.cad.2016.07.006. 2

[ZFH*22] ZHANG T., FANG G., HUANG Y., DUTTA N., LEFEBVRE S., KILIC Z. M., WANG C. C. L.: S3-Slicer: A General Slicing Framework for Multi-Axis 3D Printing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 41*, 6 (2022). doi:10.1145/3550454.3555516. 2, 3, 7, 12

[ZLD*25] ZHANG T., LIU T., DUTTA N., CHEN Y., SU R., ZHANG Z., WANG W., WANG C. C.: Toolpath generation for high density spatial fiber printing guided by principal stresses. *Composites Part B: Engineering 295* (2025), 112154. doi:https://doi.org/10.1016/j.compositesb.2025.112154. 5, 6

[ZXZL23] ZHONG F., XU Y., ZHAO H., LU L.: As-Continuous-As-Possible Extrusion-Based Fabrication of Surface Models. *ACM Trans. Graph. (Proc. SIGGRAPH)* (2023). doi:10.1145/3575859. 2, 3

This is the submitted version of the article, later accepted for publication in Computer Graphics Forum, published by Wiley and Eurographics. The final version is available at https://doi.org/10.1111/cgf.70189