A Framework for Adaptive Width Control of Dense Contour-Parallel Toolpaths in Fused Deposition Modeling

Kuipers, Tim; Doubrovski, Eugeni L.; Wu, Jun; Wang, Charlie C.L.

DOI
10.1016/j.cad.2020.102907

Publication date
2020

Document Version
Final published version

Published in
CAD Computer Aided Design

Citation (APA)
Kuipers, T., Doubrovski, E. L., Wu, J., & Wang, C. C. L. (2020). A Framework for Adaptive Width Control of Dense Contour-Parallel Toolpaths in Fused Deposition Modeling. *CAD Computer Aided Design*, *128*, 1-15. Article 102907. https://doi.org/10.1016/j.cad.2020.102907

Computer-Aided Design 128 (2020) 102907

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

A Framework for Adaptive Width Control of Dense Contour-Parallel Toolpaths in Fused Deposition Modeling

Tim Kuipers [a,b], Eugeni L. Doubrovski [b], Jun Wu [b,*], Charlie C.L. Wang [c]

[a] Ultimaker, Utrecht, The Netherlands
[b] Department of Sustainable Design Engineering, Delft University of Technology, The Netherlands
[c] Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong Special Administrative Region of China

ARTICLE INFO

Article history:
Received 31 August 2019
Received in revised form 3 February 2020
Accepted 13 June 2020

Keywords:
Adaptive extrusion width
Toolpath generation
Additive manufacturing
Geometrical accuracy
Medial axis transform

ABSTRACT

3D printing techniques such as Fused Deposition Modeling (FDM) have enabled the fabrication of complex geometry quickly and cheaply. Objects are produced by filling (a portion of) the 2D polygons of consecutive layers with contour-parallel extrusion toolpaths. Uniform width toolpaths consisting of inward offsets from the outline polygons produce over- and underfill regions in the center of the shape, which are especially detrimental to the mechanical performance of thin parts. In order to fill shapes with arbitrary diameter densely the toolpaths require adaptive width. Existing approaches for generating toolpaths with adaptive width result in a large variation in widths, which for some hardware systems is difficult to realize accurately. In this paper we present a framework which supports multiple schemes to generate toolpaths with adaptive width, by employing a function to decide the number of beads and their widths. Furthermore, we propose a novel scheme which reduces extreme bead widths, while limiting the number of altered toolpaths. We statistically validate the effectiveness of our framework and this novel scheme on a data set of representative 3D models, and physically validate it by developing a technique, called *back pressure compensation*, for off-the-shelf FDM systems to effectively realize adaptive width.

1. Introduction

3D printing enables the fabrication of complex geometry under few design constraints compared to conventional fabrication techniques. Recent developments have seen a rapid growth in both the use and capabilities of desktop 3D printing systems. Fused Deposition Modeling (FDM) is one of the most common 3D printing techniques. It is widely used because of the versatility in the types of plastic which can be used and the relatively low running costs. FDM printers are used, for example, in showcasing scale models of buildings, casings for electronics, prototypes for blow molded parts, jigs and fixtures. Recent research addressed manufacturing complex volumetric structures such as microstructures [1–3] and topology optimized structures [4–6]. Many of these applications involve 3D models with detailed features within the order of magnitude of the nozzle size, which restrains the field of the process planning algorithms.

FDM printers extrude semi-continuous beads of molten plastic through a nozzle, which moves along a planned toolpath within each layer of a 3D object. A common strategy is to extrude along a number of parallel toolpaths which follow the shape of the contour of the layer and fill up the remaining area using parallel straight toolpaths. Contour-parallel toolpaths fit to the layer outlines more accurately, because the resolution of the positioning system is an order of magnitude smaller than the size of the hole in the nozzle. This paper is concerned with the generation of such contour-parallel toolpaths and addresses several issues which commonly occur in 3D models with narrow geometry.

The simple technique for generating the dense contour-parallel toolpaths of a layer consists of performing uniform inward offsets with the size of the nozzle from the outline shape. However, for geometrical features which are not an exact multiple of the nozzle size this method produces areas where an extrusion bead is placed twice: *overfill* areas; and areas which are not filled at all: *underfill* areas. See Fig. 1(a). Overfills cause a buildup of pressure in the mechanical extrusion system, which can result in bulges or even a full print failure. Underfills, on the other hand, can cause a drastic decrease in the part stiffness or even for small features not to be printed at all. These problems are exacerbated for models with layer outlines with small features, because the over- and underfill areas are relatively large compared to the those features.

One promising direction to avoid over- and underfills is to employ toolpaths with adaptive width. Ding et al. developed a toolpath strategy for wire and arc additive manufacturing which produces a width variation typically lower than a factor of 3,

* Corresponding author.
E-mail address: j.wu-1@tudelft.nl (J. Wu).

https://doi.org/10.1016/j.cad.2020.102907

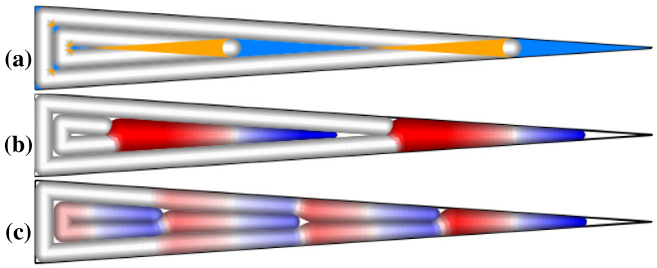


Fig. 1. Illustration of different toolpaths for a shape showcasing a range of shape radii (black). These results can be read as a graph with feature size on the horizontal axis and its corresponding beading along the vertical axis. (a) Toolpaths using uniform offsetting results in large overfill (orange) and underfill (azure). (b) Toolpaths with adaptive width [9] where beads that are wider or narrower than the nozzle size are indicated in red and blue, respectively. (c) Our approach minimizes over- and underfill with less extreme widths. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

but is far greater for some parts [7,8]. However, the range of bead widths manufacturable by FDM systems is limited. A nozzle of $w = 0.4$ mm will typically start to cause fluttered extrusion around lines narrower than 0.3 mm and lines will start to bulge upward if they are wider than the flat part of the nozzle, which is typically 1.0 mm.

The current state of the art of contour-parallel toolpath generation developed by Jin et al. employs a strategy which alters the widths of the centermost beads within a range of widths $[0.25w, 1.8w]$ [9], which is similar to the strategy employed by the open source industry standard software package Ultimaker Cura [10]. See Fig. 1(b). Still, controlling the extrusion width through movement speed changes or through volumetric flow control (e.g. linear advance) yields diminishing accuracy for deposition widths deviating more from the nozzle size, since process parameters such as nozzle temperature are optimized for beads with the nozzle size. Moreover, reducing the variation in width is beneficial for limiting the variation in mechanical properties of the resulting product, meaning it conforms better to a simulation which employs a homogeneity assumption. We therefore reduce the bead width range by distributing the workload from the centermost bead over neighboring beads.

Our contributions are as follows:

- A geometric framework allowing various adaptive bead width control schemes used to generate contour-parallel toolpaths which minimize under- and overfill.
- A specific beading scheme, which reduces the variation in the extrusion widths to within $[0.75w, 1.5w]$.
- A back pressure compensation approach to accurately realize adaptive bead width on Bowden style hardware systems.

2. Related work

Toolpath generation consists in generating a path in a planar contour, representing the intersection of a plane and a 3D solid object. The nozzle is then instructed to move along the path while extruding material. Sites along the toolpaths are assigned several properties such as movement speed, but for this paper we will focus on the assigned width of the extruded bead. Toolpath generation is an integral part of process planning for 3D printing. For an overview of the processing pipeline, we refer to the survey by Livesu et al. [11]. For reducing printing time and material cost, sparse infill structures such as triangular and hexagonal patterns have been used to approximate the interior of 3D shapes. In this paper, we focus on the generation of toolpaths for dense regions, such as the boundary shell of 3D shapes. This is sometimes called 'dense infill' [11].

The toolpath has a direct influence on the printing time, material cost, and mechanical properties of the printed object [12,13]. FDM calls for toolpaths with several desirable properties. First, the extrusion path should be as continuous as possible. A discontinuous path requires to stop and restart material extrusion. For certain materials, such as TPU, this could lead to printing defects or even print failure [14]. Second, the toolpath is preferred to be smooth. Sharp turns require to reduce the movement speed of the nozzle, and so this prolongs the printing process. Third, the extruded path should cover the region of the contour without underfilling. Such underfill negatively influences the mechanical performance of the parts. Fourth, the extrusion paths should not overlap with one another. Such overfill causes a pressure build up in the mechanical system, which leads to overextrusion in later paths and in extreme cases cause print failure [14]. An analysis of under- and overfill from a vertical cross-section was presented in [15]. Our method is primarily concerned with minimizing under- and overfill within horizontal cross-sections.

Two basic strategies for dense toolpath generation are the direction-parallel strategy and the contour-parallel strategy. Direction-parallel (or zig-zag) toolpaths fill an arbitrarily shaped contour with a set of parallel, equally spaced line segments. These parallel segments are linked together at one of their extremities, to avoid discontinuous extrusion. Contour-parallel toolpaths typically consists of a set of equally spaced offsets from the slice boundary outline polygons. Steuben et al. presented a method for generating sparse infill toolpaths based on the isocontours of surface plots of some variable generated on each 2D contour [16]. In order to increase the continuity of contour-parallel toolpaths, a strategy to connect dense toolpaths into spirals was introduced by Zhao et al. [17] and later extended to also connect a mixture of dense and sparse toolpaths together [14]. Jin et al. discuss several approaches for connecting direction-parallel and contour-parallel toolpaths into continuous paths [18]. Spiral toolpaths have also been applied to (CNC) machining [19,20]. One of the problems with contour-parallel toolpaths is that it tends to leave gaps between the toolpaths (see Fig. 1(a)). This is due to the fact that the diameter of the part is not exact multiple of the (constant) deposition width in those regions. To avoid problems with such gaps, hybrid approaches that combine direction and contour-parallel are often used [21,22]. Close to the slice boundary, there are several contour-parallel curves, while the interior is filled using a zig-zag pattern. For complex shapes, the entire cross-section could be decomposed into a set of patches, and for each of them the basic strategies can be applied [18,23]. Alternative toolpath patterns, seen also in CNC machining, include space-filling curves [24–26].

Reducing under- and over-filling can be accurately achieved by making use of adaptive deposition width. Adaptive width can be used to locally match the nonuniform space between adjacent paths, and thus to ensure a better filling of the area. Kao and Prinz propose smooth adaptive toolpaths based on the medial axis skeleton of the boundary contour [27]. Their approach handles simple geometry where there are no branches in the medial axis. An extension was proposed by Ding et al. to handle complex shapes [7]. However, this extension inherits a problem in the original method: from any point in the skeleton to the boundary, the number of toolpaths is constant. Depending on the size of small and large features in the layer outlines, this strategy can require a range of bead widths beyond the capabilities of the manufacturing system. Jin et al. proposed a strategy of adding toolpaths with varying width along the center edges of the skeleton, while leaving other paths unchanged [9]. The resulting beads have widths within the wide range of $[0.25w, 1.8w]$ (see

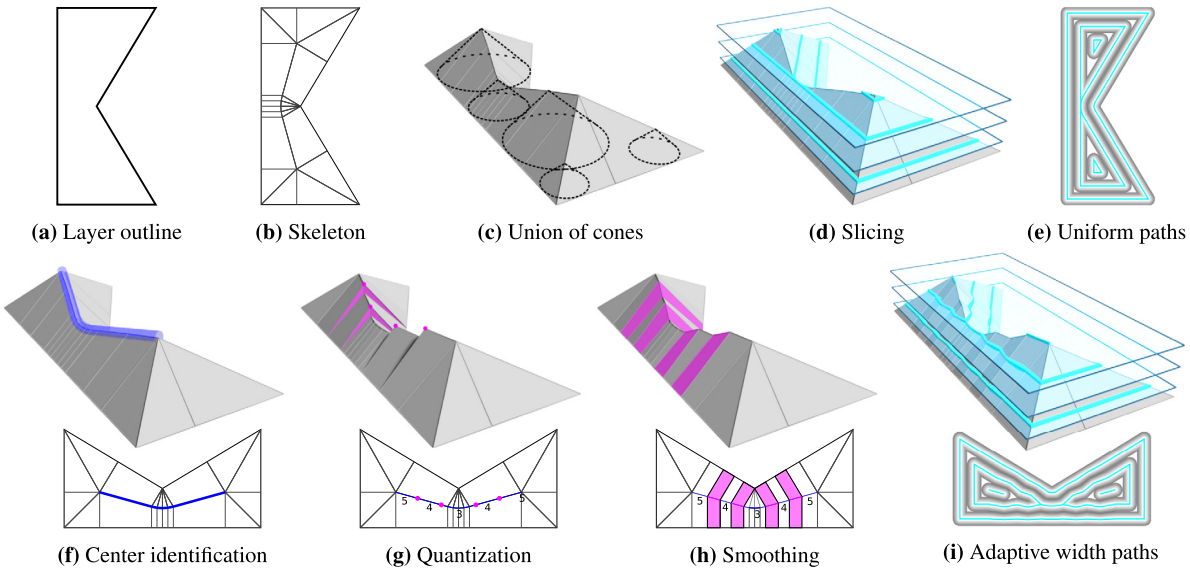


Fig. 2. The first row illustrates the generation of uniform paths (e) by interpreting the path as the intersection between horizontal planes and the union of cones (c), which is an alternative visualization of the skeleton (b). The second row depicts the stages with both 2D and 3D visualizations for generating paths with adaptive width (i). Central elements in the skeleton are first identified (blue in (f)). The heights are then quantized in terms of number of beads (the integer values in (g)), and smoothed (h).

Fig. 1(b)). In this paper we propose a novel scheme to distribute the width alterations throughout a region around the center, and thus limit the occurrence of extreme variation in width (see Fig. 1(c)).

Under- and over-filling issues have a large proportional impact for thin geometric features. Jin et al. proposed a sparse wavy path pattern for thin-walled parts [28]. Besides under- and over-filling, there are a few other robustness issues in toolpath generation for thin geometric features. Moesen et al. proposed a method to reliably manufacture thin geometric features using laser-based additive manufacturing techniques [29]. Behandish et al. presented a method to characterize local- topological discrepancies due to material under- and over-deposition, and used this information to modify the as-manufactured outcomes [30].

For ease of reference we have included a legend showing the terms employed in this manuscript in Fig. 4. These terms will be further explained as they first appear throughout this paper.

3. Method

3.1. Overview

Given arbitrarily shaped polygons which represent the 2D outline of a layer of a 3D model, our method generates extrusion toolpaths with varying width, i.e. a set of polylines where each site consists of a location and an extrusion width; in between the sites we linearly interpolate the position and extrusion width.

Our method starts with computing a graph which represents the topology of the input polygon: its *skeleton*. Our skeleton is based on the medial axis transform (MAT), a strategy that has been commonly used for generating contour-parallel toolpaths [31]. We visualize the skeleton as the union of cones (UoC) (Section 3.2), by raising each point in the domain to a height that equals the shortest distance from the point to the polygon boundary (Fig. 2(c)). Contour-parallel toolpaths with uniform width can be interpreted as the intersection of the union of cones with a set of horizontal planes at equally spaced heights (Figs. 2(a)–2(e)).

As depicted in Fig. 2(f), our method first identifies edges and nodes of the skeleton in the center of the polygon, which correspond to ridges and peaks in the UoC (Section 3.3). The heights $\tilde{b}$ at these elements are then quantized to an integer number of beads $\bar{b}$. To ensure a smooth toolpath between regions with quantized integer heights that differ, we add new nodes in the skeleton with quantized heights and interpolate the heights $\hat{b}$ in between (Section 3.4). The union of cones corresponding to the smoothed skeleton is then sliced at regular intervals to obtain toolpaths with varying spacing, which translates into varying width (Section 3.6). The video in the supplementary material provides an example animation of this approach.

This section explains how we generate toolpaths using our framework with uniform bead widths and evenly distributed locations between the center of the polygon and the outline. In Section 4 we describe how to apply the framework to different beading schemes and we show several such beading schemes.

3.2. Union of cones

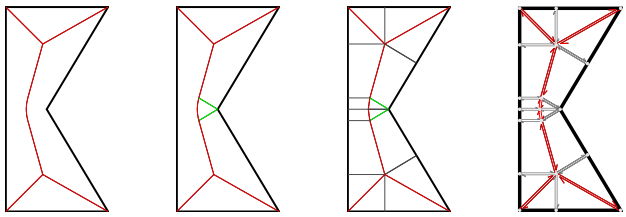
The union of cones (UoC) is derived from a common skeletonization of the polygonal outline shape: the medial axis. By assigning each node in the skeleton a height equal to its shortest distance to the outline we obtain the shape of the UoC. Starting from the medial axis we further decompose the shape into simple fragments, so that the domain contains only quads and triangles. This decomposition constitutes an approximation of the UoC.

Medial axis transform. The medial axis is a representation commonly used to analyze a shape. It is defined as the set of positions where the inscribed circle meets the boundary in at least two locations [32,33]. The resulting skeleton consists of straight edges and parabolic edges. An example is illustrated in Fig. 3(a). We call the set of points on the outline polygon P closest to a skeletal point v its *support*:

$$\sup(v) = \arg\min_{x \in P} |x - v|. \tag{1}$$

The shortest distance for a point on the skeleton is called its feature radius, $R(v)$. The medial axis along with the feature radius values along the skeleton form a complete shape descriptor, known as the medial axis transform (MAT).

By vertically raising the center of an inscribed circle to a height that equals the center's feature radius, a cone is formed. The



(a) Medial Axis **(b)** Voronoi Diagram **(c)** Skeletal Trapezoidation **(d)** Data structure

Fig. 3. Skeletonization of an outline shape (black). Relation between the medial axis (red), the limited Voronoi Diagram (red and green) and the Skeletal Trapezoidation (red, green and gray): MAT ⊂ Limited VD ⊂ ST. (d) The skeleton is represented using a half-edge data-structure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

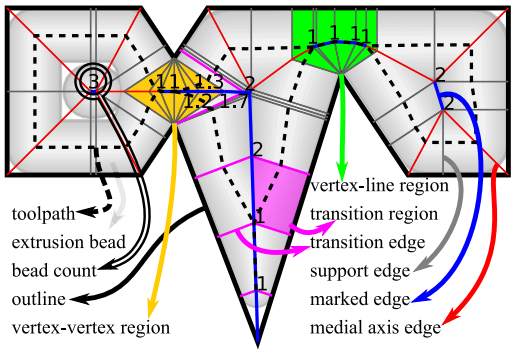


Fig. 4. Illustrative explanation of terms and color coding that are consistently used in this paper.

union of all such cones forms a 3D solid volume. The medial axis can thus also be interpreted as ribs of the surface of the union of cones [32].

Skeletal trapezoidation. Starting from the medial axis we decompose the input polygon into a set of quads and triangles, so that we can perform the slicing stage on simple shapes. We employ a shape decomposition similar to the one proposed by Ding et al. [7]. The basic idea is to add edges connecting each node v on the medial axis to each of its support points $\sup(v)$. The resulting skeleton decomposes the outline shape into trapezoids and triangles. Considering the fact that the concept of trapezoidation conventionally allows for the degenerate case where a trapezoid resolves into a triangle [34,35], we call this shape decomposition the *Skeletal Trapezoidation* (ST).

The edges generated by the MAT are classified into three types: 1. line–line edge — straight edge generated from two line segments in the outline polygons, 2. vertex–line edge — parabolic edge resulting from an outline vertex and a line segment in the outline, and 3. vertex–vertex edge — straight edge resulting from two outline vertices. The vertex–line and vertex–vertex edges are discretized into pieces with a length up to 0.2 mm, which gives an approximation error of only ±0.01 mm. This allows to approximate the feature radius between two discretized nodes v_0 and v_1 by linear interpolation. Again we connect the newly inserted nodes to their support, which results in vertex–line regions and vertex–vertex regions such as depicted in Fig. 4.

Approximation of union of cones. The skeletal trapezoidation (ST) provides a means to visualize the union of cones (UoC) approximated by a 3D surface mesh composed of quadrilateral and triangular patches. We assign each node in ST a (real number) height value measured in terms of beads, referred to as the *bead count* b. We define the bead count as the number of beads to fit along the *diameter* of the inscribed circle centered at node v, i.e. $2R(v)$, by

$$\tilde{b}_v = 2R(v)/w^* \tag{2}$$

where w^* is the nozzle size. We divide the diameter rather than the radius as this allows to deal with an odd number of beads while using integer logic. Note that although the overview of the method was described geometrically in terms of the UoC, the actual toolpath generation relies on the two-dimensional ST; the use of the bead count as a height value is only a visualization aid.

Implementation. The medial axis of a polygonal shape is a subset of the Voronoi Diagram generated from the line segments and vertices of the shape [33]. The edges of the Voronoi diagram that fall outside of the outline shape are irrelevant for our purpose and are thus discarded. Note that besides the full medial axis, the Voronoi diagram also contains edges connecting to concave vertices in the outline shape (see Fig. 3(b)). These extra edges are a subset of the edges connecting a node to its support, so we keep them in. From the Voronoi diagram we add nodes to discretize parabolic edges and edges formed by two concave outline vertices, and then connect all nodes to their supports, forming a skeletal trapezoidation. We then assign each node the bead count values using Eq. (2). We compute the Voronoi diagram using the Boost C++ libraries [36], which implements the algorithm proposed by Fortune [37]. A half-edge data-structure is used to represent the Voronoi diagram (Fig. 3(d)).

3.3. Center classification

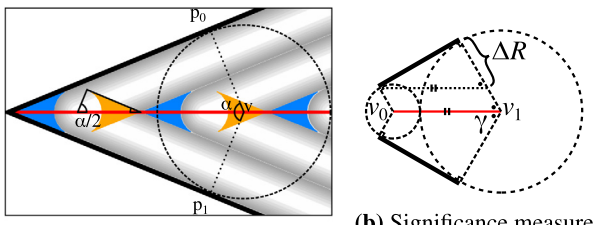
In order to prevent over- and underfill from occurring in central regions, parts of the ST are marked as being central. Our framework will decide on a beading at all the marked nodes in 'the center' and apply the beading outward to the unmarked nodes (Section 3.5).

A node in the ST is marked as as central if its feature radius is larger than that of all its neighboring nodes, i.e. a local maxima. An edge and its two nodes are also marked as being central if it is significant according to a significance measure.

Significance measure. We make use of the *bisector angle* as an indicator of significance which is commonly used in shape analysis. The bisector angle α is the interior angle $\angle p_0 l p_1 \leq 180°$, between any location l on an edge of the ST and its two supporting points $\{p_0, p_1\} = \sup(l)$ [38]. An edge is significant if the bisector angle on any location on the edge exceeds a prescribed $\alpha_{\max}$. As illustrated in Fig. 5(a), for a polygon with a pointy wedge area of an angle β, we have $\alpha = 180° - \beta$. This corresponds to overfill areas and underfill areas the size of $\frac{1}{4}(w^*)^2(\tan(\alpha/2) - \alpha/2)$ when filled using the simple technique of uniform bead width w^*. A too large $\alpha_{\max}$ may leave a lot of under-/overfill, while a too small value may introduce toolpaths to fill in negligibly small underfills. We therefore set $\alpha_{\max} = 135°$. Although significance measures are commonly used as a heuristic for finding the parts of a skeleton which are in some sense relevant [38,39], we use the bisector angle as an *exact* indicator of the amount of overfill and underfill in the uniform toolpaths of constant width.

To avoid evaluating the bisector angle at any location on all edges, we devise an efficient measure which operates only on the two nodes of an edge. Because all locations along a line–line edge have the same bisector angle we can evaluate whether the edge is significant by checking whether

$$|R(v_1) - R(v_0)|/|v_1 - v_0| > \cos(\alpha_{\max}/2) \tag{3}$$



(a) Over- and underfill **(b)** Significance measure

Fig. 5. Properties of the significance measure along a skeletal edge (red) generated from two polygon lines (black) using the properties of inscribed circles (gray) and their radii (dashed). (a) The size of overfill (orange) and underfill areas (azure) for the uniform toolpathing technique can be calculated from the bisector angle. (b) The significance measure can be simplified using $\alpha = 2\gamma = 2\cos^{-1} \Delta R/|v_1 - v_0|$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(see Fig. 5(b)). This ratio has a clear geometrical interpretation as the slope of the ridge in the UoC surface. For vertex–line edges and vertex–vertex edges only a portion of the edge is significant. We therefore introduce nodes at the boundaries of the significant portion during the discretization of such edges (see Appendix A). The significance of all edges can then accurately be evaluated using Eq. (3).

Marking filtering. After initializing the marking at all edges and nodes, we filter out high frequency changes in the marking in order to ensure that the generated toolpath is smooth. The filtering is performed by additionally marking some unmarked elements, rather than the opposite since unmarking central regions reintroduces large over- and underfill areas. From each marked node v_0 with an upward unmarked edge attached we walk along the upward edges; if the total length traversed until we reach another marked node v_1 is shorter than some filter distance $d_{\max}^{\text{unmarked}}$, we mark all edges encountered as being central. We use $d_{\max}^{\text{unmarked}} = w^*$ in order to filter out high frequency oscillations in the order of magnitude of the nozzle size, while keeping close to the significance measure.

3.4. Central height adjustment

After the central regions have been identified, their heights are quantized. First, the initial bead count $\tilde{b}$ is quantized into an integer bead count $\bar{b}$ at the marked nodes using a quantization operator q, then the locations along the edges where q makes a jump from one bead count n to another $n+1$ are identified and then ramps are introduced to smoothly transition from n to $n+1$ using fractional bead counts $\hat{b}$ along the smooth transition.

Quantization. We define a quantization operator q to map a feature diameter ($d = 2R(v)$) to a bead count: $q : \mathbb{R} \to \mathbb{N}$. Because our quantization scheme should round to the nearest integer multiple of the nozzle size, we have $q(d) = \lfloor d/w^* + 1/2 \rfloor$. Alternative quantization schemes are discussed in Section 4. By applying q to the heights of central nodes we quantize the bead count:

$$\bar{b}_v = q(2R(v)) = \left\lfloor \tilde{b}_v + 1/2 \right\rfloor \tag{4}$$

Transition anchors. For a marked edge which connects nodes v_0 and v_1 with $\bar{b}_{v_0} \le n < \bar{b}_{v_1}$, we determine the *transition anchor locations* at which the bead count transitions from n to $n+1$. To this end, we introduce the function

$$q^{-1}(n) := \arg\max_d q(d) = n, \tag{5}$$

which gives the feature diameter d at which the bead count q transitions from n to $n+1$. The location of the anchor v_x is then computed by inversely interpolating $R(v_x) = q^{-1}(n)$, i.e.

$$v_x = v_0 + (v_1 - v_0)\frac{q^{-1}(n) - R(v_0)}{R(v_1) - R(v_0)}. \tag{6}$$

An illustration of the anchors is shown in Fig. 6(b).

We perform a filtering step to prevent frequently changing the bead count back and forth within a short distance. For two consecutive anchors which transition to opposite directions, if the distance between them is smaller than some limit $d_{\max}^{\text{transition}}$, the bead counts at all nodes in-between are set to the surrounding bead counts, and consequently these anchors are removed (see Fig. 6(c)). A value of $d_{\max}^{\text{transition}} = 1\,\text{mm}$ seems to produce satisfactory results. This means that for some small regions we generate toolpaths with bead widths outside the typical range.

Smooth transitions. A sharp transition from n to $n+1$ beads at an anchor location creates sharp turns in the toolpath (see Fig. 7 top). We introduce a transition length $t(n)$ to ensure a smooth transition (see Fig. 7). The length of the transition is set to $t(n) = w^*$ and it is centered at the anchor, i.e. the distance from the lower end v_0 to the anchor position v_x is set to $t_0(n) \equiv \Delta(v_0 v_x) = t(n)\left(q^{-1}(n)/w^* - n\right)$, where Δ is the total distance along the edges between two nodes. The transition length $t(n)$ ensures that the center beads do not overlap with the innermost transitioning beads, while keeping the amount of underfill low and the toolpath smooth. The transition anchor position $t_0(n)$ ensures that the transitions never overlap with each other or with locations where all beads have the preferred width w^*.

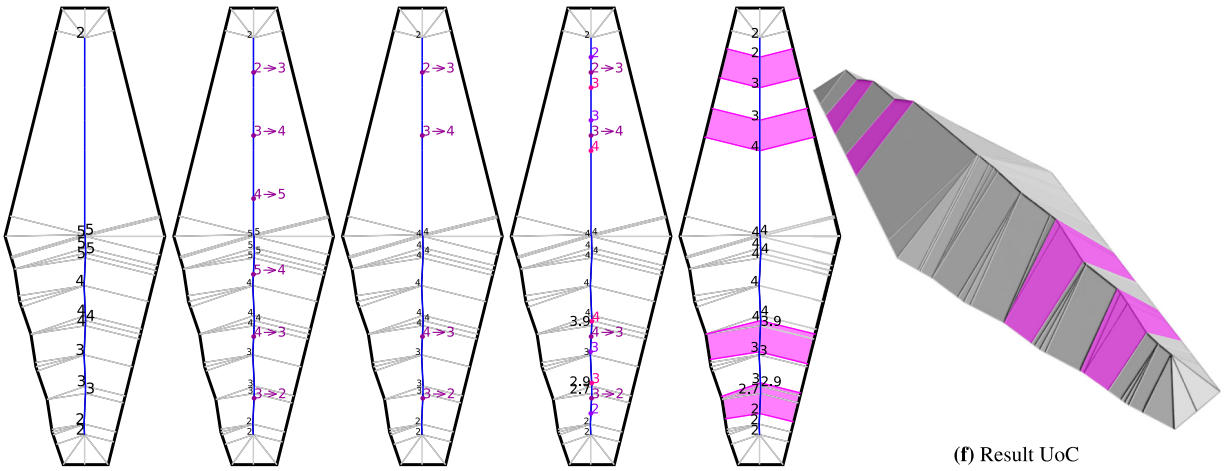
We discard any transition anchor which is too close to the end of a chain of marked edges for the smoothed transition to fully fit within the marked region. In order to make the transition ramps robust against small perturbations in the outline shape which cause extra (support) edges in the skeleton, we modify the nodes v_x which are between the two ends v_0 and v_1 of the transition by (re-)assigning them a fractional bead count $\hat{b}$ which is linearly interpolated between the two ends of the transition (see Fig. 6(e)):

$$\hat{b}_{v_x} = n + \Delta(v_0 v_x)/\Delta(v_0 v_1) \tag{7}$$

Note that although the ST is not stable against noise in the boundary shape, the distance field itself is, so by designing our algorithms such that they are stable against changes in the topology of the skeleton our method is stable against small perturbations in the outline. Finally we update the ST by adding support edges at the transition ends. As shown in Fig. 6(f), the marked regions in the UoC mesh have become horizontal at integer multiples of $1/2 w^*$ for long stretches with ramps in between.

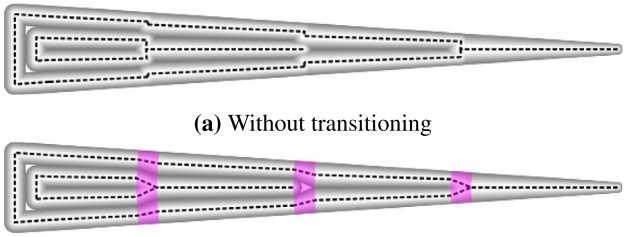
3.5. Beading

Now that we know how to determine the bead counts in the marked central regions the question is how the unmarked regions are handled. Determining bead count values for the unmarked nodes and interpolating linearly along the unmarked edges would mean that toolpath sites would be distributed evenly along each unmarked bone; while that would suffice for the evenly distributed beading scheme, it would not allow for more sophisticated, non-linear schemes. Instead we determine the radial distance to the boundary at which each bead should occur from the boundary to the center. Each central node is associated with a sequence of radial distances L which control the locations of the beads, starting from the outer bead and ending in the center. Together with a sequence of bead widths W, these form what we call a *beading* B. For our distributed beading scheme we



(a) Quantized bead counts (b) Transition anchors (c) Filtered anchors (d) Transition ramp ends (e) Radial support edges (f) Result UoC

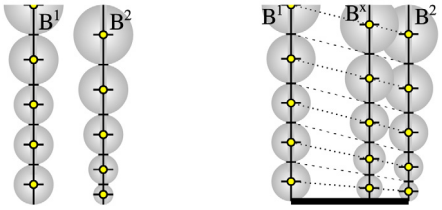
Fig. 6. Applying bead counts and transitioning on a shape showing the difference between a simple ST (top) and a mirrored version with small perturbations in the outline (bottom). Outline in black, central edges marked in blue, radial edges in gray. (a) First we initialize the bead counts (black) in the marked edges (blue). (b) We then extract the anchor locations (purple) where the bead count transitions. (c) We then filter out regions which exhibit frequent transition. (d) We then calculate the end locations (magenta, pink) of the transitions and modify the bead count at nodes in between to fractional values. (e) Finally we introduce nodes at the ends and introduce radial edges (purple) as per the trapezoidation constraint. The symmetry in the result shows that transitioning is robust against small perturbations in the outline shape. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Without transitioning

(b) Transitioning

Fig. 7. Sharp turns around regions where the bead count changes are prevented by transition regions (highlighted in cyan). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Example beadings (b) Interpolation

Fig. 8. Interpolation between two beadings B^1 and B^2 with odd and even bead count resulting in a beading B^x at $n + 2/3$. Bead indices are counted inward from the outline (thick black). Interpolation of locations in dots, interpolation of widths in dashes.

compute the beading for a central node v with $n = \lfloor \hat{b}_v \rfloor$ beads and a diameter $r = R(v)$ as:

$$
\begin{aligned}
B(n, r) = (W(n, r), L(n, r)) \quad &= \left(\{w_0 \dots w_{\lceil n/2 \rceil - 1}\}, \{l_0 \dots l_{\lceil n/2 \rceil - 1}\}\right) \\
w_i = r/n \quad &\text{for all } i \in \mathbb{N} : i < n/2 \\
l_i = r/n(i + 1/2) \quad &\text{for all } i \in \mathbb{N} : i < n/2
\end{aligned}
$$

where w_i and l_i are the width and location of the ith bead, respectively, counting from the outline inward. Example beadings for an odd and even bead count with arbitrary widths are visualized in Fig. 8(a).

Beading interpolation. The beading is defined in terms of an integer number of beads, while we have assigned a fractional bead count to nodes within a transition region. In order to generate a beading for a node v with $n < b_v^* < n + 1$ we linearly interpolate the bead widths and locations between a beading B^1 based on n and a beading B^2 based on $n + 1$ (see Fig. 8). Such interpolation is also used to deal with beading conflicts (see Fig. 9). There we also apply beading interpolation from a marked node v_m upward along unmarked bones, and interpolate between v_m and the beading at the top of the slope over some distance t_{beading} from the lower marked node, which we set to $t_{\text{beading}} = w^*$, so that the transition is not too swift.

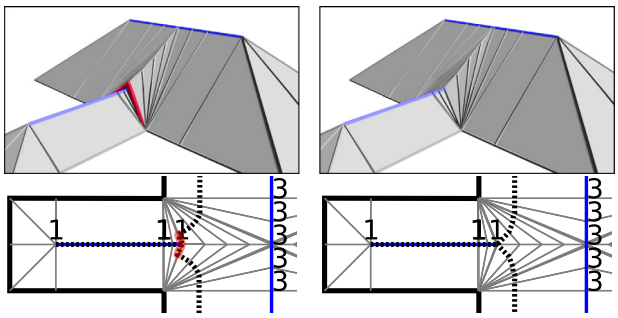
Beading propagation. The beading information is then broadcast throughout the ST from central regions outward, so that each unmarked node v knows the beading of the marked node on top of the ramp on which v is placed. We first broadcast the beading information upward from all marked nodes, so that we can then deal with beading conflicts in a downward phase. The downward phase makes sure that all nodes have a beading associated with it, so that the slicing algorithm can efficiently slice the edges leading up to a marked or unmarked node.

3.6. Toolpath extraction

Once each node has been assigned a beading, we proceed to generate the toolpath sites along the edges of the ST. A site S consists of a location v a width w and an index i, which are computed for an edge v_0v_1 from the beading B of the upper node v_1:

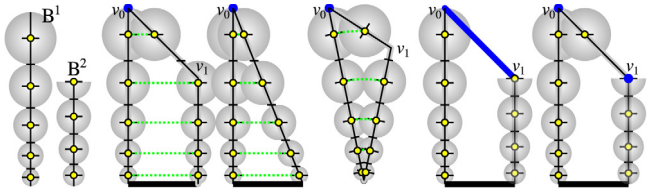
$$
\begin{aligned}
S &= \{v, w, i\} \\
v &= v_1 + (v_0 - v_1)\frac{R(v_1) - l_i^B}{R(v_1) - R(v_0)} \\
w &= w_i^B
\end{aligned}
$$

for any i for which $R(v_0) < l_i^B \leq R(v_1)$. See Fig. 10. We store all sites of an edge in a mapping from edge to a list of sites.



(a) Beading conflict **(b)** Conflict resolution

Fig. 9. (a) The beading propagated from above conflicts with the beading below. (b) The beading conflict is resolved by gradually interpolating between the two beadings. The ramp to the upper ridge does not line up with the lower ridge, which means that the toolpaths (dashed) resulting from the beading propagated from above does not align with the beading from the thin outline feature (highlighted in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



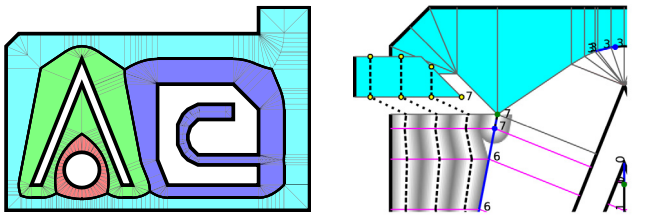
(a) Beadings **(b)** Single beading propagated to all nodes **(c)** Separate beadings at either top node

Fig. 10. Applying beadings to generate sites along trapezoids. (a) shows the locations l_i and widths w_i of two arbitrary different beadings. (b) shows the application of B^1 to the various types of trapezoid. (c) shows how a trapezoid with a marked edge will have two different beadings assigned, which will generate their respective sites along the support edges. No sites will be generated along marked edges. Wide black lines are outline segments, marked nodes and edges in blue, the sites in yellow and green wavefronts of equidistant radial distance at $R = l_i$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We then generate extrusion segments for each trapezoid by connecting together the sites of the same index. See Fig. 11. If the amount of sites on both sides of the trapezoid is not the same then this trapezoid is in a transition and we leave one inner site unconnected.

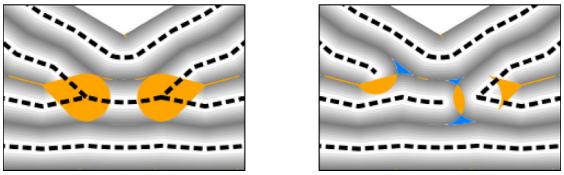
Because the bead count is defined in terms of the feature diameter rather than the radius, only some of the bead count values $\hat{b}$ in a central region coincide with a slicing height. When the bead count $\hat{b}$ is even, the ridge is sliced as normal; the intersection between a slicing plane and the mesh surface results in a polyline on both sides of the ridge, which are connected together into a polygonal toolpath. When the bead count $\hat{b}$ is odd, the ridge will coincide exactly with a slicing height, which results in a single polyline toolpath being generated along the middle of the feature. In that case we should prevent the algorithm from generating the center extrusion segment twice from the trapezoids on either side of that segment. We therefore use some arbitrary condition to decide which one of the two to include based on the ordering of the coordinates of v_0 and v_1: $x_0 < x_1 \vee (x_0 = x_1 \wedge y_0 < y_1)$.

All trapezoids in the ST are assigned to separate domains, corresponding to which boundary polygon they are connected to (see Fig. 11(a)) [7]. By traversing the trapezoids per domain in order we can efficiently connect all segments into polylines. See Fig. 11. In a final step we connect the ends of polylines together, so that the final toolpaths contain both polygons and polylines.



(a) Polygon domains **(b)** Extrusion segment chaining

Fig. 11. Generating toolpaths on a part of the test outline shape by chaining together extrusion segments along each polygon domain. Each edge is assigned toolpath sites (yellow) which are connected together as shown in the singled out trapezoid. By following the trapezoids along the domain (cyan) of a single outline polygon, the extrusion segments can efficiently be connected into existing polyline toolpaths (light and dark gray). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) No reduction **(b)** Reduction

Fig. 12. Reducing polyline toolpaths away from intersections in order to prevent overfill. Toolpath locations in black, underfill in azure and overfill in orange. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Around the transition locations and around nodes with odd bead count and more than two marked edges attached there will be intersections in the toolpaths. Such intersections cause overfill because the nozzle passes the location multiple times. We deal with this special case by forcing a new polyline when traversing the trapezoids, and in the final polyline connection step we greedily connect the first two polylines ending in the same location and retreat all other polylines ending in that same location in order to prevent the overfill. In order to retreat a polyline which ends in a site S, we remove part of the polyline paths up to the intersection by a distance of $w^S d_{\max}^{\text{intersection}}$. We set $d_{\max}^{\text{intersection}} = 75\,\%$ in order to slightly favor overfilling over underfilling. This ratio effectively deals with the balance between overfill and underfill generated at that location after the retreat has been applied. See Fig. 12.

4. Beading schemes

A critical component in toolpath generation is how to distribute the beads over the feature radius. While the framework presented in the previous section takes evenly distributed beads as an example, it allows to apply different beading schemes to configure the bead distribution to cater for specific requirements from the application, 3D printer or material.

Definition 4.1. A beading scheme is defined by the quantization operator q and the beading operator B: $\{q(d), B(n, r)\}$. The beading function $B(n, r)$ consists of $(W(n, r), L(n, r))$, which provides sequences of n bead widths and of n distances from the outline to fill up a radial distance r.

For the smoothness and continuity of toolpaths we require that W_n is monotonic and continuous at each bead index n for constant bead count c: $0 \leq \frac{\partial W(c,r)_n}{\partial r} \leq 1$. We further ensure that

Table 1
Beading schemes.

(a) Uniform scheme

$$q^-(d) = 2\left\lfloor \frac{d}{2w^*} + \frac{1}{2} \right\rfloor$$

$$W(n,r)_i = w^* \text{ for all } i$$

(b) Outer bead

$$q(d) = \begin{cases} 1 & \text{if } d < w^* \\ 2 & \text{otherwise} \end{cases}$$

$$W(n,r)_i = \begin{cases} 2r & \text{if } n = 1 \\ w^* & \text{otherwise} \end{cases}$$

(c) Constant bead count

$$q(d) = C$$

$$W(n,r)_i = 2r/n \text{ for all } i$$

(d) Evenly distributed

$$q(d) = \left\lfloor \frac{d}{w^*} + \frac{1}{2} \right\rfloor$$

$$W(n,r)_i = 2r/n \text{ for all } i$$

(e) Centered

$$q(d) = q^-(d) + \begin{cases} -1 & \text{if } q^-(d)w^* - d > w^* - d_{\max} \\ 1 & \text{if } q^-(d)w^* - d < w^* - d_{\min} \\ 0 & \text{otherwise} \end{cases}$$

$$W(n,r)_i = \begin{cases} 2r - (n-1)w^* & \text{if } i = \frac{1}{2}(n-1) \\ w^* & \text{otherwise} \end{cases}$$

(f) Inward distributed

$$q(d) = \left\lfloor \frac{d}{w^*} + \frac{1}{2} \right\rfloor$$

$$W(n,r)_i = w^* + E(n,r)\frac{\omega(n,r)_i}{\sum_{j=0}^{n-1}\omega(n,r)_j} \text{ for all } i$$

$$E(n,r) = 2r - nw^*$$

$$\omega(n,r)_i = \max(0, 1 - N^{-2}(i-(n-1)/2)^2)$$

beads do not overlap, that beads are extruded from the center of where they end up and that odd bead counts produce a single polyline toolpath exactly in the center by determining the bead locations from the widths:

$$L(n,r)_i = \begin{cases} -\frac{1}{2}W(n,r)_i + \sum_{j=0}^{i} W(n,r)_j & \text{if } i < \frac{1}{2}(n-1) \\ r & \text{if } i = \frac{1}{2}(n-1) \end{cases}$$

We introduce several beading schemes which determine the bead count and their widths in various ways. We can emulate a variety of toolpath generation methods from related literature by defining new beading schemes. We also introduce new beading schemes which produce toolpaths with less extreme widths compared to techniques from existing literature.

Uniform beading scheme. We can define a beading scheme which emulates the uniform width offsetting technique by disabling the marking of edges, so that we never employ transitioning. We can simply set $\alpha_{\max} = 180°$ and supply a simple beading scheme given by Table 1a.

Outer bead. We can emulate the method from Moesen et al. by carefully choosing how the beading scheme functions deal with the outermost bead. Also we turn off the reduction of toolpaths near 3-way intersections $d_{\max}^{\text{intersection}} = 0\,\%$, so that the polygonal toolpaths emulate the remaining area to be filled by another path planning technique similar to their technique. We do not need transitioning, so we also set $t(n) = 0$. See Table 1b.

Constant bead count. We can emulate the method from Ding et al. by dividing the feature radius over the widths of a constant number of beads. Additionally in order to emulate their definition of "branches" we mark all ST edges ($\alpha_{\max} = 0°$) and we unmark the outer edges connected to the outline shape in a separate algorithm. Note that this deviation from the proposed framework violates the robustness against small perturbations in the outline polygon, since this marking depends on the topology of the graph of the ST. See Table 1c.

Centered. We can emulate the method from Jin et al. by transcribing how they deviate from the uniform width toolpaths. We therefore base the beading scheme on the bead count $q^-(d)$ defined by the uniform beading scheme. Jin et al. replace two beads from the uniform toolpaths by a single one when the distance between the center of those beads falls short of $d_{\min} = 0.8w^*$, which gives us $w_{\max} = d_{\min} + 2 \cdot \frac{1}{2}w^* = 1.8w^*$. Conversely, they place an extra bead when the distance exceeds $d_{\max} = 1.25w^*$ [9], which gives us $w_{\min} = d_{\max} - 2 \cdot \frac{1}{2}w^* = 0.25w^*$ [9, p. 72]. We emulate the rounded polygonal path rerouting they define by supplying a transition length $t(n) = \frac{1}{2}w^*$ which results in a discretized version of their rounded polygon segment. See Table 1e.

Evenly distributed. By taking the advantages of the above two schemes we can define a beading scheme which constitutes a novel toolpathing technique. We can evenly divide the local feature diameter over the widths of all beads, but choose a local bead count better matching the local feature size. We determine the local bead count by dividing the diameter by the preferred bead width and rounding to the nearest integer. This reduces the demands on the system and deviation from mechanical properties caused by beads with extreme deviations from the preferred width. See Table 1d.

Inward distributed scheme. The evenly distributed scheme can be conceptualized as calculating the total discrepancy E between the actual feature diameter d and the total preferred width nw^*, dividing the total discrepancy by the number of beads and setting the width of each bead to $w^* + E/n$. However, depending on the application we might want a different distribution of widths. We therefore supply a beading scheme which supports an arbitrary distribution of the discrepancy. The distribution is determined by some weighing function $\omega(n,r)$, which defines the portion of the discrepancy to distribute to each bead. See Table 1f. For example, we can choose an ω which distributes the discrepancy over the innermost N beads, and distribute most of it to the inner beads. See Fig. 13. That way we limit the region of impact of the distributed scheme to a central region and have the preferred bead width w^* in regions farther away. This limits the impact of transitioning regions so that transitions keep the toolpaths smooth farther away from the central regions.

Widening meta-scheme. Complementary to any of these schemes we can enforce a minimum feature size and minimum bead width in our framework. Regions where the model is narrower than some $r_{\min}$ can be printed with a bead width $w_{\min}$ larger than the model thickness. See Figs. 14(a) and 14(b). We can simply override

$$q'(d) = \begin{cases} 0 & \text{if } 0 \le d < 2r_{\min} \\ 1 & \text{if } 2r_{\min} \le d < w^* \\ q(d) & \text{otherwise} \end{cases}$$

$$W'(n,r)_0 = \begin{cases} \max(w_{\min}, 2r) & \text{if } 2r < w^* \\ W(n,r)_0 & \text{otherwise} \end{cases}$$

Shell meta-scheme. The industry standard of FDM is to generate only a limited contour-parallel perimeters and to fill the remainder using a direction-parallel strategy. We therefore provide a meta-scheme to generate adaptive bead width toolpaths only in narrow regions and generate the limited number of perimeters M using the preferred width in regions which are wide enough. We also take care not to leave gaps which are too small to be filled using the direction-parallel strategy:

$$q'(d) = \min(M, q(d))$$

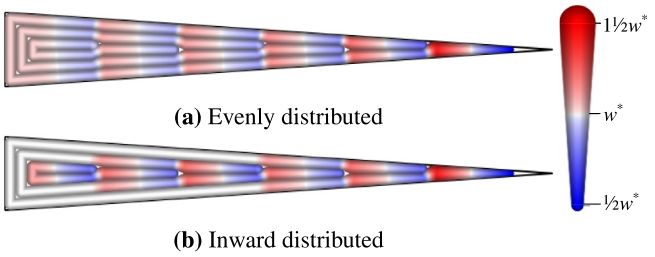


Fig. 13. Closeup of toolpaths generated with the distributed and inward ($N = 1.5$) beading schemes for a large wedge shape. Colors represent bead widths. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

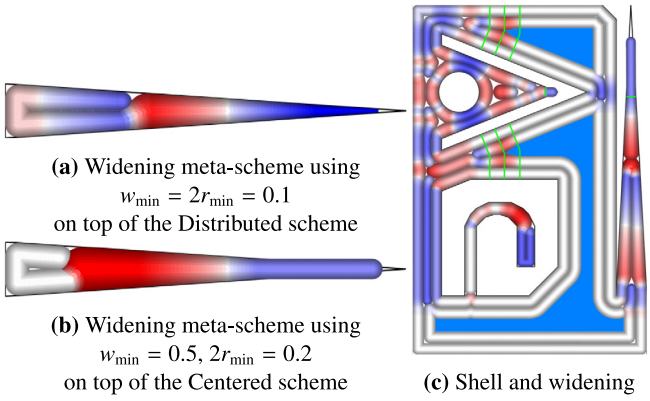


Fig. 14. Toolpaths using the widening and shell meta-schemes. (a) and (b) show widening. (c) show toolpaths generated with the inward distributed strategy ($N = 1.5$) in conjunction with the shell meta-scheme ($M = 4$) and the same widening as in (b). Widening and shell require extra edges (green) at key locations in the skeleton. The azure area is to be filled using some direction-parallel toolpaths. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$W'(n,r)_i = \begin{cases} W(n, Mw^*)_i & \text{if } 2r > q^{-1}(M) \\ W(n,r)_i & \text{otherwise} \end{cases}$$

These meta-schemes introduce non-linearities in the quantization function. Because the beading is only evaluated at nodes in the skeleton, we need to make sure that there are nodes at the locations along the skeleton where the non-linearities happen. We therefore insert extra nodes along with their ribs at locations v with a radial distance $R(v) = r_{min}$ for widening and at $R(v) \in \{Mw^*, q^{-1}(M), q^{-1}(M) + \frac{1}{2}w^*\}$ for the transition from narrow shell to unconstrained shell. Combining all meta-schemes functionality we can generate results such as depicted in Fig. 14(c).

5. Fabrication

In order to accurately manufacture adaptive width toolpaths using an off-the-shelf 3D printing system, we need a model which relates the required width to process parameters such as movement speed and filament extrusion speed. A different approach might be appropriate depending on whether the filament feeder is mounted directly on the print head (a.k.a. *direct drive*) or the filament fed from the back of the printer to the print head via a *Bowden tube*. Because Bowden style 3D printing systems have the filament feeder relatively far away from the nozzle, changing the internal pressure in the system requires a large amount of filament movement, which requires a prohibitive amount of time.

5.1. Back pressure compensation

Because changing the internal pressure is difficult in our setup, we keep the internal pressure constant, and vary the movement speed instead. One approach would be to keep the filament inflow f (in $mm^3\,s^{-1}$) constant by varying movement speed accordingly [40]. However, that does not result in the intended filament outflow variation — see Fig. 15(a). We conjecture that the filament outflow is related to the total pressure in the system, which depends not only on the amount of filament in between the feeder wheel and the nozzle (which we keep constant), but also depends on the back pressure that the previous layer exerts on the filament protruding from the nozzle. The amount of back pressure is most likely monotonically related to the requested line width. We compensate for the back pressure using a simple linear model:

$$v(w) = \frac{f(w)}{hw} \tag{8}$$

$$f(w) = f_0 - k\,(w/w_0 - 1) \tag{9}$$

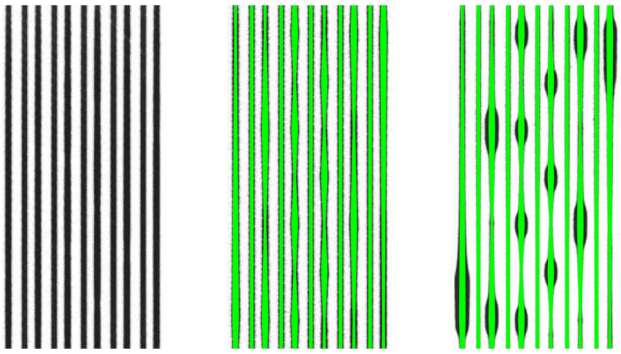
where $v(w)$ is the movement speed as a function of requested bead width w, $f(w)$ is the filament outflow, f_0 is a constant reference flow, w_0 is a constant reference bead width and k is the amount of back pressure compensation.

Our back pressure compensation method effectively changes the speed to realize adaptive width, but this approach is limited, since the movement speed is constrained by acceleration considerations near bends in the toolpath [41]. Moreover, as the layer height is decreased the back pressure becomes larger compared to the internal pressure, which might cause the back pressure compensation method to demand prohibitively slow movement speeds. Furthermore, the shape and filling of the previous layer might influence the amount of back pressure. Accurate flow control can be further enhanced by using a direct drive hardware system and by employing *pressure advance algorithms* which dynamically change the internal pressure [42]. Conversely such a setup might benefit from some form of back pressure compensation as well.

5.2. Print results

Using increments of 0.1 we established that using a factor of $k = 1.1$ yields satisfactory bead width variation for our setup where we use $f_0 = v_0 w_0 h$ with $v_0 = 30\,mm\,s^{-1}$, $w_0 = 0.4\,mm$ and $h = 0.1\,mm$. See Fig. 15(b). The fact that the printed lines are wider than intended is compensated for using a flow reduction to 90 %. Test prints were performed on an unmodified Ultimaker S5 system, with a standard 0.4 mm nozzle and PLA filament. The printing order is determined greedily by choosing the closest point of a polygonal extrusion path, or the closest of either end point in case of an open polyline extrusion path. Because the machine instructions file format *G-code* does not natively support adaptive width beads, we discretize adaptive width extrusions into 0.2 mm long segments of the average width. The print results can be viewed in Fig. 16.

In Fig. 16(b) the underfill problem of the naive uniform offset approach is most prevalent for the Ultimaker word mark, which negatively impacts the visual quality and the stiffness of the part. Moreover, in the case of the spatially graded honeycomb, there are several fully disconnected hexagons, which means the object falls apart when picked up. The honeycomb print is also missing all parts which are slightly more thin than the preferred bead width w^*. Fig. 16(c) still shows some underfill, but considerably less than the uniform approach. These prints also exhibit dark regions where the translucency of the layer is less because the bead



(a) $k = 0$ **(b)** $k = 1.1$ **(c)** $k = 2.0$

Fig. 15. Print results (black) of the varying width test on top of a dense white raft. Target widths in green. (a) Simple flow equalization without back pressure compensation results in nearly constant bead widths. (b) A value of $k = 1.1$ seems to produce good results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is higher. This can be explained by inaccuracies in the back pressure compensation method, which arise for bead widths which deviate from the preferred width by a large amount. Fig. 16(d) diminishes the underfill nearly completely and the visual quality of these prints is more homogeneous than those of the other methods. Moreover, the absence of dark regions signifies that our proposed method is more robust against inaccuracies in the deposition system. However, both the centered and inward distributed approach introduce transitions to a different bead count in the word 'Delft', which reduces the dimensional accuracy on the outline around those locations.

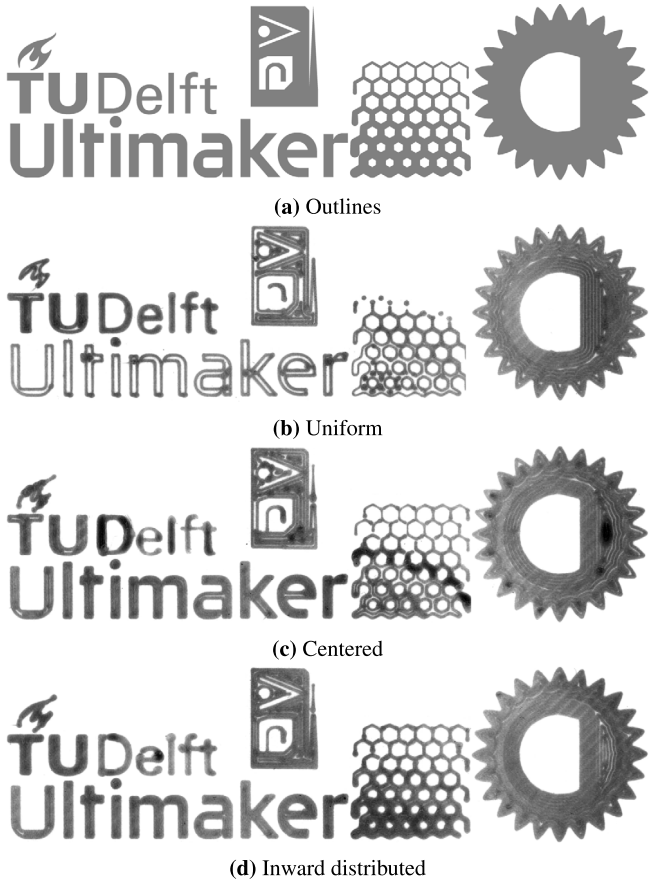
6. Results and discussion

We evaluate the proposed framework and the various beading schemes on a set of different types of 3D models, ranging over various applications and various types of geometry. The data set is described in Appendix B. We sliced all models in the data set and selected 300 random slices for analysis. Toolpaths of these 300 outline shapes are generated using the uniform technique as implemented by Clipper [43] — a state-of-the-art polygon offset library, and by our framework using four beading schemes, i.e. the constant bead count scheme with a bead count of $C = 4$, the centered, the evenly distributed, and the inward distributed beading scheme using $N = 2$, all with a preferred bead width of $w^* = 0.5$ mm and using the widening meta-scheme to enforce a minimum printed feature size of $w_{\min} = 2r_{\min} = 0.3$ mm. The tests were performed on a desktop PC equipped with an Intel Core i7-7500U CPU @ 2.70 GHz (a single core is used) and 16.3 GB memory. We report on the total statistics summed over the whole data set, because averaging would be biased.

6.1. Computational results

6.1.1. Accuracy

We first evaluate the accuracy of different beading schemes in terms of the relative amount of the overfill and underfill. We construct the over- and underfill area by comparing the shapes covered by each extrusion move with each other and with the total shape of the boundary polygons. (For implementation details see Appendix C.) This results in polygonal shapes such as visualized in the top half of Fig. 17: there are orange shapes where the beads overlap and azure shapes in the voids in between the beads. We compare the total area in mm^2 of these overfill and underfill shapes to the total area of the boundary for each sample



(a) Outlines **(b)** Uniform **(c)** Centered **(d)** Inward distributed

Fig. 16. Test shapes printed using the uniform scheme, centered scheme and the inward distributed scheme. The uniform technique produces distinct underfill areas. The centered scheme shows some defects due to inaccurate control of extreme deposition widths. The inward distributed scheme produces the least defects.

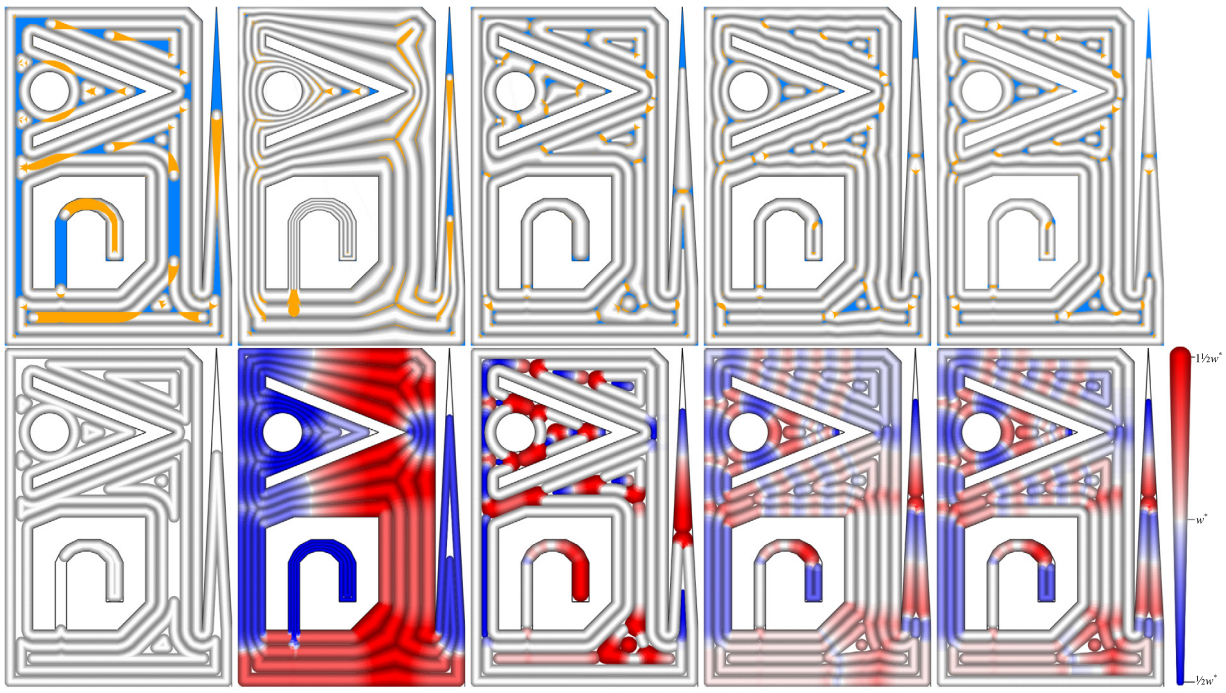
in the data set and report the average percentages in Fig. 18(a). The inward distributed scheme has a calculated overfill of 0.30 % and an underfill of 0.24 %. This is lower compared to the uniform scheme, which results in 1.63 % overfill and 1.62 % underfill in the data set.

6.1.2. Uniformity

We visualize the bead widths resulting from the different schemes in the bottom of Fig. 17. We binned the toolpaths into width bins at 0.01 mm increments and determine the total toolpath length pertaining to each bin. From these statistics we calculate the mean and standard deviation and report them in Fig. 18(d). We found that the mean width of the inward and evenly distributed schemes is close to the preferred bead width of 0.5 mm, while their standard deviation is lower than for the centered and constant bead count scheme. These results show that, while causing less overfill and underfill, inwards distributed and evenly distributed schemes deviate less or less often from the preferred bead width compared to the other schemes.

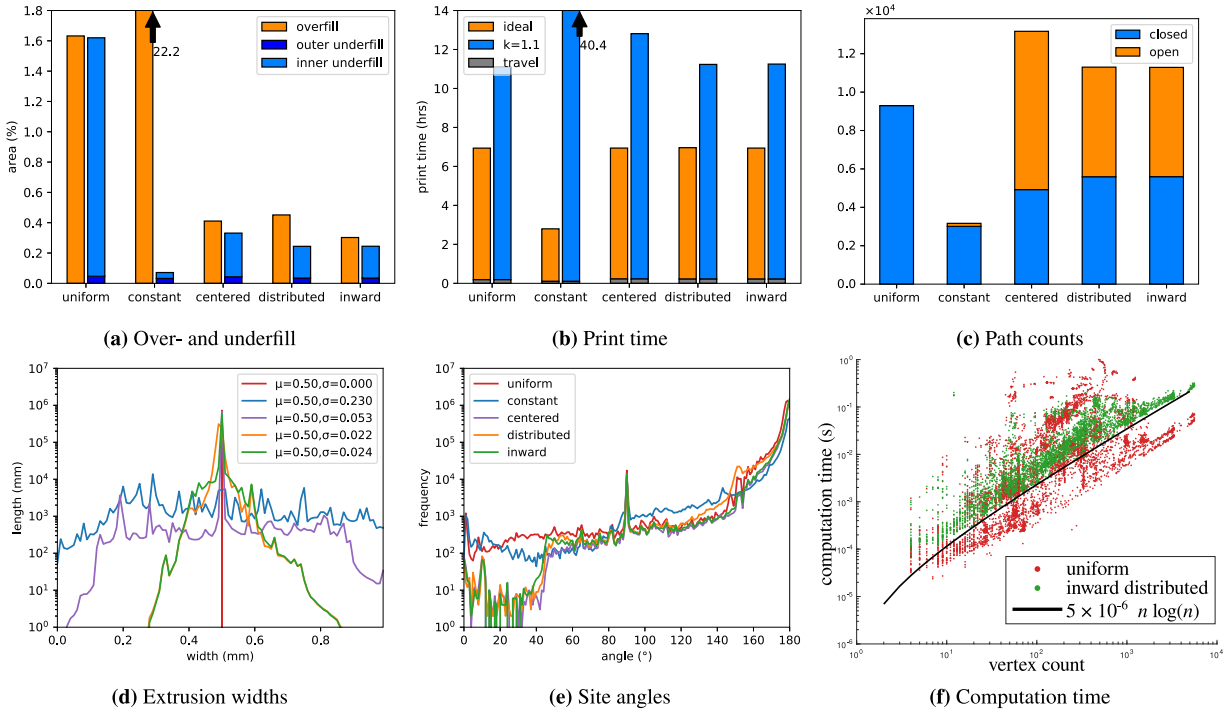
6.1.3. Print time

The total time it takes to print a part is influenced not only by our back pressure compensation scheme, but also by the geometry of the toolpaths. In order to separate these effects we report on the total print time when using back pressure compensation and when using a constant (maximum) movement speed in Fig. 18(b). We estimate print times using a simulation



(a) Uniform (b) Constant (c) Centered (d) Distributed (e) Inward ($N = 2$)

Fig. 17. Visualization of the overfills and underfills (top) and the widths (bottom) for various beading schemes. Extrusion beads in gray tones, overfill in orange, underfill in azure, narrow beads in blue and wide beads in red. In order to distinguish clearly from the Distributed scheme the Inward is limited to $N = 2$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Over- and underfill (b) Print time (c) Path counts

(d) Extrusion widths (e) Site angles (f) Computation time

Fig. 18. Statistical analysis of the toolpaths from applying the uniform width technique and various beading schemes using our framework to a data set of 300 slices. Note the use of a logarithmic scale in the bottom graphs on the Y-axes and for (f) on the X-axes as well.

of the Marlin firmware using the default movement settings of the setup described in Section 5.2. While the idealized print time is predominantly determined by the total toolpath length, the print time using back pressure compensation is predominantly determined by the occurrence of wide beads, because they have a reduced the flow in $mm^3\ s^{-1}$. Because of acceleration constraints imposed by the hardware the maximum movement speed is not reached near sharp corners. We therefore also report on the

angles of the bends in the toolpaths in Fig. 18(e). Furthermore, the print time is negatively affected by discontinuities in the extrusion process. Between extrusions the printer needs to stop extrusion, travel to the next extrusion path and restart the extrusion process, which may introduce defects and incurs extra print time. For closed polygonal toolpaths we can start anywhere within the path, so we can optimize the starting location so as to minimize the travel time. We therefore report both on the open and closed path count in Fig. 18(c).

6.1.4. Computational performance

Fig. 18(f) plots the computation time against the vertex count of the layer for the full data set, comparing the uniform technique implemented using Clipper [43] to our framework with the inward distributed scheme. For polygonal shapes with as many as 10^4 vertices, the computation for both approaches is less than 1 s, with our method being approximately five times that of the uniform technique. These results could be improved upon by utilizing the locality inherent in our algorithms for parallelization on the GPU.

The computational complexity is limited by the generation of the Voronoi Diagram, which is $O(n \log n)$, where n is the number of vertices in the input shape. The other steps in our framework have a complexity of $O(m)$, where m is the number of elements in the ST. Therefore, the total running time of our algorithm is $O(n \log n)$. Results in Fig. 18(f) confirm that both our framework and the uniform technique have an expected running time of approximately $5 \times 10^{-6} n \log n$ seconds.

6.2. Comparison of beading schemes

We can see from Figs. 17(a)(top) and 18(a) that the uniform technique causes a lot of overfills and underfills: on average 1.6% of the total target area is covered by underfill and likewise for overfill. To our knowledge, the uniform beading scheme, as well as the outer beading scheme, is of little use to FDM printers.

The constant bead count scheme effectively deals with underfills, but generates orders of magnitude more overfills compared to the other schemes. Also, the scheme comes at the cost of greatly varying bead widths and an average bead width that is not close to the preferred bead width. Note that most overfill areas occur near regions of alternating bead width. While the scheme results in short toolpaths, as indicated by the idealized print time, it also results in a wide range of bead widths, which cause the back pressure compensation print time to be very large. See Fig. 18. For an input outline shape which contains both very small and very large features, the constant bead count scheme produces bead widths which can fall outside of the range of manufacturable bead widths. Moreover the centrality marking is not robust against small perturbations in the outline; adding a small chamfer in a corner causes the unmarked ST to be very small at that location, which results in tiny bead widths. See top right of Fig. 17(b).

In Fig. 17(c) we can see that the centered beading scheme effectively deals with overfill and produces desired bead widths in all locations, except for the extrusion paths in the center, where the bead widths range between $0.25w^*$ and $1.8w^*$. However, it does produce some narrow underfill regions. Compared to the uniform technique the centered technique increases the (open) path count, but considerably reduces over- and underfill and decimates the number of toolpath angles below 45°. See Fig. 18.

However, according to Fig. 18(d) the centered scheme exhibits a wider range of bead widths than the distributed schemes: the standard deviation of the bead widths in the centered scheme is approximately 53 μm, while that of the distributed schemes is approximately 23 μm. [1] Moreover, because the quantization operator rounds to the nearest number of beads, in the worst case where we switch from a single to two beads the widths switch from $0.75w^*$ to $1.5w^*$, which is a considerably smaller range than in the centered scheme. We therefore conclude that the distributed schemes exhibit a lower bead width variation and lower (open) path count compared to the centered scheme.

Figs. 13, 17(d) and 17(e) show that in the inward distributed scheme the outer toolpaths have the preferred width more often than in the evenly distributed scheme, which means that the outline accuracy of the inward distributed beading is less affected by inaccuracies in the adaptive width control system. Furthermore, we find that compared to evenly distributed, the inward distributed scheme produces less corners with angles above 130° and less overfill, because the area of influence that bead count transitions have is limited in the inward distributed scheme. Thus the inward distributed scheme prevents over- and underfill, generates smooth toolpaths with more homogeneous width and affects smaller more centered parts of the print than the other schemes, while incurring little to no extra print time.

6.3. Limitations

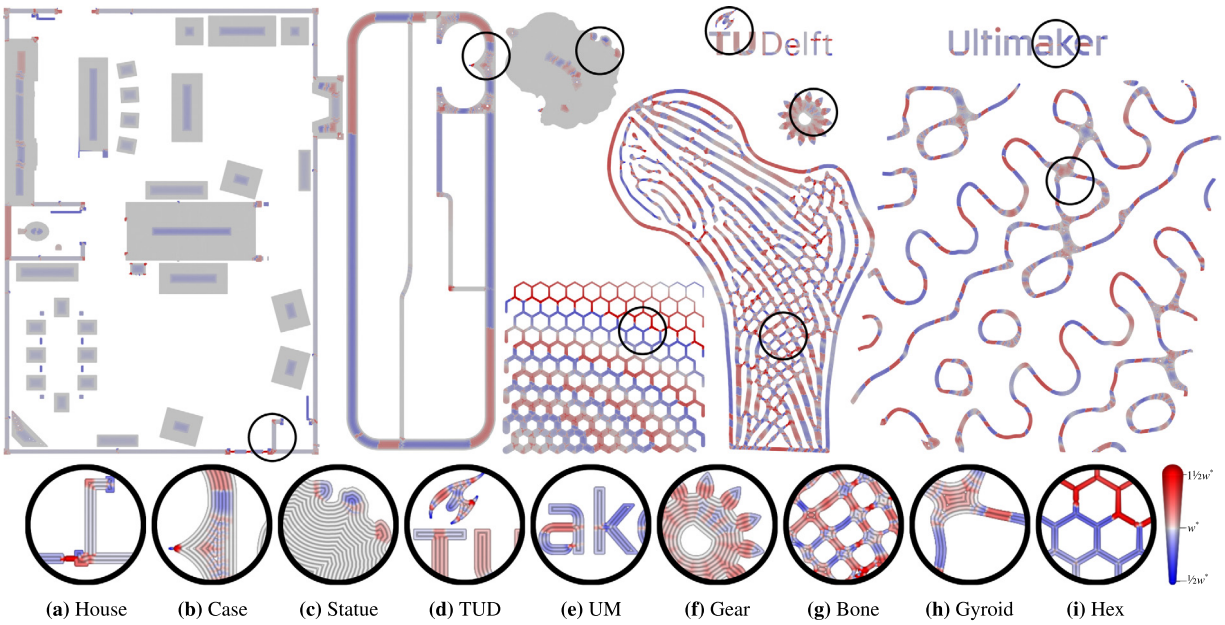
Because the performance of the various toolpathing techniques depends on the geometry of a model, they have ramifications for the practice of design for additive manufacturing. Because the naive method produces under- or overfill for parts of an outline with a constant diameter $d \neq 2iw^*$ it is best practice to design a model such that horizontal cross-sections have a feature diameter of an even integer multiple i of the bead width. For the centerscheme and for the current state of the art one should only avoid parts for which $(2i + 1.8)w^* < d < (2i + 0.25)w^*$ in order to avoid underfill. For the distributed schemes however, there is no diameter at which the framework produces under- or overfill for a part with a constant diameter d. The design consideration therefore reduces to limiting the diameter of your parts to be within the range $[w_{\min}, \infty)$, where $w_{\min}$ is a configurable parameter when using the widening meta-scheme.

The default limit bisector angle $\alpha_{\max} = 135°$ ensures that we do not employ transitioning in shallow wedge regions, which would result in a lot of short odd single bead polylines, which would break up the semi-continuous nature of polygonal extrusion paths; $\alpha_{\max} = 135°$ corresponds to $w^*/\cos \frac{1}{2}\alpha_{\max} \approx 0.4$ mm long segments and under-/overfill areas of $\frac{1}{4}(w^*)^2(\tan(\alpha/2) - \alpha/2) \approx 0.05\,\text{mm}^2$. However, future work might be aimed at reducing under-/overfill in regions with a low bisector angle without the introduction of short single polyline extrusion segments. If the over-/underfill problem is also solved for non-significant regions we might be able to increase $\alpha_{\max}$ and reduce the discontinuity introduced by short extrusion segments.

Another limitation of our method is that in a location v with locally maximal $R(v) \approx (i + \frac{1}{2})w^*$ the odd bead count will result in a single polyline extrusion segment consisting of only a single point. This can be viewed in the bottom right of Fig. 17(e) for example. In order to print such a dot, we make it into a 10 μm long extrusion segment, with an altered width such that the total volume remains correct. A more principled way of dealing with such a situation remains future work.

Finally it should be noted that although our framework can accurately emulate the constant bead count approach by Ding et al., its emulation of the centered approach by Jin et al. is

[1] Although the standard deviation σ of the inward distributed scheme is slightly higher than that of the evenly distributed scheme, the mean absolute deviation is lower (i.e. 9 μm versus 11 μm), because its distribution is more peaked.



(a) House (b) Case (c) Statue (d) TUD (e) UM (f) Gear (g) Bone (h) Gyroid (i) Hex

Fig. 19. Visualization of the widths for the output toolpaths of the inward distributed beading scheme ($N = 3$) applied to various example application objects. From left to right and top to bottom: a house, a case for electronics, a statue, two common logos, a gear, a topologically optimized bone structure, a tilted homogeneous gyroid structure and a heterogeneous thickness hexagonal grid.

imperfect. The transitions resulting from out framework introduce sharper corners and there is more width variation in those corners. Whereas the width of the connecting segment in the approach by Jin et al. is the preferred width w^*, the bead widths closer to the center resulting from our framework will be twice the local radius, which is larger than w^*. However, this inflated bead width variation is expected to have an insignificant impact on the measured bead width variation.

6.4. Applications

Toolpaths with varying width is particularly meaningful for narrow parts, since there the negative effect of under- and overfill is more pronounced than in wide parts. In extreme cases, thin features will not be filled at all. Therefore, our framework, while working for wide parts as well, shows most of its potential for objects which contain thin parts.

Fig. 19 collectively shows the application of the proposed inward distributed scheme for various types of 3D model, including both thin parts (architectural models, casings, embossed text, gears and microstructures) and wide parts (Fig. 19(b)) and organic shapes (Fig. 19(c)).

For architectural models and casings, preventing over- and underfill is expected to make them stronger. For embossed text, preventing underfill reduces the various holes in the top surfaces, which is detrimental to the visual quality of those top surfaces. For gears and similar mechanical parts that are designed with finite element analysis, the less variation in extrusion widths is closer to the assumptions under fast analysis (e.g. using homogenization [44]).

Of particular interest are microstructures that could be uniquely fabricated by 3D printing. For example, topology optimized bone-like structures [45] contain filaments of varying thickness that follow a varying stress direction (Fig. 19(g)). An angled Gyroid structure with uniform thickness also results in outline shapes with varying width (Fig. 19(h)). These structures are accurately densely filled using our framework. Another class of microstructures consists of parameterized patterns with varying thickness to achieve functional gradation. Fig. 19(i) shows the contour-parallel toolpaths with varying width of a hexagonal grid neatly switches between different bead counts over the volume, preventing the jagged moves a direction-parallel toolpaths would create for such a case [1].

7. Conclusion

In this paper we have introduced a framework for computing contour-parallel toolpaths employing adaptive bead width in order to minimize underfill and overfill areas. We introduced beading schemes which improve on the state of the art, and we have introduced a back pressure compensation method for accurate fabrication of adaptive width.

Our framework is flexible, demonstrated by the several beading schemes which emulate existing techniques. The computation times of our framework are on par with the state-of-the-art library for performing offsets of non-adaptive bead width. Our framework is stable: small local changes in the outline shape cause only small changes in the toolpath.

Compared to the state of the art, the inward distributed beading scheme reduces the amount of beads with a width deviating extremely from the preferred bead width by changing the width of several beads near the center instead of only the center-most bead. It is therefore expected to limit the impact of varying the bead width in terms of production accuracy and homogeneity of material properties, which in turn is helpful to efficiently simulate an FDM manufactured part.

The proposed beading scheme greatly improves the process planning for parts with thin contours, which often occur for example in architectural models, prototypes for casings or microstructures. Meanwhile it leaves most of the toolpaths the same as the uniform width technique in large features, meaning that existing studies which relate process parameters with mechanical properties of the print are still applicable. Compared to the naive approach of constant width toolpaths our beading scheme is expected to improve the stiffness, dimensional accuracy and visual qualities of the manufactured model. It is expected that as distributed beading schemes are implemented in commercial

Table 2
3D models used for validation.

Model name	Author
AirCasting Air Monitor Casing	HabitatMap
Air hose splitter	frizinko
Al Hamra Tower	TurnerConstructionCompany
canon NP-E3 battery cap	kosuyoung
David	Thunk3D
Deck Assembly Tool	PSomeone
Ender 3 Cable Chain	johnniewhiskey
Ergonomic Hacksaw Handle	mmOne
Gap measurement tool	ravm84
G-Clamp fully printable	johann517
Gyroid	Tim Kuipers
3D Printable Jet Engine	CATIAV5FTW
Lawn Mower Throttle Replacement	Spammington
OpenRC F1 Internal gear box mod	intoxikated
PCB Test Fixture	JMadison
Pioneer Radio Holder for Ford Focus	Perugino
Replicator Dual Fan Mount	aubenc
Atuador vers ao 2 * actuator version 2	Caroline Holanda
TEPocketOperatorHardcase	Salvation76
Screw sizer	Pierrolalune63
Bone-like optimized infill	Jun Wu
Two-Story Spec House	pwc-phil

software packages and bead width variation control become commonplace, the practice of design for additive manufacturing can disregard some of the nozzle size considerations.

The presented framework is open source available at github.com/Ultimaker/libArachne.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: A patent application based on the research reported in this paper has been filed.

Appendix A. Edge discretization

We calculate the location l of the boundary between a significant and nonsignificant portion of an edge analytically. For example, the parabolic MAT edge generated from the outline vertex $(0, 1)$ and an outline segment aligned with the X-axis follows $y(x) = 1/2x^2$ and $R(x) = y(x)$. We can determine the significant portion $[-x_{\text{bound}}, x_{\text{bound}}]$ by evaluating $\frac{\partial R}{\partial x} > \cos(\alpha_{\max}/2)$, which is $|x_{\text{bound}}| = (\tan(\alpha_{\max}/2))^{-1}$. Similarly, a MAT edge generated from two vertices at $(0, 0)$ and $(0, 1)$ follows $y(x) = 1/2$ and $R(x) = \sqrt{1/4 + x^2}$. The boundaries of significance are given by $|x_{\text{bound}}| = 1/2(\tan(\alpha_{\max}/2))^{-1}$. From these we can derive the locations $l = (\pm x_{\text{bound}}, y(x_{\text{bound}}))$. These specific cases can easily be transformed into all possible cases using scaling and rotation operations.

Appendix B. Data set

The data set we tested on was a custom selected set of open source 3D models found on the internet which was selected to cover a broad range of different types of application and geometry. Applications range from prototypes, to fixtures and mechanical end-use parts. The geometry covers a wide range including thin filaments, smooth surfaces, organic shapes, chamfered shapes, small shapes and large shapes. The models are described in Table 2.

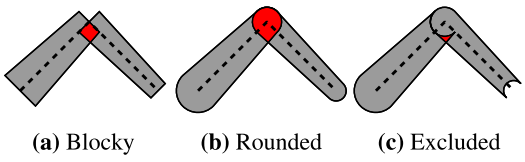


Fig. 20. Extruded area of two extrusion segments. Red areas signify doubly extruded areas. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Appendix C. Accuracy calculation

In order to estimate the overfill and underfill, we need to accurately calculate the area covered by a single extrusion path. If we would simply use an isosceles trapezoidal area, we would get overfill artifacts at corners in the toolpath (Fig. 20(a)). We therefore use a semi-circle (Fig. 20(b)) with a diameter equal to the starting width in the one end of each segment, and exclude it at the other end, because it will be included in the next segment. For polyline extrusion paths which are not closed, we also include the semi-circle of the destination location (Fig. 20(c)).

Using boolean operations we can obtain the polygonal regions for overfill and those for underfill. In order to deal with rounding errors we perform a morphological close of 5 μm, before calculating the total area in mm. We also calculate regions which are covered thrice by different extrusion segments and add twice its area to the total overfill area amount.

Appendix D. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2020.102907.

References

[1] Bates SR, Farrow IR, Trask RS. Compressive behaviour of 3D printed thermoplastic polyurethane honeycombs with graded densities. Mater Des 2018;162:130–42. http://dx.doi.org/10.1016/j.matdes.2018.11.019.
[2] Al-Ketan O, Rowshan R, Abu Al-Rub RK. Topology-mechanical property relationship of 3D printed strut, skeletal, and sheet based periodic metallic cellular materials. Addit Manuf 2018;19:167–83. http://dx.doi.org/10.1016/j.addma.2017.12.006.
[3] Maskery I, Sturm L, Aremu AO, Panesar A, Williams CB, Tuck CJ, et al. Insights into the mechanical properties of several triply periodic minimal surface lattice structures made by polymer additive manufacturing. Polymer 2018;152:62–71. http://dx.doi.org/10.1016/j.polymer.2017.11.049.
[4] Zegard T, Paulino GH. Bridging topology optimization and additive manufacturing. Struct Multidiscip Optim 2016;53(1):175–92. http://dx.doi.org/10.1007/s00158-015-1274-4.
[5] Wu J, Wang W, Gao X. Design and optimization of conforming lattice structures. IEEE Trans Vis Comput Graphics 2019;1–14, http://arxiv.org/abs/1905.02902.
[6] Cheng L, Bai J, To AC. Functionally graded lattice structure topology optimization for the design of additive manufactured components with stress constraints. Comput Methods Appl Mech Engrg 2019;344:334–59. http://dx.doi.org/10.1016/j.cma.2018.10.010.
[7] Ding D, Pan Z, Cuiuri D, Li H, Larkin N. Adaptive path planning for wire-feed additive manufacturing using medial axis transformation. J Cleaner Prod 2016;133:942–52. http://dx.doi.org/10.1016/j.jclepro.2016.06.036.
[8] Xiong Y, Park S-I, Padmanathan S, Dharmawan AG, Foong S, Rosen DW, et al. Process planning for adaptive contour parallel toolpath in additive manufacturing with variable bead width. Int J Adv Manuf Technol 2019. http://dx.doi.org/10.1007/s00170-019-03954-1.
[9] Jin Y, Du J, He Y. Optimization of process planning for reducing material consumption in additive manufacturing. J Manuf Syst 2017;44:65–78. http://dx.doi.org/10.1016/j.jmsy.2017.05.003.
[10] Ultimaker. Ultimaker cura 4.2.1 software. 2019, https://ultimaker.com/software/ultimaker-cura.
[11] Livesu M, Ellero S, Martínez J, Lefebvre S, Attene M. From 3D models to 3D prints: an overview of the processing pipeline. Comput Graph Forum 2017;36(2):537–64. http://dx.doi.org/10.1111/cgf.13147.

[12] Turner BN, Strong R, Gold SA. A review of melt extrusion additive manufacturing processes: I. Process design and modeling. Rapid Prototyping J 2014;20(3):192–204. http://dx.doi.org/10.1108/RPJ-01-2013-0012.
[13] Ahn SH, Montero M, Odell D, Roundy S, Wright PK. Anisotropic material properties of fused deposition modeling ABS. Rapid Prototyping J. 2002;8(4):248–57. http://dx.doi.org/10.1108/13552540210441166.
[14] Kuipers T, Wu J, Wang CC. CrossFill: Foam structures with graded density for continuous material extrusion. Comput Aided Des 2019;114:37–50. http://dx.doi.org/10.1016/j.cad.2019.05.003.
[15] Han W, Jafari MA, Danforth SC, Safari A. Tool path-based deposition planning in fused deposition processes. J Manuf Sci Eng 2002;124(2):462–72. http://dx.doi.org/10.1115/1.1455026.
[16] Steuben JC, Iliopoulos AP, Michopoulos JG. Implicit slicing for functionally tailored additive manufacturing. Comput Aided Des 2016;77:107–19. http://dx.doi.org/10.1016/j.cad.2016.04.003.
[17] Zhao H, Chen B, Gu F, Huang Q-X, Garcia J, Chen Y, et al. Connected fermat spirals for layered fabrication. ACM Trans Graph 2016;35(4):1–10. http://dx.doi.org/10.1145/2897824.2925958.
[18] Jin Y, He Y, Fu G, Zhang A, Du J. A non-retraction path planning approach for extrusion-based additive manufacturing. Robot Comput-Integr Manuf 2017;48:132–44. http://dx.doi.org/10.1016/j.rcim.2017.03.008.
[19] Held M, Spielberger C. A smooth spiral tool path for high speed machining of 2D pockets. Comput Aided Des 2009;41(7):539–50. http://dx.doi.org/10.1016/j.cad.2009.04.002.
[20] Huang N, Lynn R, Kurfess T. Aggressive spiral toolpaths for pocket machining based on medial axis transformation. J Manuf Sci Eng 2017;139(5). http://dx.doi.org/10.1115/1.4035720.
[21] McMains S, Smith J, Wang J, Sequin C, Carlo S. Layered manufacturing of thin-walled parts. In: ASME Design Engineering Technical Conference, Baltimore, Maryland. Citeseer; 2000.
[22] Jin YA, He Y, Fu JZ. An adaptive tool path generation for fused deposition modeling. In: Advanced Materials Research, vol. 819. 2013, p. 7–12. http://dx.doi.org/10.4028/www.scientific.net/amr.819.7.
[23] Ding D, Pan Z, Cuiuri D, Li H. A tool-path generation strategy for wire and arc additive manufacturing. Int J Adv Manuf Technol 2014;73(1–4):173–83. http://dx.doi.org/10.1007/s00170-014-5808-5.
[24] Cox JJ, Takezaki Y, Ferguson HRP, Kohkonen KE, Mulkay EL. Space-filling curves in tool-path applications. Comput Aided Des 1994;26(3):215–24. http://dx.doi.org/10.1016/0010-4485(94)90044-2.
[25] Griffiths JG. Toolpath based on Hilbert's curve. Comput Aided Des 1994;26(11):839–44. http://dx.doi.org/10.1016/0010-4485(94)90098-1.
[26] Shaikh S, Kumar N, Jain PK, Tandon P. Hilbert Curve based toolpath for FDM process. In: CAD/CAM, Robotics and Factories of the Future. Lecture Notes in Mechanical Engineering, Springer; 2016, p. 751–9. http://dx.doi.org/10.1007/978-81-322-2740-3_72.
[27] Kao J-h, Prinz FB. Optimal motion planning for deposition in layered manufacturing. In: Proceedings of DETC, vol. 98. Citeseer; 1998, p. 13–6.
[28] Jin Y, He Y, Du J. A novel path planning methodology for extrusion-based additive manufacturing of thin-walled parts. Int J Comput Integr Manuf 2017;30(12):1301–15. http://dx.doi.org/10.1080/0951192X.2017.1307526.
[29] Moesen M, Craeghs T, Kruth JP, Schrooten J. Robust beam compensation for laser-based additive manufacturing. Comput Aided Des 2011;43(8):876–88. http://dx.doi.org/10.1016/j.cad.2011.03.004.
[30] Behandish M, Mirzendehdel AM, Nelaturi S. A classification of topological discrepancies in additive manufacturing. Comput Aided Des 2019. http://dx.doi.org/10.1016/j.cad.2019.05.032.
[31] Eiamsa-ard K, Liou FW, Landers RG, Choset H. Toward automatic process planning of a multi-axis hybrid laser aided manufacturing system: skeleton-based offset edge generation. In: ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers Digital Collection; 2003, p. 227–35. http://dx.doi.org/10.1115/DETC2003/DAC-48726.
[32] Blum H, et al. A transformation for extracting new descriptors of shape. Models Percept Speech Vis Form 1967;19(5):362–80.
[33] Lee D-TT. Medial axis transformation of a planar shape. IEEE Trans Pattern Anal Mach Intell 1982;PAMI-4(4):363–9. http://dx.doi.org/10.1109/TPAMI.1982.4767267.
[34] Chazelle B, Incerpi J. Triangulation and shape-complexity. ACM Trans Graph 1984;3(2):135–52. http://dx.doi.org/10.1145/357337.357340.
[35] Fournier A, Montuno DY. Triangulating simple polygons and equivalent problems. ACM Trans Graph 1984;3(2):153–74. http://dx.doi.org/10.1145/357337.357341.
[36] Schäling B. The boost c++ libraries. Boris Schäling; 2011.
[37] Fortune S. A sweepline algorithm for voronoi diagrams. Algorithmica 1987;2(1):153. http://dx.doi.org/10.1007/BF01840357.
[38] Attali D, Montanvert A. Modeling noise for a better simplification of skeletons. In: Proceedings of 3rd IEEE International Conference on Image Processing, vol. 3. IEEE; 1996, p. 13–6. http://dx.doi.org/10.1109/ICIP.1996.560357.
[39] Sud A, Foskey M, Manocha D. Homotopy-preserving medial axis simplification. In: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling. Association for Computing Machinery; 2005, p. 39–50. http://dx.doi.org/10.1145/1060244.1060250.
[40] Kuipers T, Elkhuizen W, Verlinden J, Doubrovski E. Hatching for 3D prints: Line-based halftoning for dual extrusion fused deposition modeling. Comput Graph 2018;74:23–32. http://dx.doi.org/10.1016/j.cag.2018.04.006.
[41] Ertay DS, Yuen A, Altintas Y. Synchronized material deposition rate control with path velocity on fused filament fabrication machines. Addit Manuf 2018;19:205–13. http://dx.doi.org/10.1016/j.addma.2017.05.011.
[42] Tronvoll SA, Popp S, Elverum CW, Welo T. Investigating pressure advance algorithms for filament-based melt extrusion additive manufacturing: theory, practice and simulations. Rapid Prototyping J. 2019;25(5):830–9. http://dx.doi.org/10.1108/RPJ-10-2018-0275.
[43] Johnson A. Clipper 6.4.2 - an open source freeware library for clipping and offsetting lines and polygons. 2017.
[44] Liu X, Shapiro V. Homogenization of material properties in additively manufactured structures. Comput Aided Des 2016;78:71–82. http://dx.doi.org/10.1016/j.cad.2016.05.017.
[45] Wu J, Aage N, Westermann R, Sigmund O. Infill optimization for additive manufacturing – Approaching bone-like porous structures. IEEE Trans Vis Comput Graphics 2018;24(2):1127–40. http://dx.doi.org/10.1109/TVCG.2017.2655523.