Iterative carving for self-supporting 3D printed cavities

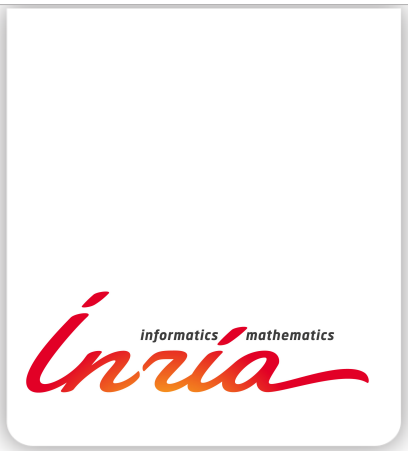
Samuel Hornus, Sylvain Lefebvre

Iterative carving for self-supporting 3D printed cavities

Samuel Hornus, Sylvain Lefebvre

RESEARCH REPORT
N° 9083
July 2017
Project-Team Alice

ISSN 0249-6399 ISRN INRIA/RR--9083--FR+ENG



Iterative carving for self-supporting 3D printed cavities

Samuel Hornus, Sylvain Lefebvre

Project-Team Alice

Research Report n° 9083 — July 2017 — 19 pages

Abstract: Additive manufacturing technologies fabricate objects layer by layer, adding material on top of already solidified layers. A key challenge is to ensure that there is always something below the layer being fabricated: otherwise added material simply falls under the effect of gravity – this is a critical issue with most technologies.

This difficulty is typically treated differently for the interior and the exterior of an object: the exterior is supported through structures akin to scaffoldings that are removed afterwards, while the interior volume is filled with a sparse, self-supporting infill pattern. The infill acts as a support for the tops and also provides mechanical strength to the final print. However, as objects grow in size even sparse infills become prohibitively expensive in time and material usage, and if made too sparse they no longer provide the required support. This prevents printing large objects with empty interiors.

In this work we investigate how to compute as large as possible empty cavities. Our technique is based on an iterated geometric carving algorithm, that is fast to compute and produces nested sets of inner walls. The walls have exactly the minimal printable thickness of the manufacturing process everywhere. Remarkably, our technique only manipulates two consecutive slices of the model at a time, sweeping through the model along the vertical direction, from top to bottom. The inner cavity is self-supporting almost everywhere. It becomes fully printable after adding only a few auxiliary, bridge-like, support structures. We discuss an efficient, robust implementation in details, based on a filtered polygonal medial axis extraction and an exact polygon offsetting technique. Using our approach, we can print large parts on filament printers using as little as a single filament thickness, providing one order of magnitude reduction in print time and material.

Key-words: additive manufacturing, FDM, cavity, support

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Modélisation itérative de cavités auto-portantes pour la fabrication additive

Résumé : Nous étudions la modélisation de cavités auto-portantes aussi volumineuses que possible à l'intérieur d'un volume donné. Nous montrons comment les utiliser pour modéliser une structure très légère à l'intérieur d'un objet, qui viendra supporter les surfaces de l'objet tournées vers le haut, afin de pouvoir les construire correctement dans un mécanisme de fabrication additive.

Mots-clés : fabrication additive, FDM, cavité, support

This paper has been updated with an errata, regarding limitations and specific cases of local minima. Changes to the text are highlighted in blue. These additions also apply to [12] and [10].

1 Introduction

Additive manufacturing enables the fabrication of objects of unprecedented complexity. This is achieved by building an object layer after layer, from bottom to top, each successive layer bonding to the layer below to form the final shape. This affords for the production of parts having complex shapes, that may also include complex structures and cavities inside.

Due to its sequential nature the 3D printing process is relatively slow, especially as thin layers are required to achieve a good precision. Its performance unfortunately scales poorly with object size as the inner volume grows quickly, to the cube of the object extent. Thus, several techniques have been proposed to attempt to reduce the time taken by the process. One efficient way to do so is to empty the interior of the object, keeping only a thick shell around the surface. To provide additional mechanical strength, the interior is then filled with a special sparse infill pattern. This is the approach implemented in most slicing software.

In principle, the infill pattern could be entirely removed, thus resulting in large savings (print time and material usage then depend on surface area instead of volume). Unfortunately, this does not work due to the constraints of the additive manufacturing process.

Indeed, most technologies operate by solidifying small areas along paths (FDM, SLS) or at discrete locations (DLP-SLA). The solidification process often requires the solidified area to be supported by the layer below. This is for instance the case with filament printers (FDM), with resin printers (DLP-SLA) but also with metal sintering (SLM) where the melted metal would sink in powder unless a solid layer lies below. This leads to the requirement of having external support structures to be able to fabricate shapes with overhanging areas with respect to the build direction.

The same applies to cavities. When a part is hollowed by a standard operation (*e.g.* taking the difference between the part and an erosion of it), a hull with constant thickness is produced. Most often, the tops of the cavity cannot be printed directly, as they are not supported from below. This imposes the use of an infill pattern, dense enough to provide the necessary support. While this is acceptable on small parts – it provides mechanical strength – it quickly becomes prohibitive in time and material on larger parts. Another approach would be to use the equivalent of an external support inside the print, which would then remain trapped inside. Unfortunately these are either large and will use as much time and material as an infill, or they are sparse but require a relatively slow optimization (see Section 2).

In this work we introduce a simple, efficient technique to generate large empty cavities inside 3D prints. The synthesized cavities are self-supporting and use the minimal printable thickness of the process: using our technique and no infill, all printed walls exactly have the minimal thickness thus significantly reducing material use and print time. Computing our cavities requires a single top-to-bottom sweep, maintaining only two slices in memory. Therefore, the process scales to arbitrary large parts. The print paths created by our technique have additional desirable properties: they are smooth and continuous, and they remain close to the original surface, resulting in faster and more reliable prints.

2 Previous work

We know discuss related works in the area of support structures and interior infills. As there is a large body of work we focus only on the most closely related to ours and refer the interested reader to the survey by Livesu and colleagues [21].

Note that part orientation plays a primordial role in minimizing the need for supports – this is however often a multi-criteria problem (see [21], Section 3.2). We assume here that the part orientation has been optimized beforehand and cannot be changed.

2.1 External supports

External supports are typically designed to have a small size, print fast, and be easily removable. A first set of methods, implemented in most 3D printing software, extrude volumes downwards from the faces of a model requiring support. These are then filled with a sparse pattern. Several methods reduce the size of the volume, using sloped walls [14] and decreasing support complexity as the distance below the part increases [8]. Cacace *et al.* [1] fill-in regions of overhang with self-supporting volumes of minimal size, formulating a shape optimization problem in a level set technique. The generated support volumes are small but still completely filled, and the optimization in a volume field is expensive.

Other techniques create sparse support structures made of beams, decimating a base mesh forming a rhombic self-supporting structure [6], or forming a scaffolding with sets of vertical, angled and horizontal bars [31, 13, 33, 35, 5]. The set of pillars is determined through greedy procedures which can be time consuming when large surfaces are to be supported.

All these approaches are primarily designed for external supports. They cannot exploit the fact that when used inside, they are surrounded by the object which itself may provide additional support. In addition, sparse structures printed with beams are less reliable on many technologies, in particular filament deposition: they produce many discontinuities in motion and extrusion and are subject to warping and sagging.

2.2 Internal supports

In additive manufacturing, objects are rarely fabricated completely solid. The main reasons are to reduce the part weight, material use, as well as print time. We first mention methods that fill an object with a pattern that is mechanically strong, and techniques that on the contrary attempt to make the part as hollow as possible – which is our objective.

Most software for additive manufacturing propose two typical dense fill patterns [4]: infilling contours with parallel lines of varying spacings (referred to as a zigzag pattern [26]), and infilling with offset contours. Both patterns can be mixed: a few offset contours are created, and then a zigzag pattern fills-in the remainder. The zigzag pattern can be made sparse by simply increasing the spacing between each successive line, thus forming internal vertical walls. Two such patterns are often interleaved, one along X and one along Y, to produce a grid in the XY plane, vertically extruded along the Z build direction. These patterns print fast, provide reasonable strength, and act as inner supports for top surfaces as long as the spacing is kept small. More elaborate patterns have been proposed, such as Hilbert spirals [7], Moore's spirals [3], Peano spirals [23] and Fermat spirals [38]. The later are specifically designed to maximize continuity and smoothness, thus resulting in more reliable prints. Our cavities offer similar properties.

Sparse patterns forming 3D cells have been proposed to provide an increased mechanical strength. The software *Slic3r* proposes an infill pattern that produces a 3D honeycomb structure. These however have many sharp turns within a slice. Simpler rhombic 3D structures have been

obtained by printing sets of parallel lines in each slice, carefully offsetting the lines from one slice to the next such as to produce slanted, self-supporting walls [18]. These structures can be locally subdivided as they define a slanted regular grid, for instance following a stress analysis [36]. Many other techniques have been proposed to create strong infills, based on Voronoi diagrams [22, 24], periodic tiles [2, 29, 19, 32, 28], trusses [30, 35, 37], and subdividing tessellations [27].

All these approaches focus on mechanical properties. Instead, we seek to produce parts which are as hollow as possible but can be fabricated.

2.3 Hollowing

When mechanical strength is not a primary objective, one can consider completely emptying the part interior, for instance using surface offset techniques and boolean operations [9, 34, 25]. However, the resulting interior is typically non self-supporting, and cannot be printed on technologies with overhang constraints. To produce large inner cavities, Lee *et al.* [17] propose to start from a full 3d rhombic structure similar to [18] and decimate it. The algorithm greedily removes all faces which are not required to support the structure above, leaving a sparse self-supporting hollowed interior. Besides the difficult optimization, a drawback of this approach is that it requires to choose the density of the initial structure: if too dense it will use too much material, if too sparse it will not provide enough support. This is a choice difficult to make without trial and error printing. We compare our result to this technique as it shares a similar objective.

Closer to our work, Hornus and colleagues [11] considered the following problem: Given a volumetric shape O, what is the largest shape W that fits in O while having a printable boundary? The word *printable* here, means that given a sliced representation of W, it should be possible to print only the contour of each slice without failure, leaving the inside of W actually empty in the resulting 3d print. In their report, they propose a technique to model such a large cavity. We write $\textsc{Cav}(O)$ to denote the shape obtained by applying this technique.

While this inspired our technique, we introduce the following novel ideas. First we show that the $\textsc{Cav}()$ procedure can be applied iteratively in order to obtain printed objects that are almost completely empty, having only walls of the minimal printable thickness. This significantly decreases the fabrication time and, in particular, allows the printing of large object. Second, while the work in [11] uses a bitmap slice representation, we describe here a robust *polygonal* implementation of the $\textsc{Cav}()$ procedure and its iterative application, which is significantly faster when processing tall objects, and produces smoother paths.

3 Carving cavities inside a shape

We now describe our algorithm for synthesizing self-supporting cavities. The idea of the approach is to sweep through the object slices from top to bottom and, at each slice, analyze whether a cavity can be started by "tearing" a filled area open. The tear is then propagated downwards through morphological dilation, producing a self-supporting surface.

A single cavity would leave potentially large filled volumes between its walls and the actual surface. Thus, instead of producing a single cavity, we produce nested cavities: remaining areas are iteratively analyzed and teared, spawning sub-cavities.

Note that the iterations happen within a single slice, therefore the whole process requires a single sweep from top to bottom.

After introducing notations (Section 3.1) we describe the synthesis of simple, non-iterated cavities in Section 3.2 and the full process in Section 3.3.

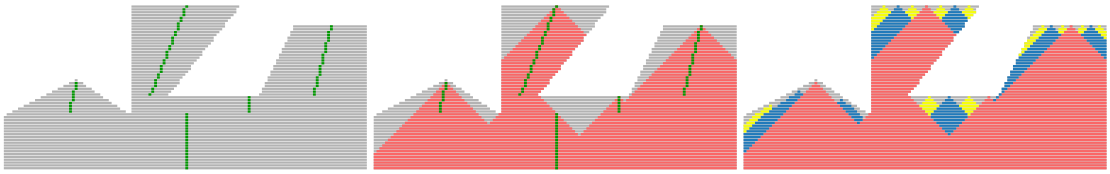


Figure 1: In this 2D world, the objects are represented as a stack of slices and each slice is a set of collinear horizontal line segments. *Left:* The main object is O (gray). The seed shape for the cavity is the medial axis of each slice of O, which, in 2D is simply the center of each segment (green). *Middle:* The cavity $C^1 = \textsc{Cav}(O)$ is drawn red. *Right:* Iterated cavities. C^1 is drawn red, C^2 blue, C^3 yellow and C^4 green.

3.1 Notations

Given a shape $X \subset \mathbb{R}^3$ we define the *slice of* X at height z, written $X_{|z}$, as the intersection of X with an horizontal plane at height z. Shapes are manipulated as a finite set of slices $\{X_{|i}, i = 1 \dots n\}$ ordered from the bottom up. Here, the notation i is an index in the bottom up ordering, instead of the actual height of the corresponding slice. If S is a planar shape and r a non-negative number, the *dilation* $S^{\uparrow r}$ is the set of points at distance r or less from S: $S^{\uparrow r} = \{p \mid \exists q \in S, |p - q| \leq r\}$. Similarly, the *erosion* of S of radius r is $S^{\downarrow r} = \overline{\overline{S}^{\uparrow r}}$ where $\overline{X}$ is the complement of X: $\overline{X} = \mathbb{R}^2 \setminus X$. The *opening* of S of radius r is $\textsc{Open}(S, r) = S^{\downarrow r \uparrow r}$. The *closing* of S of radius r is $\textsc{Close}(S, r) = S^{\uparrow r \downarrow r}$.

Let s be the radius of the extrusion nozzle (typically 0.2 mm). Let r be the distance by which we dilate each cavity when propagating it from one slice to the slice below (see below). The value of r is typically chosen so that the slope of the surface of a cavity is 45 degrees, see [11].

3.2 Non-iterated carving

The goal of the (non-iterated) carving technique is to model a large volume $C \subset O$ that can be completely carved out of a given object O, while still maintaining 3d-printability.

3.2.1 Expansion using morphological dilation.

Assuming that the object O is already supported from the outside if necessary, carving C out of O implies that the part of the boundary of C that lies inside O must be self-supported. This translates into a lower bound on the slope of this surface, which ensures that a deposited filament has enough support below it to stay put and not fall. Some part of ∂C may coincide with ∂O, in which case we assume that they are supported, if required, from the outside of O.

Since our cavity should be as large as possible, we model the cavity C by sweeping its slices from top to bottom while expanding the volume as much as possible. This is achieved by incorporating the shape $C^{\uparrow r}_{|i+1}$ into the slice $C_{|i}$ immediately below.

3.2.2 Seeding the growth of the cavity.

In order to seed the growth of a cavity C inside O, we need a shape that can serve as topmost slice of C and bootstrap the growth of the cavity C in the slices below. Since the boundary of C must be closed and we print only the boundary of its slices, the seed shape has to be close to a zero- or one-dimensional shape. We use a filtered medial axis of each slice as the seed shape.

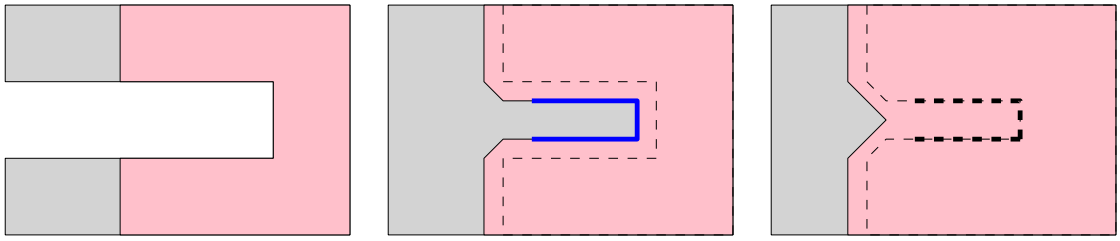


Figure 2: Illustration of an edge-like local minima. This shows three slices of an object. The slice $i+1$ (leftmost) has a narrow concavity on its left side, while the two next slices below are rectangular. The cavity region is shown in pink. The narrow concavity in $i+1$ becomes a concavity in the cavity polygon in slice i. It entirely disappears when dilating the cavity polygon from i to $i-1$, leaving the cavity boundary above (in blue) unsupported.

Section 5.2 gives more details. The pseudocode for the procedure CAV is shown in Algorithm 1. Figure 1 demonstrates the modeling of a cavity in a 2D world. Figure 3 illustrates the cavity modeling process in 3D on a selected number of slices. In this figure, the input shape is a cube of side length 12 mm (60 slices of height 0.2 mm) indented at an upper corner by a cuboid of size $6 \times 6 \times 3$mm. The left column shows the seeding (green) and growth of the first cavity $C^1 = \text{CAV}(O)$ (red) where O is the input shape with some "cover" removed (hence the empty 58-th slice).

Algorithm 1 CAV computes a single cavity inside O.

```
1: function CAV(Shape O)
2:     n ← the number of slices of O
3:     C_{|n+1} ← empty slice
4:     for i = n downto 1 do
5:         G ← C^{↑r}_{|i+1}                    ▷ Dilate the previous slice
6:         S ← SKEL(O_{|i})                     ▷ Seed shape for that slice
7:         C_{|i} ← (G ∪ S) ∩ O_{|i}            ▷ C_{|i} has to stay in O_{|i}
8:     return cavity C
```

Local minima. Once modeled using the CAV algorithm, the surface of the cavity may exhibit local minima. We distinguish between two types of local minima.

The first ones are *islands*: these are downward facing spikes. Their contour disappears through successive dilations of the cavity regions. Cavity C^1 in Figure 1-middle has three such minima.

The second ones are *edge-like* local minima. They correspond to strips of width below $2r$ that extend inward the cavity polygon (narrow concavities). Such a case is shown in Figure 2. These strips are completely removed by the dilation of the cavity, potentially resulting in large dangling narrow regions.

Since the surface of the cavity has to be printed, additional support is required below each local minimum. We model this support as simple bridges anchored on the side of the cavity; we give details in Section 4.4. Hornus *et al.* [11] argue that using the medial-axis as seed shape tends to minimize the number of local minima.

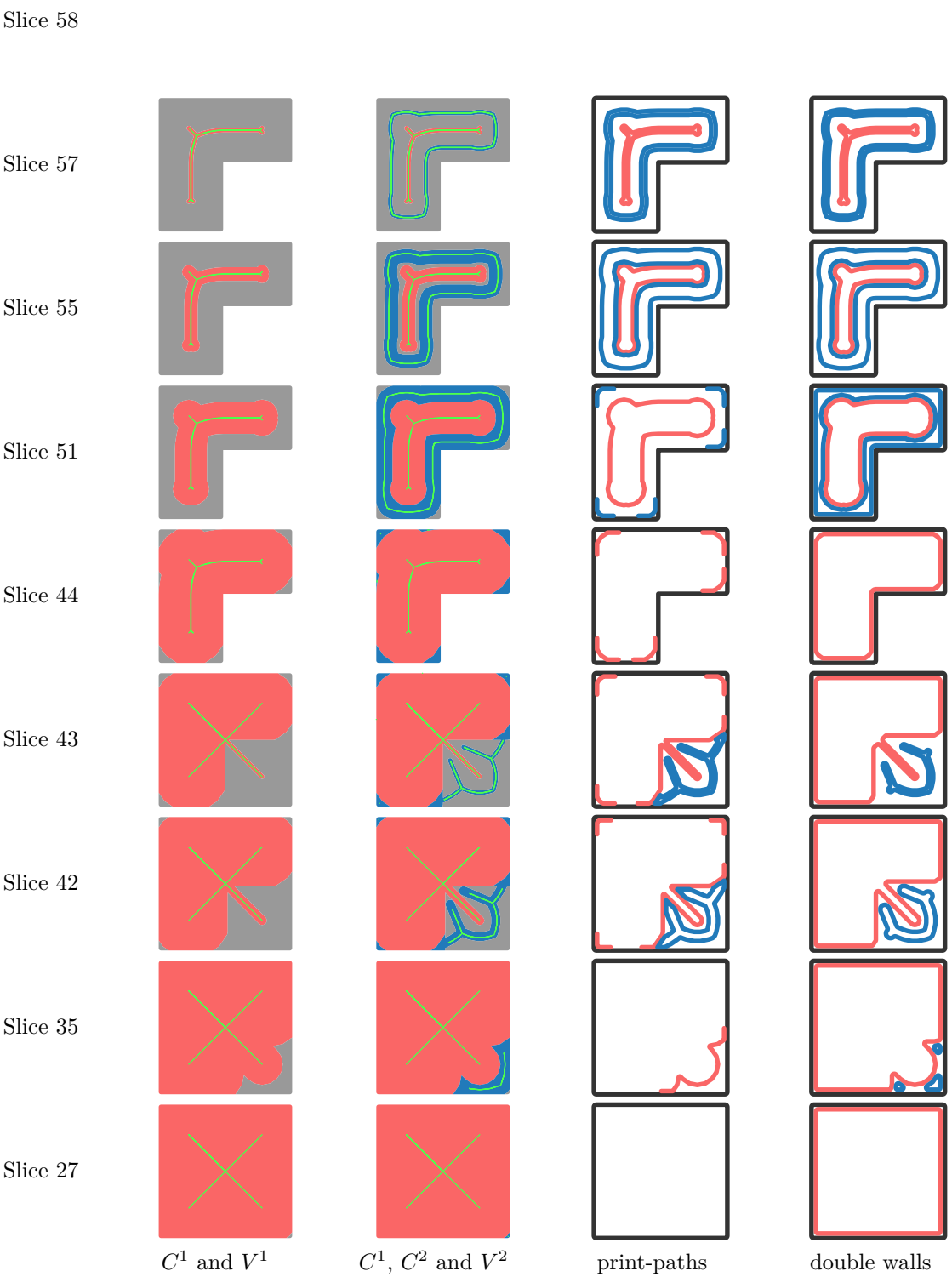


Figure 3: An example of modeling cavities with two iterations. Same color scheme as Figure 1.

3.3 Iterated carving

By observing Figure 1-middle, one can see that the complement of the cavity, $O \setminus C^1$, still makes for a significant volume. It then does make sense to apply the CAV procedure again in order to model a second cavity in this region, and to iterate this carving process a number of times, each time carving a cavity in the volume that the previous cavity was not able to cover. The pseudocode for the procedure CAV is shown in Algorithm 2. Figure 1-right demonstrates the modeling of a 4 iterated cavities in a 2D world. The second column of Figure 3 illustrates the 3D modeling of the second cavity C^2 (blue), including the seed-shape computed for $V^1 = O \setminus C^1$ (green).

Algorithm 2 The function ITERATIVECARVING.

```
1: function IterativeCarving(Shape O, Integer k)
                                   ▷ k is the number of iterations to perform.
2:     V^0 ← O
3:     for i = 1 to k do
4:         C^i ←Cav(V^{i−1})
5:         V^i ← V^{i−1} \ C^i
6:     return (C^1, C^2, ..., C^k, V^k)
```

The number of iterations is either fixed (from 3 to 6 typically) or the iterative process stopped when a condition is met: when no useful cavity can be created in the remaining volume V^i. This roughly corresponds to the case when any slice of V^i disappears when eroded with a disk of radius equal to the diameter of the extrusion head of the printer.

For simplifying the exposition, we write $C^0 = \mathbb{R}^3 \setminus O$. At the end of the call to ITERATIVECARVING one obtains a partition $O = V^k \sqcup \bigsqcup_{i=1}^{k} C^i$, and it holds that $\mathbb{R}^3 = O \sqcup C^0 = V^k \sqcup \bigsqcup_{i=0}^{k} C^i$. (The symbol $\sqcup$ stands for the disjoint union.)

Top-facing surfaces close to being horizontal are usually printed densely so as to avoid holes in the surface of the object. This dense part is known as the *cover* and is typically 2 to 6 layers thick. When modeling the cavities inside a shape, one should not touch the cover, so that the parameter O given to the ITERATIVECARVING procedure should corresponds to the main object of interest with its cover subtracted. When enough iterations are used, the union of the boundaries of all the cavities just below the cover are usually enough to provide a good support for the cover (Figure 1-right). Alternatively, one may choose to fill the volume V^k with some common infill pattern.

4 From cavities to print-paths

If we target a printer that solidifies discrete elements (e.g. DLP-SLA) then the cavity outlines can be thickened and rasterized directly. For a vector solidification process, such as FDM or SLS, the sliced representation of the cavities has to be turned into curves along which material should be solidified or deposited in a specific order. We call these curves *print-paths*. In the remainder we focus specifically on the case of widely available filament printers.

If we think of a real (thick) slice of a printed object as a flat planar shape, then that shape is equal to the Minkowski sum (or convolution) of the print-paths for that slice with a zero-centered disk of radius s, the radius of the extrusion nozzle.

The slices are processed independently of each other so that one can describe the process for a single slice. In a slice at height z, the object O and each cavity C^i are reduced to planar shapes

$O_{|z}, C^i_{|z}$. Each boundary coincides locally with the boundary of another shape (a cavity C^i or V^k or O), so these boundaries can not be used directly as print-paths.

4.1 Double walls

The simplest way to turn these boundaries into non-overlapping print-paths is to erode each planar shape by a disk of radius s: $P^i_{|z} \leftarrow C^{i\downarrow s}_{|z}$. Then, the common boundary curve of two cavities becomes two print-paths running in parallel at a distant of $2s$ from each other ($2s$ is the extrusion diameter). [1]

Small components are then removed and the remaining ones are smoothed in order to ease the motion of the extrusion head. This is accomplished by opening the cavities: $P^i_{|z} \leftarrow \textsc{Open}(P^i_{|z}, s')$ where s' is slightly smaller that s. We use $s' = \frac{9s}{10}$.

After this step, the boundaries of the planar shapes in slices $C^0_{|z}$ and $P^i_{|z}, 1 \leq i \leq k$, are reinterpreted as collections of print-paths, which can be readily written, say, in a GCode file, for the layer at height z. As one can see on the rightmost column of Figure 3 we obtain what looks like a network of double parallel print-paths. In Section 4.3, we describe a refined technique to obtain a similar network of isolated print-paths, so as to almost halve the amount a material required to print the slice.

4.2 Print-paths ordering

In a given slice, the (possibly numerous) contours of a single cavity can be printed in any order. They are independent of each other. However, print-paths from several cavities have to be printed in a specific order: A small contour from a cavity may have as only support a contour from another cavity with which it is in contact. We must guarantee that when a contour is printed, at least one touching contour has already been printed. This is the case if we print the boundaries of the cavities in the order they were modeled: $C^0, C^1, \ldots, C^k$.

4.3 One-filament-thick cavity walls

We take advantage of the ordering of the cavities as follows: When extruding fused material along a print-paths p on the boundary curve of cavity C^i, we can assume that the print-paths for cavities $C^j, j < i$ have already been "printed." We then simply avoid extruding material on the parts of print-path p that coincide with a boundary of some cavity that has already been printed.

In order to describe the process in detail, we recall some notation: V^i is the volume left from O after carving the first i cavities: $V^i = O \setminus \bigcup_{j=1}^{i} C^j$. Said another way, V^i is the volume out of which we model the cavity C^{i+1} using the Cav procedure. The boundaries of a slice $V^i_{|z}$ is composed of pieces of boundaries from $C^0, \ldots, C^i$ that can be thought of as print-paths, although they haven't been through smoothing and filtering of small components yet. The future print-paths for $C^{i+1}_{|z}$ must lie at a distance at least $\approx 2s$ away from these boundaries, *i.e.*, lie in an erosion of $V^i_{|z}$. For this reason, we compute a *restriction shape* $W^i_{|z} \leftarrow V^{i\downarrow t}_{|z}$. We use a value of $t = 1.4s$ that is slightly smaller than $2s$ for reasons explained below. The print-paths are obtained as follows:

First, the cavity slices are smoothed and small components removed: $P^i_{|z} \leftarrow \textsc{Open}(C^i_{|z}, \frac{9s}{10})$.

[1] In this setup, the planar shape of the main object, $O_{|z}$, should be dilated instead of eroded. Of course, since the shape O is an input that we cannot modify, the cavities are instead modeled from pre-eroded slices $O^{\downarrow s}_{|z}$. In Section 4.3 we do not need to pre-erode the slices $O_{|z}$ of the main object.

Then, we compute the intersection of both the cavity and its boundaries with the restriction shape:

$$Q^i_{|z} \leftarrow P^i_{|z} \cap W^i_{|z} \tag{1}$$
$$\mathcal{R}^i_{|z} \leftarrow \partial P^i_{|z} \cap W^i_{|z}. \tag{2}$$

Unlike all the other elements above, $\mathcal{R}^i_{|z}$ is not a 2D planar shape but a set of open and closed curves in the plane. These curves form a subset of the boundaries of $Q^i_{|z}$: $\mathcal{R}^i_{|z} \subset \partial Q^i_{|z}$. The parts of $\partial Q^i_{|z}$ missing in $\mathcal{R}^i_{|z}$ are precisely the parts where a print-path is not necessary because one will already be placed there in the processing of the slice of a cavity $C^j, j < i$.

The open curves thus obtained make for fragile structures when melted material is extruded along them: Indeed, in that case, the filament has no strong support at its ends, neither below (the support is weak because the surface is strongly angled, typically at 45 degrees) nor on its sides since we are printing 1-filament-wide cavity walls. This may lead to the filament not staying in place over several layers at the bottom of a new cavity wall, until the situation stabilizes. To overcome this issue, we extend the open curves of $\mathcal{R}^i_{|z}$ along the boundaries $\partial Q^i_{|z}$, over a distance of about 1 mm at either ends of the each curve. The purpose of this extension is to sticks to the filaments already extruded along the boundaries of the cavities $C^j, j < i$. The value of t is chosen a bit smaller than $2s$ to have the melted filament adhere well to the neighboring filaments.

4.4 Bridge supports for local minima

As we mentioned earlier, local minima on the boundary of a cavity should be supported from bellow by adding a bridge inside that cavity. We describe below how the local minima are detected and the bridges modeled.

4.4.1 Detecting local minima

Intuitively, when a horizontal plane at height z slices a cavity C just above a local minima, the slice for that cavity ($C_{|z}$, a planar shape) contains a hole h that itself does not contain any other component of the cavity. We use a hierarchical representation of the slice boundaries to detect these empty holes and consider them as candidate local minima. (The hierarchical representation is available in the *Clipper* library that we use [15].) Once the slice for the same cavity at the height $z' < z$ immediately below z is available, we check that $C_{|z'}$ completely covers the candidate hole h, which indicates that that hole h vanished due the growth of the cavity (or any other factor involved in the change of the slice.) In practice we check that $h \setminus C_{|z'} = \emptyset$. Holes that are completely covered are considered as local minima that should be supported.

In our current implementation we do not add extra support for the edge-like local minima — in most cases these are narrow strips extending inwards from the cavity border that do not endanger printability. However, some bad cases can appear where these features support larger parts above, requiring additional bridge supports. Edge-like local minima can be detected as follows. Let us denote by $\mathcal{U}$ the polygon $\mathcal{U} = C_{|z} \setminus C^{\downarrow r}_{|z'}$ that contains the cavity regions which are not supported by the cavity below. If we ignore the previously detected local minima, the remainder are edge-like local minima. Some of these are acceptable: any corner sharper than r produces such a tiny spurious spike, but those will not produce any significant defects. Therefore, the polygon $\mathcal{U}$ can be filtered to remove any region whose area is below some multiple of r^2. What remains has to be supported by an auxiliary structure such as a bridge.

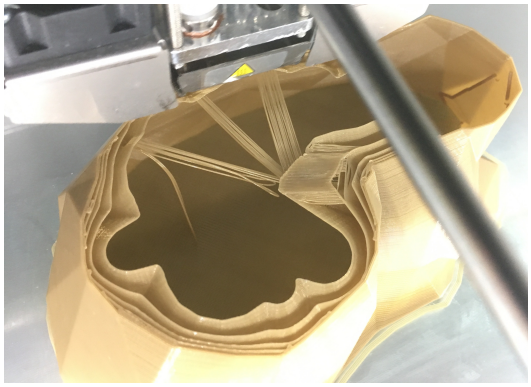


Figure 4: An unusually high number of bridges appears during the printing of the Fawn model.

4.4.2 Modeling bridges

Let h be a hole in slice $C_{|z}$ that has been detected as a local minima. We fill the hole in $C_{|z}$ and, when generating the GCode for that slice, replace it by a set of parallel lines. The lines are placed so that the extruded filament along them would completely cover the hole h. The lines are extended in both direction up to the boundary of the cavity, where the extruded filament can stick. The angle is chosen so as to minimize the total length of the lines, among a discrete set of angles. In practice, we hop around the circle by jumps of 5 degrees. Figure 4 illustrates the extrusion of bridges. For this model (Fawn), the number of bridges is unusually high. We believe this is the main reason why this model is longer to print in our comparisons (Table 1).

5 Implementation

One of the goal of minimizing the amount of material used inside a printed object, is to afford for larger objects to be printed in reasonable time. For complex, detailed objects spanning a large number of slices (say from 1 to several thousands) it is preferable to use a vectorial geometric description of the slices, for the complexity of a discrete, voxel-based representation grows too quickly.

We represent a slice as a set of polygons with holes whose vertices lie on the integer grid. We scale coordinates as necessary so that one unit represents 1 µm. We use the *Clipper* library to perform boolean operations of the slices [15].

5.1 Minkowski sum for dilations and erosions

One particularity of the technique presented in this paper is its strong reliance on morphological dilation and erosion. The Clipper library offers an OFFSET procedure that implements erosion or dilation by a disk. This procedure is well adapted to a single-use case: It creates new vertices so as to minimize the error in the approximated polygonal result. However, our technique applies dilations and erosions iteratively, as many times as the number of slices. In this case, the OFFSET procedure produces a very large number of vertices. This both slows the process down and dramatically increases the appearance of degenerate geometric features (small holes, long spikes, *etc.*). Instead we replace the dilation operation by a Minkowski sum with a convex regular polygon D having 24 vertices with integer coordinates, which is enough for our purpose. The advantages are twofold: First, $C \oplus D$ can be computed almost exactly with a very fast technique

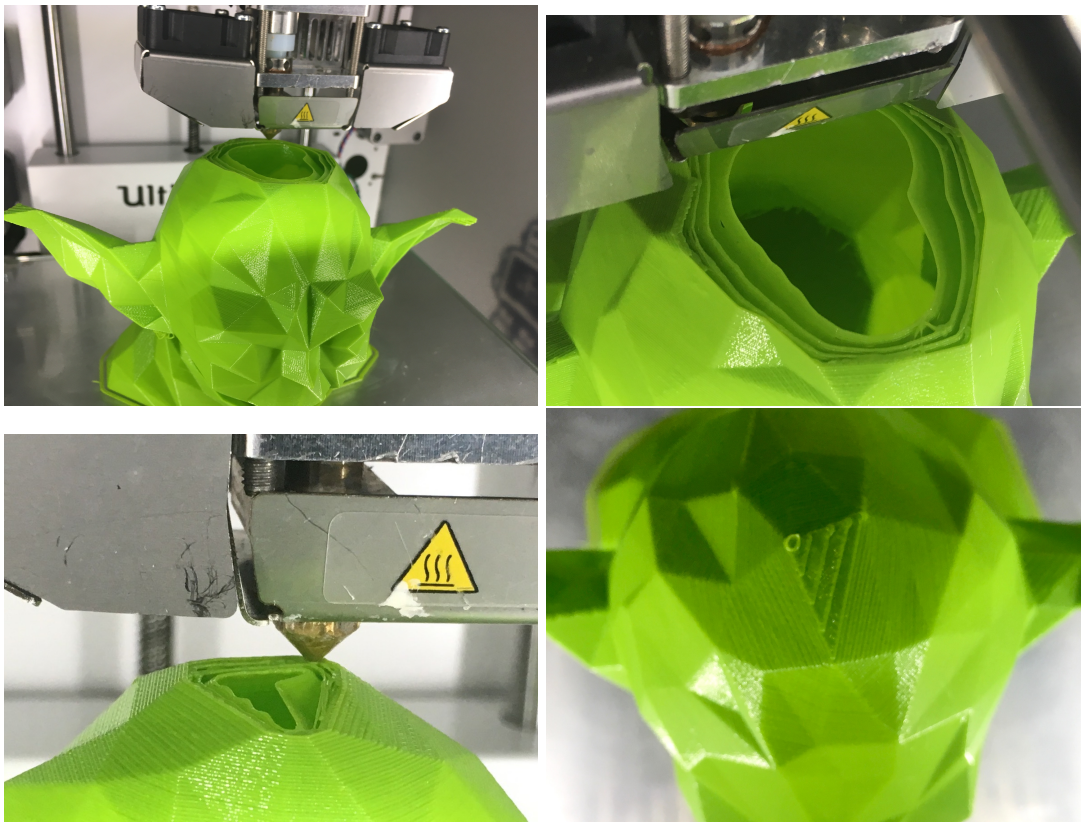


Figure 5: Timelapse of the printing of a Yoda model. *(Top-right.)* Note how the print is mostly empty and the nested cavity walls. *Bottom-left.* Approaching the top of the head. *Bottom-right.* Closing the top of the head.

(Appendix A; for dilations, approximations appear around the reflex vertices of C). Second, the operation is associative: $(C \oplus D) \oplus D = C \oplus (2D)$, so that the number of vertices on subsequent dilations or erosions never grows out of control.

5.2 Computing the seed shape

In the CAV procedure, during the top-down cavity propagation, a "central part" of each swept slice serves as a seed shape for the growing cavity. As a "central part" we use a subset of the medial axis of the input slice $O_{|z}$. This subset is obtained by pruning the medial axis using the extended distance function, as defined and studied by Liu *et al.* [20]. This particular subset of the medial axis has the advantage to be particularly robust to small perturbations along the contours of the input polygons. We can thus obtain very clean seed-shapes that permit a fast and smooth motion of the extrusion nozzle on the 3D printer (Algorithm 3).

We compute the medial axis of a slice using the *Segment Delaunay Graph* of the CGAL library [16]. The pruned medial axis is a network of curves in the plane. We linearize each curve segment of the network (parabolic arc are approximated using an appropriate number of line segments). Then, we dilate the network by a disk of diameter equal to that of the printer nozzle. (As explained above, this dilation is approximated using a Minkowski sum with a regular polygon.) The contour of the dilation is now interpreted as a print-path and is ready to be used in the CAV procedure.

Algorithm 3 SKEL computes a thin shape for seeding the growth of cavities.

```
1: function SKEL(Shape O)
2:     m ← medial axis of O [16]
3:     p ← pruned medial axis [20]
4:     return dilation of p
```

6 Results

6.1 Scalability

In this section, we process a single input model (the kitten from https://www.thingiverse.com/thing:12694) at various scales, from 1 to 10, and examine a few metrics. The number of iterations for modeling cavities is fixed at 6 so that the cavities (almost) completely cover the volume of the input model. Since the cavities are large and we only print their surface, we expect the quantity of material used to print the model to grow quadratically with the scaling factor. Figure 6 presents our measurements.

6.2 Comparison with state of the art

We compare our technique with the recent state of art work of Lee *et al.* [17]. The measurements are shown in Table 1. Seven simple but large models are used for this benchmark (Figure 7 left). The authors provided us with their GCode files for each model, which let us see that they use three covers and one shell (in addition to the perimeter). We use the same parameters for our technique in column "3c 1s." However, the volume and weight computed from the GCode files do not correspond to the weights shown in the paper and the authors did acknowledge that the GCode files differ from those use in their paper. For this reason, we report the weight of the technique "M2" as they appear in their publication. For all models but the *Fawn* our technique

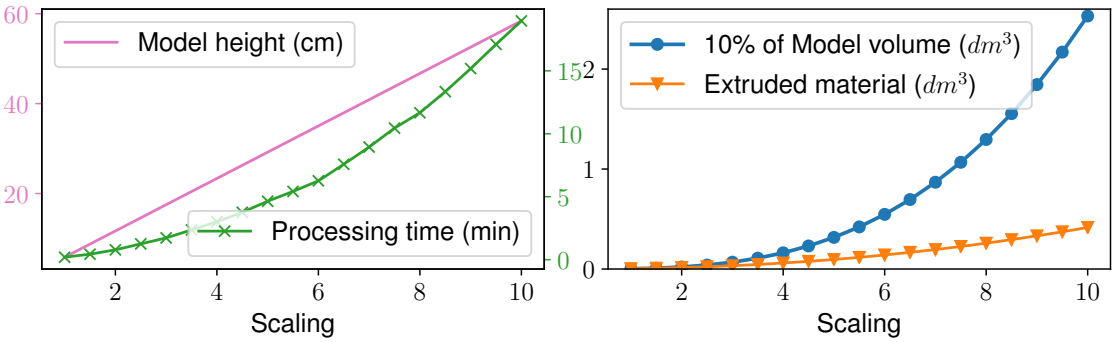


Figure 6: The horizontal axis is the scaling factor applied to the input model. *Left.* Height of the model and processing time, including loading the STL file, processing the slices and writing the GCode file. *Middle.* Volume of the input model (divided by 10 for clarity) and volume of the extruded material for printing the modeling with our cavities. *Right.* Estimation of the exponent α when approximating the curves in *Left* and *Middle* with a formula of type Rs^α where R is a constant. s is the scaling factor (in abscissa). As expected, the exponent is exactly 3 for the volume. It is just above 2 for the volume of extruded material, which indicates that the volume of our cavities walls scales more like a surface than like a volume.

Figure 7: The models used for comparing with the work of Lee *et al.* [17]. *Right.* Printing without inner shell makes the models transparent enough to let us see the inner cavities.

yields lower material usage and faster printing time. The Fawn model contains many bridges and we suspect this is the main reason why our technique is less efficient for this model. It should be possible however to improve our technique to remove many of these bridges which are not absolutely necessary.

We have also experimented with printing the models with the minimal amount of material. For this, we used just 2 layers of "cover" and no shell, so that the surface of the model is printed with a single-filament thickness (only the perimeter is printed). Results are reported in the same table, in columns "2c 0s." Because of their thinness a strong back light let us see the cavities printed inside the object (Figure 7 right).

7 Conclusion

We have proposed a technique to produce large self-supporting empty cavities inside an object. It strongly reduces the required material and time to print large objects and outperforms existing techniques. In addition, it requires a single top-down sweep through the slices, keeping only two

			Time (min.)			Volume (cm^3)			Weight (g)		
Model	Volume	Cavities	2c 0s	3c 1s	M2	2c 0s	3c 1s	M2	2c 0s	3c 1s	M2
Cat	261.1	5	183'	**223'**	240'	20.3 (7.77 %)	29.0 (11.11 %)	–	24.4	**35.5**	37.6
Fawn	387.6	6	249'	301'	**284'**	27.8 (7.17 %)	39.4 (10.17 %)	–	31.3	47.9	**45.6**
Fox	516.0	5	192'	**261'**	300'	22.0 (4.26 %)	36.3 (7.03 %)	–	26.4	**45.0**	52.7
Giraffe	355.5	5	203'	**258'**	301'	22.6 (6.36 %)	34.6 (9.73 %)	–	25.7	**42.1**	49.4
Moai	836.4	5	279'	**369'**	451'	32.4 (3.87 %)	52.4 (6.26 %)	–	36.6	**62.8**	77.1
Skull	1056.4	6	279'	**368'**	382'	35.4 (3.35 %)	53.5 (5.06 %)	–	42.6	**65.5**	68.9
Yoda	390.5	5	222'	**261'**	311'	21.5 (5.51 %)	32.7 (8.37 %)	–	24.6	**38.9**	51.3

Table 1: Comparisons with the work of Lee *et al.* [17]. "Cavities" indicate the number of iterations used to model the cavities. "Volume" in the second column indicates the volume of the input object. "Volume" indicate the volume of the material used for printing. (In parenthesis, as a percentage of the total volume of the object.) The column "M2" shows the values of the best technique in the work of Lee *et al.*. The column "3c 1s" uses a cover 3 layers thick and 1 shell (in addition to the perimeter). This is the same configuration used by Lee *et al.* so that this column is directly comparable with column "M2": The better values are typed in boldface. The column "2c 0s" uses a cover 2 layers thick and no shell.

consecutive slices in memory; it therefore scales to very large objects.

A first limitation of our approach is that it requires additional internal supports, due to the appearing local minima. It would be interesting to see whether the scheme could be adapted to reduce this issue further. The internal bridges we use as supports can also become long in large objects, while supporting large features above them. In such situations they become less reliable to print. Robust external support techniques could be adapted as internal supports instead of our bridges.

The tearing scheme we proposed in Section 3.2.2 may lead to cases where the tear front — applied within each slice — advances at a rate that produces an angle exceeding the maximum overhang. This again requires extra supports. This issue, however, can be eliminated by a tearing scheme that prevents such cases to occur, constraining the 'tear' to move by no more than $\frac{r}{2}$ between slices. Overall, the question of the best tearing scheme — the one producing the largest cavities with fewest auxiliary supports — remains open.

Finally, while we believe optimization of sparse internal supports may be able to outperform our approach in terms of material use, their computation cost quickly becomes prohibitive on large parts. In addition, the smooth, continuous outlines produced by our technique print more reliably and faster than truss networks on vector deposition techniques such as FDM.

Our technique is simple to implement, prints reliably in most cases, and provides large savings. We hope to see it adopted rapidly by slicing software.

A Fast Minkowski Sum

By leveraging the efficient implementation of the union of oriented contours in the Clipper library, we build a fast implementation for the Minkowski sum of a general polygon with a convex polygon (called the structuring element). Figure 8 describes the algorithm.

References

[1] Simone Cacace, Emiliano Cristiani, and Leonardo Rocchi. A level set based method for fixing overhangs in 3d printing. *Applied Mathematical Modelling*, 44:446 – 455, 2017.

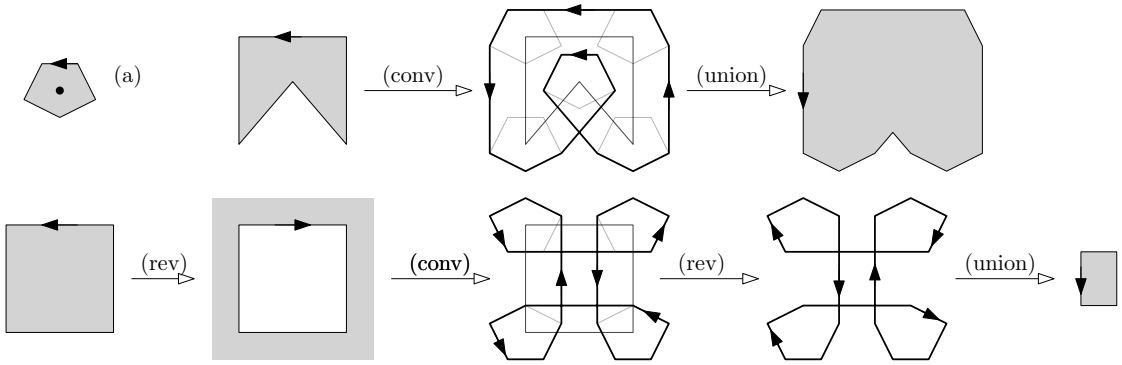


Figure 8: Computing the Minkowski sum between a general polygon and a convex structuring element. *Top.* (a) The structuring element, with its reference point. For a **dilation**, we first (conv)olve the contour(s) of the general polygon with the contour of the structuring polygon. We then compute the (union) of the convolved contours using the Clipper library. *Bottom.* For an **erosion**, we (rev)erse the contours of the general polygon (which is akin to taking the complement), (conv)olve them with then contour of the center symmetrical structuring polygon, and (rev)erse the resulting contour. Finally, we compute the (union) as above.

[2] Yong Chen. 3D texture mapping for rapid manufacturing. *Comput. Aided Des. Appl.*, 4(6):761–771, 2007.

[3] Jordan J Cox, Yasuko Takezaki, Helaman RP Ferguson, Kent E Kohkonen, and Eric L Mulkay. Space-filling curves in tool-path applications. *Computer-Aided Design*, 26(3):215–224, 1994.

[4] H. Quynh Dinh, Filipp Gelman, Sylvain Lefebvre, and Frédéric Claux. Modeling and tool-path generation for consumer-level 3D printing. In *ACM SIGGRAPH 2015 Courses*, pages 17:1–17:273, 2015.

[5] Jérémie Dumas, Jean Hergel, and Sylvain Lefebvre. Bridging the gap: automated steady scaffoldings for 3D printing. *ACM Transactions on Graphics*, 33(4):98:1–98:10, 2014.

[6] Gerald Eggers and Kurt Renap. Method and apparatus for automatic support generation for an object made by means of a rapid prototype production method, 2007. US Patent 20100228369, Materialize.

[7] John G Griffiths. Toolpath based on Hilbert's curve. *Computer-Aided Design*, 26(11):839–844, 1994.

[8] E.K. Heide. Method for generating and building support structures with deposition-based digital manufacturing systems, 2010. US Patent 20110178621 A1, Stratasys.

[9] Tim Van Hook. Real-time shaded NC milling display. In *Proceedings of SIGGRAPH*, pages 15–20, 1986.

[10] Samuel Hornus and Sylvain Lefebvre. Iterative carving for self-supporting 3D printed cavities. In *Eurographics 2018 - Short Papers*, Delft, Netherlands, April 2018.

[11] Samuel Hornus, Sylvain Lefebvre, Jérémie Dumas, and Frédéric Claux. Tight printable enclosures for additive manufacturing. Technical Report 8712, Inria, 2015. Presented at the GraDiFab workshop in Lisbon, 2016.

[12] Samuel Hornus, Sylvain Lefebvre, Jérémie Dumas, and Frédéric Claux. Tight printable enclosures and support structures for additive manufacturing. In *EG Workshop on Graphics for Digital Fabrication*, 2016.

[13] Pu Huang, Charlie C.L. Wang, and Yong Chen. Algorithms for layered manufacturing in image space. *ASME Advances in Computers and Information in Engineering Research*, 1:377–410, 2014.

[14] Xiaomao Huang, Chunsheng Ye, Siyu Wu, Kaibo Guo, and Jianhua Mo. Sloping wall structure support generation for fused deposition modeling. *Int. J. Adv. Manuf. Tech.*, 42(11-12):1074–1081, 2009.

[15] Angus Johnson. Clipper. A C++ geometric library for boolean operations on polygons. http://angusj.com/.

[16] Menelaos Karavelas. 2D Segment Delaunay Graphs. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.8.1 edition, 2016.

[17] Jusung Lee and Kunwoo Lee. Block-based inner support structure generation algorithm for 3D printing using fused deposition modeling. *International Journal of Advanced Manufacturing Technology*, 2016.

[18] Sylvain Lefebvre. 3D infilling: faster, stronger, simpler, 2015. http://sylefeb.blogspot.fr/2015/07/3dprint-3d-infilling-faster-stronger.html.

[19] Dawei Li, Ning Dai, Xiaotong Jiang, and Xiaosheng Chen. Interior structural optimization based on the density-variable shape modeling of 3D printed objects. *Int. J. Adv. Manuf. Tech.*, 83(9):1627–1635, 2015.

[20] Lu Liu, Erin W. Chambers, David Letscher, and Tao Ju. Extended grassfire transform on medial axes of 2D shapes. *Computer-Aided Design*, 43(11), 2011.

[21] Marco Livesu, Stefano Ellero, Jonàs Martínez, Sylvain Lefebvre, and Marco Attene. From 3D Models to 3D Prints: An Overview of the Processing Pipeline. *Computer Graphics Forum*, 2017.

[22] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. Build-to-last: Strength to weight 3D printed objects. *ACM Trans. Graph.*, 33(4):97, 2014.

[23] Stanislav S Makhanov and Weerachai Anotaipaiboon. *Advanced numerical methods to optimize cutting operations of five axis milling machines*. Springer Science & Business Media, 2007.

[24] Jonàs Martínez, Jérémie Dumas, and Sylvain Lefebvre. Procedural Voronoi foams for additive manufacturing. *ACM Trans. Graph.*, 35(4):44:1–44:12, 2016.

[25] Jonàs Martínez, Samuel Hornus, Frédéric Claux, and Sylvain Lefebvre. Chained segment offsetting for ray-based solid representations. *Computers & Graphics*, 46:36–47, 2015.

[26] Sara McMains, Jordan Smith, Jianlin Wang, and Carlo Séquin. Layered manufacturing of thin-walled parts. In *ASME Design Engineering Technical Conference*, 2000.

[27] Asla Medeiros e Sá, Vinícius Moreira Mello, Karina Rodriguez Echavarria, and Derek Covill. Adaptive voids. *Vis. Computer*, 31(6-8):799–808, 2015.

[28] Julian Panetta, Qingnan Zhou, Luigi Malomo, Nico Pietroni, Paolo Cignoni, and Denis Zorin. Elastic textures for additive fabrication. *ACM Trans. Graph.*, 34(4):135:1–135:12, 2015.

[29] Alexander Pasko, Oleg Fryazinov, Turlif Vilbrandt, Pierre-Alain Fayolle, and Valery Adzhiev. Procedural function-based modelling of volumetric microstructures. *Graphical Models*, 73(5):165–181, 2011.

[30] David W. Rosen. Computer-aided design for additive manufacturing of cellular structures. *Comp. Aided Des. Appl.*, 4(5), 2007.

[31] Ryan Schmidt and Nobuyuki Umetani. Branching support structures for 3D printing. In *ACM SIGGRAPH 2014 Studio*, page 9. ACM, 2014.

[32] Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. Microstructures to control elasticity in 3D printing. *ACM Trans. Graph.*, 34(4):136:1–136:13, 2015.

[33] J Vanek, JAG Galicia, and B Benes. Clever support: efficient support structure generation for digital fabrication. *Computer Graphics Forum*, 33(5):117–125, 2014.

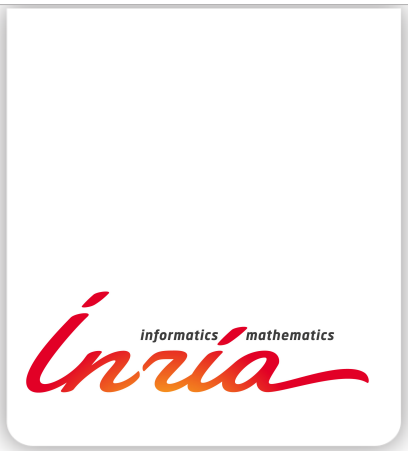
[34] Charlie C.L. Wang and Dinesh Manocha. GPU-based offset surface computation using point samples. *Computer-Aided Design*, 45(2):321–330, 2013.

[35] Weiming Wang, Tuanfeng Y Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. Cost-effective printing of 3D objects with skin-frame structures. *ACM Trans. Graph.*, 32(6):177, 2013.

[36] Jun Wu, Charlie C. L. Wang, Xiaoting Zhang, and Rüdiger Westermann. Self-supporting rhombic infill structures for additive manufacturing. *Computer-Aided Design*, 80, 2016.

[37] Xiaolong Zhang, Yang Xia, Jiaye Wang, Zhouwang Yang, Changhe Tu, and Wenping Wang. Medial axis tree – an internal supporting structure for 3D printing. *Comput. Aided Geom. D.*, 35-36:149–162, 2015.

[38] Haisen Zhao, Baoquan Chen, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, and Daniel Cohen-Or. Connected Fermat spirals for layered fabrication. *ACM Trans. Graph.*, 35(4):1–10, 2016.



RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399