Ribbed support vaults for 3D printing of hollowed objects

Thibault Tricard, Frédéric Claux, Sylvain Lefebvre

Volume xx (*200y*), *Number z, pp. 1–12*

Ribbed support vaults for 3D printing of hollowed objects

Thibault Tricard[1], Frédéric Claux[1], Sylvain Lefebvre[2]

[1]Université de Limoges, CNRS, XLIM,
[2]Université de Lorraine, Inria, LORIA, CNRS

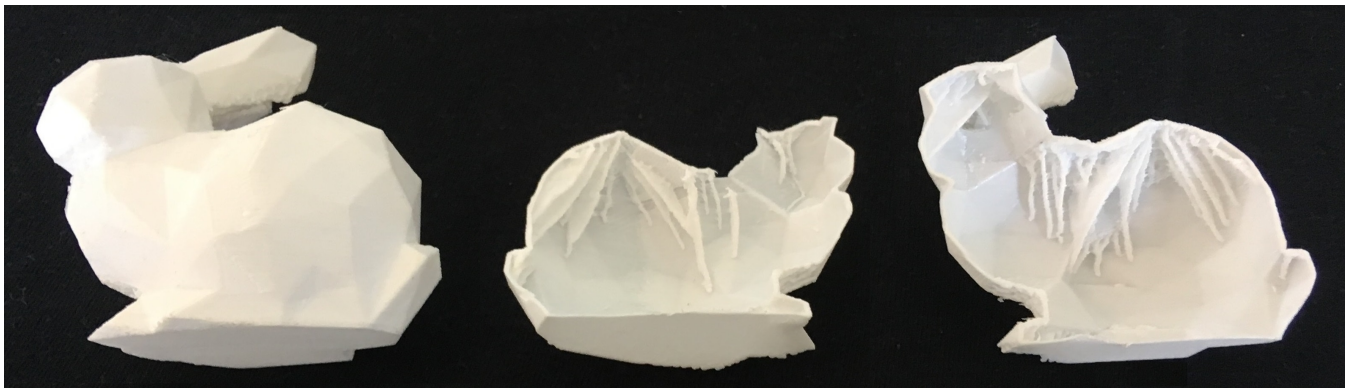

Figure 1: *Hollowed Bunny printed with our method, using only* 2.2% *of material inside (compared to a filled model).* ***Left:*** *The full model.* ***Middle and right:*** *Same model cut in half, revealing the internal support structures generated by our approach (the supports use 316mm of filament over a total of 1622mm for the print). Bunny model (size:* $54 \times 43 \times 53$ *mm). Support generation took 34 seconds. Printing time 1h 5min 30sec.*

Abstract

Additive manufacturing techniques form an object by accumulating layers of material on top of one another. Each layer has to be supported by the one below for the fabrication process to succeed. To reduce print time and material usage, especially in the context of prototyping, it is often desirable to fabricate hollow objects. This exacerbates the requirement of support between consecutive layers: standard hollowing produces surfaces in overhang that cannot be directly fabricated anymore. Therefore, these surfaces require internal support structures. These are similar to external supports for overhangs, with the key difference that internal supports remain invisible within the object after fabrication.

A fundamental challenge is to generate structures that provide a dense support while using little material. In this paper, we propose a novel type of support inspired by rib structures. Our approach guarantees that any point in a layer is supported by a point below, within a given threshold distance. Despite providing strong guarantees for printability, our supports remain lightweight and reliable to print.

We propose a greedy support generation algorithm that creates compact hierarchies of rib-like walls. The walls are progressively eroded away and straightened, eventually merging with the interior object walls. We demonstrate our technique on a variety of models and provide performance figures in the context of Fused Filament Fabrication (FFF) 3D printing.

Keywords: additive manufacturing, internal support, FDM, tree graph

CCS Concepts

• **Computing methodologies** → Shape analysis;

1. Introduction

Additive manufacturing techniques produce an object layer-by-layer, where each successive layer bonds to the one just below to form a solid object. While a wide range of technologies exist, several approaches deposit material locally, and thus the process is subject to the constraints of gravity. This is in particular the case of the very popular Fused Filament Fabrication (FFF) process where material may only be deposited at a location already supported from below, either by the printer platter or by previously deposited material at the layer immediately below. Failure to do so results in deposited matter falling down, leading to catastrophic fabrication defects.

Support structures for 3D printing are generally understood as *external* support: the necessity to support overhanging object parts so that they can be printed. In this scenario, the support structure is printed together with the object and removed after printing. The removal is usually a manual, time consuming and delicate operation. This problem has therefore been extensively studied in the literature. Reseachers propose support structures that are fast to print, minimize the quantity of deposited matter, avoid stability problems during the print and are easy to remove (Section 2).

However, support structures may also be required *inside* an object. Indeed, in the context of rapid prototyping or when mechanical resistance is not a concern (e.g. printing decorative objects), the user often seeks for a fast print, which translates to printing as-empty-as-possible. Empty inner cores translate to internal overhang areas and the related necessity to have supports for these overhangs as well.

Hollowing a part is not trivial with technologies such as FFF. In particular, the inner cavity resulting from a standard hollowing operator will not be printable: it will contain regions in overhang (with a low slope, see Figure 2) as well as *local minima*: pointed features facing downwards. There is therefore a need for support structures that can operate inside a part.

The challenges are slightly different than those of external support. Inner supports are not meant to be removed: they are trapped inside. Therefore, there is no consideration regarding ease of removal. However, they should ideally minimally impact print time and occupy little space within the cavity.

Another challenge of support structure generation is the manner

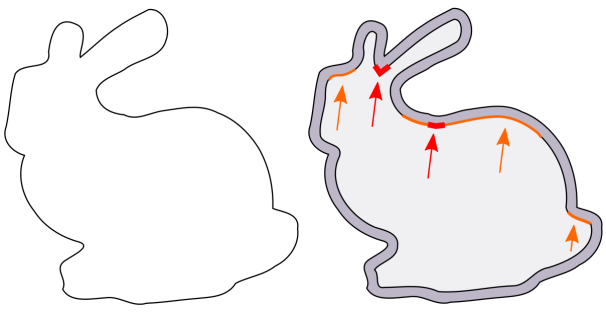

Figure 2: *A Stanford bunny model is hollowed using a standard offsetting approach. The resulting cavity, shown on the right, will not print properly on a filament 3D printer due to local minima (red) and overhanging areas (orange).*

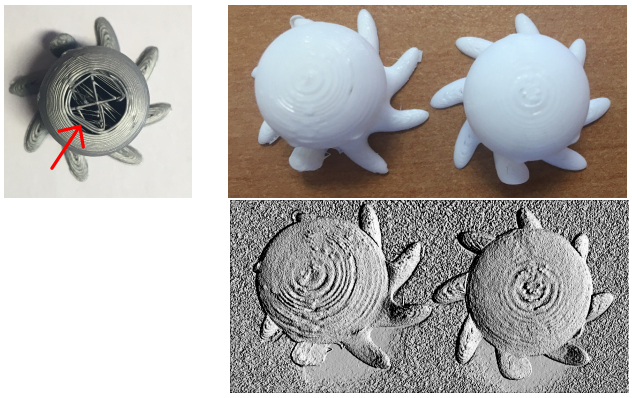

Figure 3: ***Top left:*** *Partial print of Cute Octopus. On the top layer the filament should have been deposited along circular toolpaths. Due to lack of support across infill gaps, the deposited filament does not adhere to the layer below and creates straight lines, leaving gaps in the print.* ***Top right:*** *Comparison between full prints using standard infill (left print) and our support structures (right print). By ensuring that the filament is always supported, our technique produces a higher quality surface.* ***Bottom:*** *Top right picture enhanced to highlight defects. Our method produces less artifacts. (thingiverse.com/thing:27053 by MakerBot)*

in which support structures are forming the supporting surface below the layers. This surface may be more or less dense, controlling how many support features (often points) are provided below overhanging areas. For external support, lower densities are generally preferred to ease the removal process; however this comes at the expense of residual defects on bottom surfaces. Higher densities provide a full support. These can be used inside, since removal is unnecessary, or when dissolvable material is available to print the external support structures (e.g. PVA).

It is often considered that the sparse, grid-like filling structures used inside the prints provide enough support for the roofs of inner cavities. However, Figure 3 is showing how defects – gaps in particular – can still appear. These stem from deposited filament sagging between sparse support points, and not attaching properly to the layer below. When the filament starts to be deposited over an almost empty zone, printing defects become increasingly noticeable (Figure 4). Another situation often considered acceptable is when roof layers are forming bridges: areas with outer contours are supported, even though their surface is not supported. The filament may indeed be deposited above voids in straight lines supported only at their starting and ending extremities. However, this typically requires multiple layers to limit the filament sagging and improve the print quality of topmost layers; otherwise defects remain (Figure 5). For artifact-free printing to truly take place, it is necessary to support densely, as any unsupported region, even small, would result in a downward bulge. This is aggravated when heavy material is being deposited (e.g. clay, ceramics).

Contributions: We propose an algorithm to generate internal support structures that *guarantee* that deposited material is supported everywhere from below, are reliable to print, and require little extra material. This is achieved by generating hierarchical rib-like wall

structures, that quickly erode away into the internal walls of the object. Remarkably, our method uses less material than previous works while providing a dense, reliable support everywhere.

The paper is organized as follows. Section 2 deals with previous works, reviewing existing methods for support generation. Section 3 formalizes support requirements and guarantees. Our algorithm is detailed in Section 4. Results and limitations are presented in Section 5. We discuss future work in Section 6.

2. Previous work

We review here the contributions that have been made on support structures in the context of 3D printing hollowed objects. We only focus on works relevant to ours and direct interested readers to the survey of Livesu et al. [LEM*17] for extensive details.

Sparse infills. The most straightforward method to speed up 3D prints is to use low-density infills [MSWS00]. Several infill patterns can be used [Sli] with either fixed or gradual density [Ice, Ult]. However, these infills were primarily designed to provide a good compromise between mechanical resistance and material use. They are not designed for extremely low density printing, even though users often select them in an attempt to do so. At low densities these patterns exhibit large gaps, leaving the filament above only partially supported. This leads to noticeable gaps (Figure 3). This problem can only be mitigated by using multiple covers (increasing tops/bottoms thickness) or by increasing the infill density, leading to longer print times.

Self-supported cavities. Instead, recent works explore the idea of creating large *self-supported* cavities inside models.

Several of these methods are based on an adaptive rhombic structure (angled 3D grid). Lee et al. [LL17, LLL17] use a structure of predefined resolution. In order to reduce the support size, inner grid blocks are merged together. Unprintable V-shaped features (eg. local minima) are eliminated using a block subdivision process. However, block grid cells unfortunately do not provide a dense support (see Figure 3), meaning support cannot be guaranteed within cells. The method proposed by Wu et al. [WWZW16] starts by building a coarse rhombic grid and refines it in an octree-like fashion, following a mechanical resistance criterion. Cells near model boundaries with overhang areas are refined appropriately. Xie and Chen [XC17] use a rhombic-shaped voxel representation of the object and formulate internal support generation as a combined global optimization

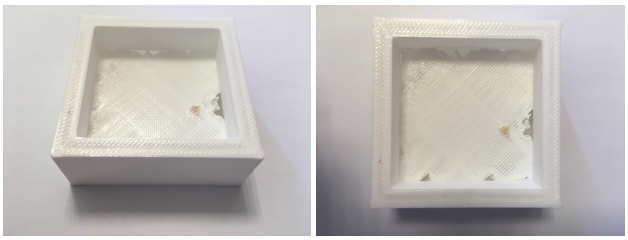

Figure 4: *Solid flat face supported by sparse infill below. The infill is too sparse which causes noticeable gaps to appear.*

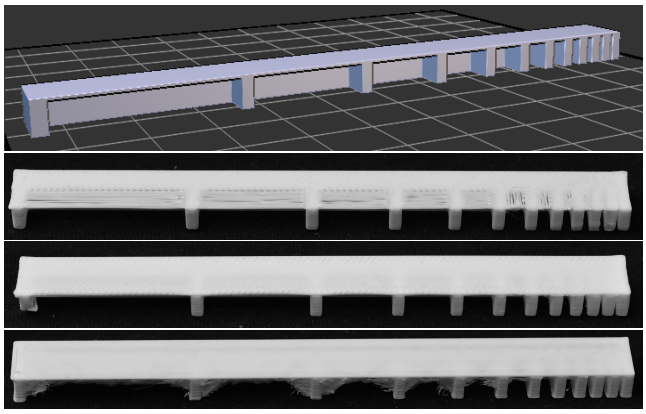

Figure 5: *Print quality comparison for the top layer of a same part (3D model shown at the top) having half the top surface printing as bridges (foremost side).* ***Top print:*** *Bridges are one layer thick and unsupported between pillars. Note the significant sagging.* ***Middle print:*** *Bridges are three layers thick and unsupported between pillars. The top surface still exhibits sagging.* ***Bottom print:*** *Using our method provides a dense support resulting in a perfectly flat top surface, even with a single top layer.*

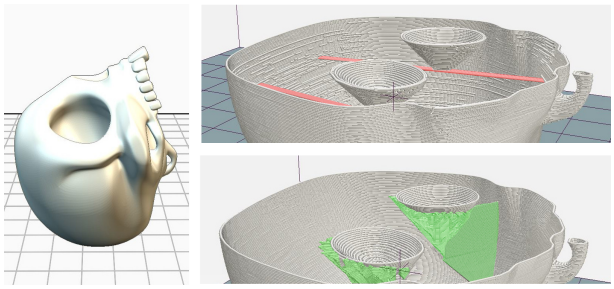

Figure 6: *3D printing a skull. On this model [HL18] produces long bridges (top right, in red) entirely supporting the eye sockets (result obtained with the implementation of [HL18]). These will be difficult to print and prone to failure. In contrast, our technique (bottom right, in green) encloses the sockets in ribbed supports firmly connected to the surrounding part walls.*

and discrete dithering processes. Their algorithm is prone to generate suboptimal, extraneous pillars for flat overhang areas. Note that, in contrast to our approach, these two latter methods can generate support-free interior voids while also satisfying user-specified functions such as mechanical stiffness, ability to spin or static stability. This is especially relevant since internal supports have an impact on the mass distribution of the object. The approach of Wang et al. [WWY*13] optimizes truss structures within parts, to reduce material consumption while preserving structural integrity.

Other methods use morphological operations on slices to carve self-supported interior spaces within the objects. The methods of Hornus et al. [HLDC16, HL18] generate a cavity within a part by initiating a tear in top slices, which then grows down as quickly as

possible within the volume. This approach is iterated in remaining pockets. The algorithm is computationally efficient and generally produces large, self-supporting cavities. Unfortunately, it is not self-sufficient: bridge supports are required in addition to the cavities themselves. These bridges support local minima (islands) that can appear in the cavities. As shown in Figure 6 they can become long while supporting large parts above, making the print prone to failure. A second issue, described in an errata by Hornus et al. [HL18] is that the morphological approach may miss overhanging edges. This is shown in Figure 7, where an edge in overhang is left dangling. In contrast our technique provides strong guarantees for printability, does not require to explicitly detect problematic cases, and often uses slightly less material.

The method of Wang et al. [WLW*18] also considers nested self-supported cavities. It segments the object interior space into simpler volumes, within which cavities are grown under self-support constraints. Cavities are iteratively added in remaining pockets. Since the cavities are restricted to their volume segments they cannot join and form local minima. The growth is driven by an optimization process which allows for auxiliary objectives such as optimizing for balance. This added flexibility however comes at the price of a computationally intensive approach that does not scale to large parts and high resolutions. We provide direct comparisons with results produced by this approach in Section 5.

The method of Zhang et al. [ZXW*15] is also related to our work, albeit it focuses on structural properties. It uses a 3D medial axis to generate a sparse, pillar based structure inside a 3D object. The algorithm is designed to ensure the printed object resists external stresses. As the method does not consider FFF printability constraints it requires auxiliary supports when fabricating on FFF printers.

External supports as internal structures. External support structures may be used within hollowed cavities, although they have not been primarily engineered with this goal in mind.

The typical external supports are volumes extruded downward from overhanging surfaces, and are therefore quite large. To reduce their size, Huang et al. [HYW*08] shrink their sections in the middle, producing support volume with slanted sides. Recent methods go further and create sparse pillar tree structures to support an object during its fabrication [SU14, VGB14]. The method of Dumas et al. [DHL14] generates stable scaffolds made of cross-shaped pillars and flat bridges. While very efficient as external supports, these techniques suffer two main limitations when considered as internal supports. The first is that pillars are actually slow to print due to the decelerations required to print each circle within a slice. Angled pillars are also subject to thermal warping and torque. The second is that, when using a high support density, the structures start fusing resulting in complex, slow to print geometries below the overhanging surfaces (Figure 8). This can be alleviated by using a sparser set of support points, but this tradeoff may again result in gaps on surfaces above.

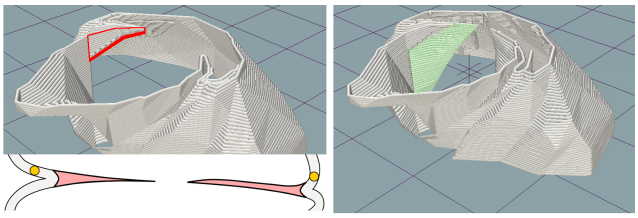

Figure 7: *Inside a Yoda model.* ***Top left:*** *The iterative carving method [HL18] misses an internal edge in overhang, outlined in red.* ***Bottom left:*** *Illustration of the problem. Sharp edges (red) are erased during morphological operations (dilation by orange disk) and entirely disappear. Please refer to the errata of [HL18] for details.* ***Right:*** *Our method correctly adds support below such edges in overhang (in green), while still using less material overall.*

Our algorithm produces structures offering a very high support density, while using little extra material. In addition, our supports print reliably as they are composed of continuous, wall-like structures that suffer less from stability issues. We compare our method to the state of the art in Section 5.

3. Supporting a shape

We formalize in this Section the well known criteria for supportability in the context of our work. We start by introducing our notations and then discuss the constraints and properties of supports for the fabrication of 3D models.

3.1. Notations

Given a 3D object $O \subset \mathbb{R}^3$, we define B as the outer boundary of a hollow version of O containing only the external shell of the object. Note that B can be obtained immediately when using meshes. B can be decomposed into I slices for printing, with B_i referring to the i^{th} slice. Slices B_i are to be understood as exact, geometrical cross-sections of B with Z-axis aligned planes. Slices are numbered from bottom to top, meaning that B_{i-1} identifies the slice below B_i.

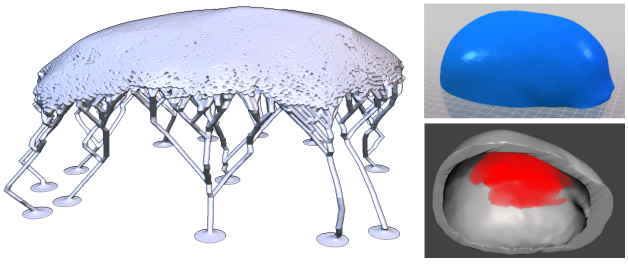

Figure 8: *Pillar structure supporting an hollowed skull top model (top right). The red areas along the cavity roof (bottom right) cannot print without support. The pillars are generated by Autodesk Meshmixer (Schmidt and Umetani [SU14]). We set support density to match our support requirements as defined in Section 3.2. Note how pillars fuse below the overhanging surface producing a compact, slow-to-print geometry. Note that the pillars are also quite unstable and subject to torque (see Figure 2 of [DHL14] for details). Model: www.thingiverse.com/thing:4378 by ssd.*

We define S as the internal object support, and its corresponding slices S_i which identify supports that are printed on layer i. Paths on layer i support the paths on the layer above, $i+1$. S_i supports the paths of both B_{i+1} and S_{i+1} that are not already supported by B_i. For simplicity, the paper sometimes omits the union ($\cup$) sign when referring to composite sets, for instance $B \bigcup S$ may be referred to as BS, and related slices to BS_i. U_i identifies the unsupported areas of layer i.

We denote by r the nozzle radius, K a disc kernel with a radius of r, K_p the K kernel centered on the point p, and h the layer height.

3.2. Supporting a slice

Supporting a 3D object can be achieved by sweeping through its slices from top to bottom, looking for unsupported parts within each slice, and adding material below these parts in the next slice. The process goes on until the bottom slice is reached, where material is supported by the printer platter.

The material that is added below unsupported parts does not have to cover the whole unsupported area. For a slice to be fully supported, it is generally considered that every single point $p \in \mathbb{R}^3$ on that slice needs to have material in a vicinity of the nozzle radius in the slice below.

$$\forall p\{x,y,z\}\, \exists p'\{i,j,z-h\},\ \sqrt{(i-x)^2+(j-y)^2} \leq r \qquad (1)$$

where h represents the layer height used for the print. This guarantees that the whole filament will be supported with a ratio of 50% of material support, which is sufficient for artifact-free printing. This ratio can be adjusted by the user.

As long as equation (1) is honored, support material can take any shape. Material in one slice can either be smaller or larger than the one in the above slice for support to be effective.

Intuitively, support material can be added immediately below overhanging parts and then undergo a *reduction* operation from slice to slice going downwards to limit the quantity of deposited material. Reduction operators are covered in Section 4.1.

The amount of material added can also be *larger* than the area needing support. Depositing more material than necessary comes at the price of longer printing times, but can be interesting to significantly improve printability. Large, simple support structures often are faster to print than complex, smaller structures. Indeed, when multiple disconnected locations need to be supported, it is in many cases more effective to print a single, large structure. It encompasses and conservatively supports many small locations. This is more effective than supporting isolated spots, which individual support size may be very small and therefore difficult to print, and which will inevitably increase the amount of travel and therefore print time (taking nozzle acceleration and deceleration into account). For instance, Hornus et al. [HLDC16] build on this idea to reduce the geometric complexity of sparse support structures (see their Figures 10 and 11).

Support requirements may here be formulated using morphology, whereby the boolean difference between the upper and dilated lower slices should give an empty set (equation (2)). BS_i has to be contained in slice $BS_{i-1}^{r\uparrow}$, with $r\uparrow$ defining a circular morphological dilation of S_i.

$$BS_i \setminus BS_{i-1}^{r\uparrow} = \emptyset \qquad (2)$$

4. Our approach

From the discussion in Section 3 we derive a general principle described in Algorithm 1. We then instantiate this general principle to define our ribbed support vault structures.

General principle. The idea behind our algorithm is to produce supports through three main operations, during a top–bottom sweep through the model slices: 1) propagation and reduction of the supports from the slice above (Section 4.1) 2) detection of appearing unsupported areas in the current slice (Section 4.2) and 3) addition of new supports required for the current slice (Section 4.3).

Model B and support geometry S are kept separate during the whole algorithm. Once S is fully generated, B and S are printed together. The general algorithm is presented in Algorithm 1. The main loop visits the slices from top to bottom, generating a support slice S_i from the previous slice S_{i+1}. We denote by T_i the propagated reduced supports and denote the propagation-reduction operator as $\rightarrow$.

Algorithm 1 General Algorithm

```
function GETSUPPORT( B, r)
    I ← B.height
    S_{I-1} ← ∅
    for i = I-2 down to 0 do
        T_i ← (S_{i+1} → B_i) // Section 4.1
        U_{i+1} ← BS_{i+1} \ BT_i^{r↑} // Section 4.2
        S_i ← getSliceSupport(U_{i+1}, B_i, T_i) // Section 4.3
    end for
end function
```

Ribbed support vaults. We instantiate this general principle to produce support structures made of hierarchies of rib-like walls.

Our inspiration comes from architecture, where supports are generally designed in an arch (and vault) like manner. In particular, vaults tend to join walls in any interior space, with only a few straight pillars directed towards the floor. Similarly, many vault structures present hierarchical aspects. Such hierarchies afford for dense supports while quickly reducing to only a few elements – much like trees.

Within each slice we favor supports having a rectilinear aspect: they provide support all around them while eroding quickly from their ends. Thus, within a given slice, we seek to produce rectilinear features covering the areas to be supported.

We propose to rely on 2D trees joining the object inner boundaries. Through the propagation-reduction operator, the trees are quickly eroded away (from their branches). Taken together across slices, the trees produce self-supported walls that soon join and merge with the object inner contours, much like the ribs of ribbed vaults. The principle is illustrated in Figure 9. Figure 10 reveals the inside of a

Kitten model split in half. The hierarchical aspect within each slice, together with the reduction moving downwards leads to very small linear structures.

Note that all our supports are permanently connected to B – this exploits the fact that internal supports do not need to be removed and remain hidden inside the parts.

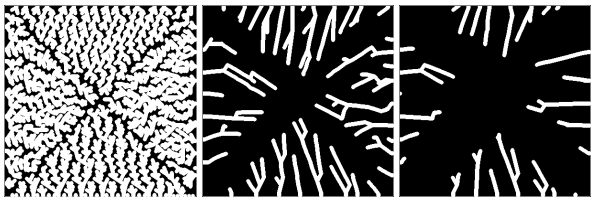

Figure 9: *Three slices (BS_i) for a hollowed cube (from higher to lower). The initial support trees quickly merge into the outer walls.*

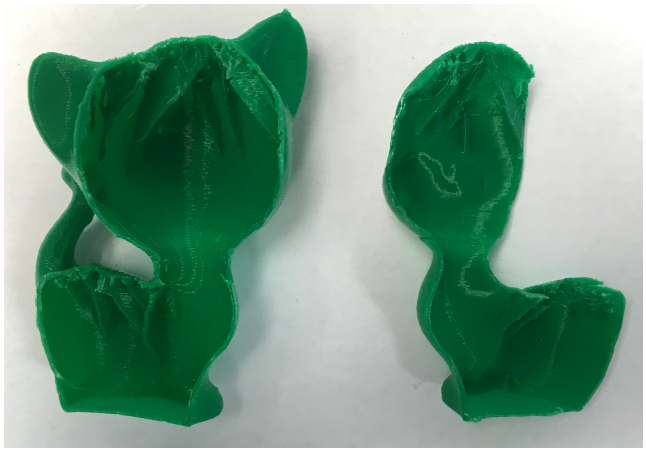

Figure 10: *Hollow kitten model printed with our method and split in half vertically. thingiverse.com/thing:12694 by MBCook*

4.1. Propagating supports down from the previous slice

Our algorithm starts by first propagating support geometry from a slice to the next one downwards. For now, let us assume we have a support slice S_i. We obtain S_{i-1} in two passes.

Copy. The copy operation first adapts S_i's shape to the boundaries of B_{i-1}. Tree trunks may be clipped so that they never cross slice boundaries, or may be extended to join slice boundaries. This is illustrated in Figure 11.

Reduction. Once support trees from the upper slice have been copied over to the lower slice and connections with the lower slice boundaries have been restored, we can deform trees in a way that is compatible with equation (1). Reducing their length ensures less material will be used when printing and that material geometry will eventually join B and vanish.

First, we erode all extremities of support trees by a ratio of r.

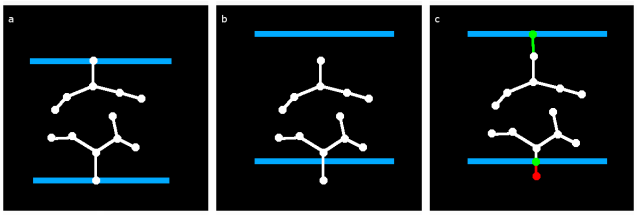

Figure 11: *Propagating S_i down to the lower slice. Projecting and reconnecting the graph on to the i-1 slice. (a): S_i and B_i, (b): S_i and B_{i-1} (c): T_{i-1}, which is a modified version of S_i fitting B_{i-1} (green: prolongated tree parts, red: clipped tree parts)*

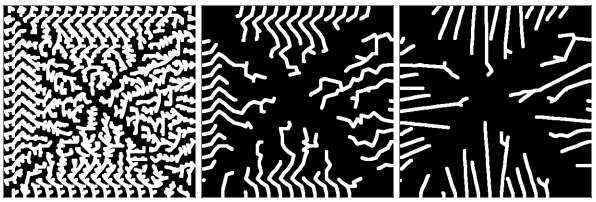

Figure 12: *Effect of the straightening operator on the supports within a cube. Left: second slice. Center: 12^{th} slice with tree straightening disabled. Right: tree straightening enabled.*

Then, we *straighten* the trees so as to reduce their overall length. Only branch-free, purely linear portions of trees are straightened. As erosion progressively erases branches from one slice to the other, our straightening operator has longer and longer segments to work with.

Let's define a linear portion of a tree as a set of point $p_i \in \mathbb{R}^2, i \in \{0,n\}$.For each point p_i we calculate the rectilinear distance D_i between p_i and p_0 defined as $D_i = \sum_1^i \|\overrightarrow{p_{i-1}p_i}\|$. We then define $d_i = \frac{D_i}{D_n}, 0 \leq d_i \leq 1$. We use d_i to determine p_i^e, the equivalent of the point p_i on the segment $\{p_0, p_n\}$. We move p_i towards p_i^e using the formula $p_i \leftarrow p_i + min(r, ||\overrightarrow{p_i^e p_i}||) * \overrightarrow{\overrightarrow{p_i^e p_i}}$. Note that r limits the displacement of p_i, and consequently that equation (1) holds.

Tree straightening is illustrated in Figures 13 and 14. Overall, it greatly reduces the amount of printed material as can be observed in Figure 12. Material quantities with and without tree straightening are compared in Section 5.

4.2. Detecting points to support within a slice

Now that supports generated for upper slices have been propagated downwards, we need to add new support for the current slice BS_i. Unsupported points U_i of BS_i are defined as

$$\{p \in BS_i : K_p \cap BS_{i-1} = \emptyset\} \tag{3}$$

where K_p identifies a disc of radius r at location p.

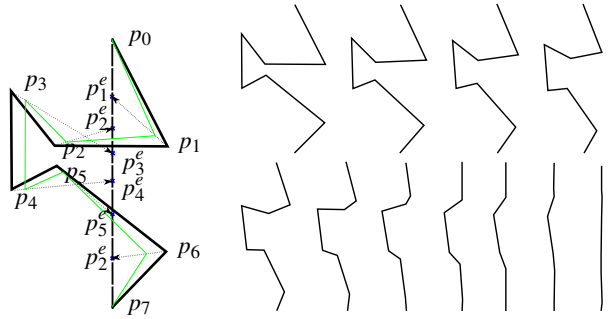

Figure 13: *Progressive straightening of a linear sequence of points.* ***Left****: The original polyline vertices are projected onto the target segment joining the first and last point of the polyline. The target location for each vertex is calculated using the rectilinear distance between the original vertex and the start of the polyline.* ***Right****: the shape of the polyline at each transformation step. Each vertex is iteratively moved by a distance equal to the nozzle radius.*

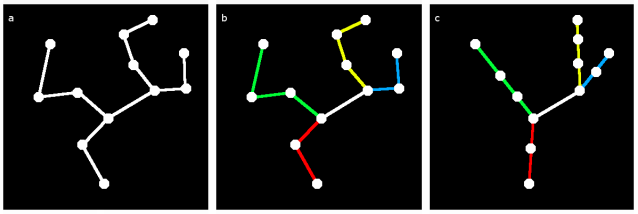

Figure 14: *Straightening of a tree. (a): original tree (b): linear tree sections (c): final tree once all sections have been straightened. Branching locations are not affected by the straightening.*

4.3. Adding support within a slice

Once unsupported points have been detected in BS_i, material needs to be added into the slice S_{i-1}, located below. Any shape can be used as long as equation (2) is verified.

For each of the unsupported points u of U_i, if u is not yet supported (equation (1)), we create a wall by drawing a straight line between u and the closest point on BS_{i-1}. The priority with which unsupported points are processed depends on their distance relative to B_{i-1}. A distance field D_B, calculated once for each slice, is used for this purpose (Figure 16). When multiple candidates with the same distance to D_B are available, we pick a candidate randomly. Having such a priority scheme tends to make the algorithm process unsupported points that are close to B first, favoring the creation of web-like structures, moving inwards starting from B.

Favoring branches over new tree trunks. As unsupported points located further away from B are processed, we encourage lines of S to merge to one another and form trees, using a heuristic. Existing supports are vastly reused this way. The reduction operators presented in Section 4.1 are very efficient when applied simultaneously to many branches on the same slice. We now describe our tree creation heuristic defining how a line segment should connect to BS.

For each unsupported point p, we locate the closest point p_b on B and associated distance $d(p_b)$ (equation (4)) and do the same for S.

$$d(p_b) = \|\overrightarrow{pp_b}\| - (0 < valence(p_b) < 4 \; ? \; 4r : 0) \qquad (4)$$

The retained point and p will then form a support segment in S. For S, the associated distance field is a function taking junction points into account. Junction points deform the distance field so as to shorten the distances towards them (Figure 16, right). As junction points are being created, they tend to attract unsupported points in their proximity. When their valence reaches 4, they no longer bend the distance field, to encourage the creation of junctions at other locations (Figure 15). Junction points with a too high valence may become more complex to print, as discussed in Section 4.4. We note D the distance field to $B \cup S$ and note $Proj_{D_{BS}}(u)$ the projection of a point u onto the zero-set of D_{BS}.

This heuristic significantly reduces the amount of material necessary to support a slice by favoring the creation of branches over new trunks. A comparaison of support generation with and without the branch heuristic in shown in Figure 17 and in Table 4. The full tree generation method is detailed in algorithm 2. An illustration is shown in Figure 18.

4.4. Printing support trees

To print support trees, we print trunks first, then branches, as illustrated in Figure 19. Likewise, when branches have sub-branches, parent branches are printed before child branches.

We avoid excessive extrusion at junction points by making sure only the first branch is printed through the junction point. Other branches do not reach the junction and stop before – taking into account extrusion width so that the deposited tracks are in contact

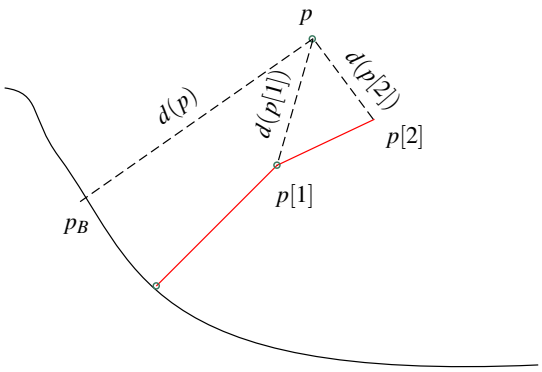

Figure 15: *Black curve: B. Red lines: S. $p[1]$ and $p[2]$ are points of S. p needs to be supported. The distance from p to B ($d(p_b)$), from p to $p[1]$ ($d(p[1])$), and from p to $p[2]$ ($d(p[2])$) are computed. Because the valence of $p[1]$ is 2, we have $d(p[1]) = \|\overrightarrow{pp[1]}\| - 4r$ (equation (4)). In the end, a support line is created from p to $p[1]$ even though the euclidean distance to $p[2]$ is smaller.*

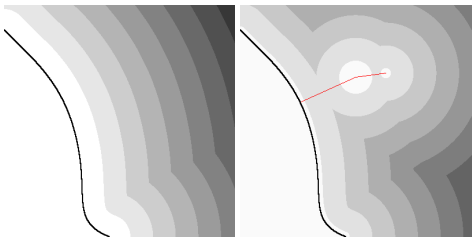

Figure 16: ***Left****: distance field D_B used to prioritize U candidates.* ***Right****: distance field D_{BS} used to add new supports.*

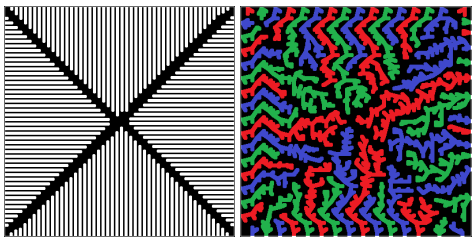

Figure 17: *Support tree generation for a flat, square slice.* ***Left****: without branch heuristic.* ***Right****: with branch heuristic. The branch heuristic favors the creation of trees, affording for more efficiency of the reduction operators used when moving from the upper to the lower slice (Section 4.3).*

but do not overlap. There is one exception to this rule when one branch is just starting. In such a case it extends to the junction to ensure it sticks to the main branch. This is explained in Figure 20.

5. Results and comparisons

In this section we present the results obtained with our method and compare them to the state of art. In all comparisons our models are printed with a layer height of 0.2 mm, one perimeter, no shells and no cover, using a 0.4 mm nozzle.

We printed our models on a variety of printers. Orange models have been printed on an Ultimaker 3 using PLA plastic; white models on a Prima P120 using PLA; the blue Buddha on a eMotion Tech MicroDelta Rework; the giant octopus on a CR-10 printer; the yellow Moai on a Ultimaker 2 printer; the dual-color Fawn on a Flashforge Creator Pro.

All the models used in the paper are listed in Table 1.

5.1. Print quality

The quality of prints using our method is indistinguishable from the quality obtained with a dense infill. This is thanks to the full support property of our structures. Figure 3, right, compares a model printed with a standard honeycomb infill and our method. Figure 5 compares top, flat surfaces printed with and without our supports. Other results are visible in Figures 1, 10, 21 and 22.

Our algorithm generates many small segments that need to be individually printed, leading to a high number of retract/prime operations surrounding travels. Depending on the printer model used, the quality of the extrusion mechanics, the user-adjustable pressure of the dented extrusion wheel on the filament, as well as the brand of the filament itself, a small amount of under-extrusion may happen. Indeed, the extrusion axis going back and forth many times tends to slightly shift the filament upstream, generating a tiny but noticeable loss of plastic[†]. To compensate for this, we perform a 5% prime surplus at the beginning of each support segment: if the filament was retracted by 3 mm before travel, we push it back by 3.15 mm after travel. Because the extra prime may create a bulge, we avoid doing it when located too close to perimeters, so as to not impact surface quality.

Unlike pillars and bridges, linear supports can be printed at relatively high speed. The advantage stems from the possibility to extract long continuous print paths from the tree structures. Models print well with our method regardless of the selected layer height, with no noticeable printing artifacts (Figure 22).

Algorithm 2 Generate the support for a slice

function GETSLICESUPPORT(U, B, T)
  $D_{BT} \leftarrow DistanceField(B \cup T)$
  $D_B \leftarrow DistanceField(B)$
  for each $u \in U$ sorted by increasing distance to B w.r.t D_B **do**
    if $K_u \cap BT = \emptyset$ **then** // if u is unsupported
      $L \leftarrow LineSegment(u, Proj_{D_{BT}}(u))$
      $T \leftarrow T \cup L$
      $D_{BT} \leftarrow DistanceField(B \cup T)$
    end if
  end for
  return T
end function

Model	Thingiverse ID
Low Poly Bunny (Figure 1)	645740
Cute Octopus (Figure 3 and 21)	27053
Skeleton (skull only) (Figure 6)	644370
Low Poly Yoda (Figure 7)	906951
Hollow Skull Top (Figure 8)	4378
Kitten (Figure 10)	12694
Owl (Figure 21)	18218
Buddha (Figure 21)	329057
Low Poly Moai (Figure 21)	908062
Fawn (Table 3, Figure 27)	906692
Low Poly Skull (Figure 22)	906562
Yoda (Figure 23)	276994
Demon Dog (Figure 26)	67935

Table 1: *Models used in the paper. They can be reached by using the following URL: http://thingiverse.com/thing:ID, where ID holds the thing identifier.*

5.2. Material quantity

We now evaluate the quantity of material required by our approach,

[†] With 1.75 mm diameter filament, a 0.4 mm nozzle and a 0.2 mm layer thickness, a shift of 17μm of the filament leads to a 0.5 mm gap in deposition.

	Input		Support-Free Hollowing		Iterative Carving		Our method	
Model	Full volume	Height	Output volume	Reduction	Output volume	Reduction	Output volume	Reduction
Demon dog	94779	82	36439	61,55%	8900	90,61%	7098	92,51%
Owl	21484	59	8780	59,13%	2764	87,13%	2374	88,94%
Yoda	101526	69	33626	66,87%	8409	91,71%	6994	93,11%
Skull	158575	64	-	-	9919	93.74%	9128	94.11%
Moai	51205	73	-	-	4493	91.22%	3663	92.84%
Kitten	15596	50	-	-	2400	84.61%	2109	86.47%
Low poly bunny	34160	55	-	-	3882	88.63%	3901	88.58%
Cube	8000	20	-	-	954	88.07%	1248	84.40%

Table 2: *Comparison with Support-Free Hollowing [WLW*18] and Iterative Carving [HL18]. The input volume represents the volume (in mm^3) and the height (in mm) of the model. For each method we present the volume of the model after hollowing, and the volume reduction in percentage* $(1 - \frac{output volume}{input volume}) \times 100$. *Please refer to the text for discussion.*

compared with both iterative carving [HL18] and support-free hollowing [WLW*18]. Results are summarized in Table 2. The results we compare with were provided by the authors of these methods.

Compared with the method of Wang et al. [WLW*18] our method uses significantly less material for a same setup (45° angle for cavities matching our 50% material overlap rule at 0.2 mm layer thickness). For instance, on the Yoda model our technique incurs a 91.71% reduction versus 66.87% for support-free hollowing. This can also be seen in the visual comparison in Figure 23. Note how our technique produces thinner structures printed in each layer with exactly one deposition path. Note that support-free hollowing [WLW*18] outputs a mesh and has to ensure sufficient wall thickness between cavities for the slicing process to operate correctly. The slicer then creates two deposition paths for each wall, which incurs an additional overhead. Our approach has an inherent advantage by directly producing deposition paths. Our algorithm is also significantly faster (discussed in Section 5.3).

Compared with the method of Hornus et al. [HL18] our algorithm uses either a similar quantity of material or slightly less. In addition, a closer inspection reveals that our method consistently provides denser support, *even when using less material*. This is a direct consequence of iterated carving missing internal edges in overhang, see Figure 7. In addition, our approach is much more reliable with respect to large isolated dangling features. These produce long fragile bridges with iterative carving, see Figure 6.

Figure 24 shows the results of our technique as the object size scales up. The last point on the curve is the 15 cm owl shown in Figure 21 (Top Left). This is especially interesting as our method is memory efficient (single sweep) and allows to process large models. To highlight the benefits of this reduction, let us take the example of a Moai model shown in Figure 21 (Bottom Left). When scaled five times, using a 15% standard infill would require 419 *meters* of filament – more than the standard 330 meters of a 1Kg PLA roll. With our approach it requires only 46 meters of filament: multiple prints fit easily a single roll. Figure 21 (Bottom Right) is an example of a print that maximizes the print bed utilization of a printer, here a CR10. The model is $273 \times 270 \times 186$ mm and uses 68.5 m of filament, where a 15% fill would require 278.2 m.

Table 3 reveals how the support size is impacted by layer thickness. The required material quantity quickly decreases when using thinner layers, which increases the overhang angle. This is true for most techniques taking into account the maximum overhang angle.

Model	Total volume (mm^3)	0.2 mm	0.1 mm
Fawn	56151.57	83.00%	87.31%
Yoda	12176.73	92.84%	94.64%
Skull	76286.19	94.11%	96.25%
Moai	51205.01	92.84%	93.49%

Table 3: *Percentage of volume reduction for different layer heights with a 0.4 mm nozzle (higher is better).*

Finally, Table 4 reveals the significant benefit of the straightening operator (Section 4.1) on material quantity.

5.3. Implementation and performance

Our method requires a single sweep from the top slice to the bottom slice of the model. Only two slices (S_i and S_{i-1}) need to be kept in memory at any given time. This is similar to the method of Hornus and Lefebvre [HL18] and provides scalability to large models.

Our framework is based on discrete slices at high resolution (typically 0.05 mm per pixel), which enables support for implicit functions, 3D textures and CSG constructs. Overhang detection is done on a pixel basis using slice bitmaps. The distance field to *BS* (Section 4.3) is reconstructed by going through each individual pixel classified as overhang and by calculating the closest distance to a discrete representation of *BS* in the inferior slice. We do however maintain a vector representation for *S* as our ribbed support walls originally come to life as line segments. Our erosion and straightening operators use this vector representation, which directly provides toolpaths for *S*.

Processing times are listed in Table 5. Note that our prototype im-

Model	With	Without
Cube	132.31 mm	165.91 mm
Skull	461.15 mm	719.78 mm
Moai	87.68 mm	149.10 mm

Table 4: *Effect of the straightening operator on filament length.*

plementation, even if not optimized and running on a single thread, is already orders of magnitude faster than the optimization based approach of Wang et al. [WLW*18]. For instance, we produce supports for the 50 mm tall Kitten in 40 seconds (single CPU thread), versus 31 minutes in their approach. Significantly improved performance can be expected if our algorithm is entirely implemented using vector geometry for both S and B, eg. when dealing with mesh models only.

Model	Time (s)	Model	Time (s)
Cute Octo	4	Fawn	9
Cube	4	Yoda	11
Skull	79	Giraffe	61
Moai	39	Waving Groot 25	7
Waving Groot 50	55	Bunny	34

Table 5: *Computation times in seconds for support generation.*

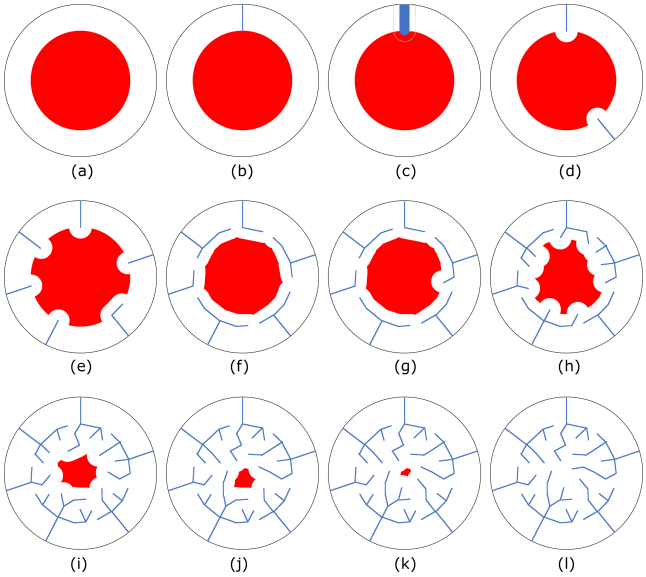

Figure 18: *Overall pipeline. (a): a slice of the outer model boundary B (black) and its unsupported area U in red. (b): a location p_u is picked on U. Location selection is sorted by distance relative to B. When multiple locations with equal priority can be selected, one is chosen at random. The point closest to p_u on either B or S is found by using the distance field defined in Section 4.3. Here, the closest location is on B, as S is empty. A line (in blue) linking p_u to that location on B is therefore created. (c): the deposited filament (in blue) and its corresponding support area. (d)-(f): other locations are picked on U and additional support lines are created. Locations on the outer boundary of U are again considered first. Their random selection tends to create support lines towards B, away from S. (g)-(l): inner locations are picked on U. Support lines link these locations to S, as S is now always closer than B. The process goes on until U is empty (l).*

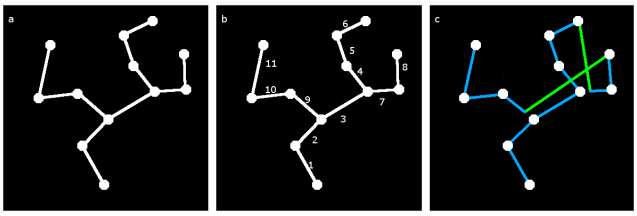

Figure 19: *Print tree ordering. (a) support tree, (b) printing order, (c) printing path (blue: deposited material, green: travel only)*

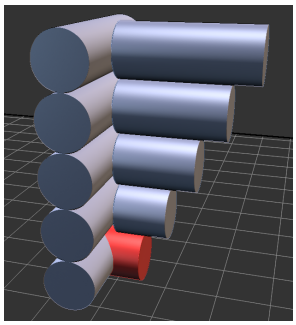

Figure 20: *Side view of two printed walls at a junction. Note the progressive, downward erosion of the right part. At a junction between walls we take special care to avoid over-extrusion. Child walls do not reach the parent wall but stop before, taking into account extrusion width. Only the first layer (in red) starts from the actual branching point to allow some overlap and ensure adherence.*

5.4. Limitations

While producing supports of small length, our algorithm is clearly not optimal. This is revealed for instance on low-angle overhangs, see Figure 25. The inefficiency is due to the local choice of connecting support walls to the closest internal surface, ignoring the material quantity that will have to appear in slices below. While a more global scheme could be devised, it could quickly become prohibitively expensive to compute.

Similarly, in some specific cases pillars could be better than walls for supports. For instance, a V-shaped roof produces a tip far away from inner walls, requiring a wall extending all the way from the side to reach the tip. This is illustrated in Figure 25. Generating a pillar in these cases could further reduce the amount of material used. However, as explained in Section 2 pillars must be printed with great care, whereas our structures print reliably.

Our method is designed to provide supports that respect the requirements detailed in Section 3.2 and does not take into account the weight of the parts requiring supports. Weight-related problems may arise when printing large, heavy objects. We however never observed the issue, even when printing the large CuteOcto model (Figure 21).

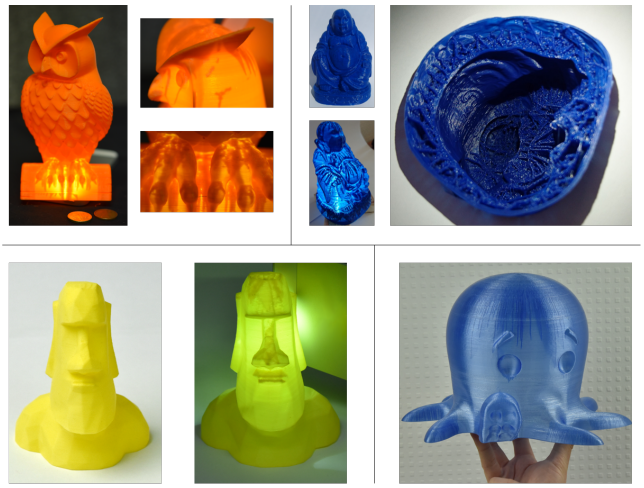

Figure 21: ***Top Left*** *: 15 cm owl printed with our method. A strong back light reveals the inner structures generated by our approach.* ***Top Right****: 88 mm Buddha model printed using our method, with no shell or cover, and using a print speed of 70 mm/s for supports and 20 mm/s for perimeters. The input model has 734,947 vertices and 1,469,760 triangles. Support generation took 107 seconds. https://www.thingiverse.com/thing:329057 by Geoffro.* ***Bottom Left*** *: Moai printed with our method, a strong back light reveals the inner structures generated by our approach. Moai model: thingiverse.com/thing:908062 by slavikk* ***Bottom Right*** *:Large CuteOcto printed with our method using transparent filament (273 mm × 270 × 186 mm).*

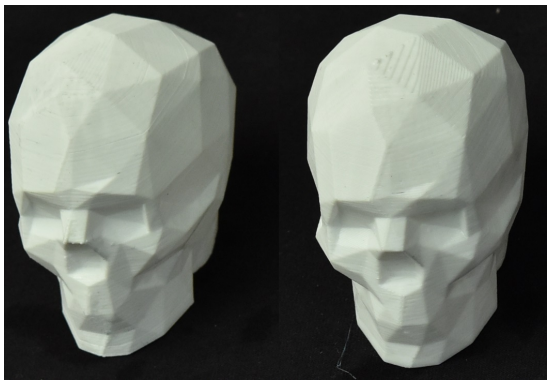

Figure 22: *Two skulls printed with our method using different layer heights.* ***Left****: 0.1 mm,* ***Right****: 0.2 mm. Corresponding material usage is shown in Table 3.*

6. Conclusion and future works

We have presented a method to print hollowed objects through the generation of internal support structures akin to ribbed vaults. Our structures use little material while providing a dense support, enabling reliable, high quality printing at a fraction of the time and cost normally required.

Despite producing smaller structures than prior works, our algorithm ensures that deposited material is completely supported. Beyond thermoplastics, we believe this property to be especially important when extruding heavy or viscous materials such as clay or concrete (*contour crafting*).

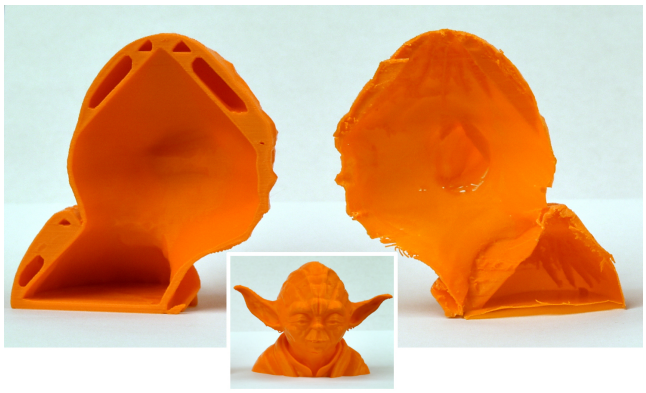

Figure 23: *Printing the Yoda model.* ***Left*** *: Hollowed by support-free hollowing [WLW*18]. **Right** : Printed with our method and cut open with a blade (hence the raw outline).* ***Center*** *: Result before cut-out. We did not add external supports which explains small defects around the ears.*

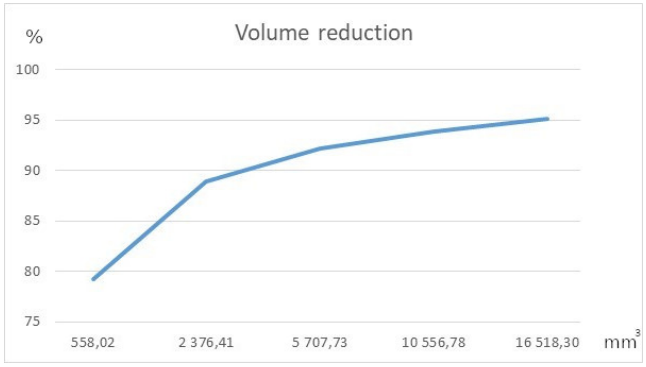

Figure 24: *Volume reduction as a function of the object size, computed on the owl model. The method becomes increasingly efficient.*

As future work, we think our technique could lead to novel *external* support structures. We show preliminary results in Figures 26 and Figure 27, as well as Table 6, comparing to the bridge structures of Dumas et al. [DHL14]. When used externally our technique produces walls that extend from the print to support areas in overhang above. This produces surprisingly small and effective supports, apart from of few places where large walls appear (see also Section 5.4). The structures are however hard to remove if printed with PLA or ABS, as they are strongly connected to the model, but printed with soluble materials (PVA/HIPS) they could be efficiently dissolved. This is a natural direction for future research.

Acknowledgements

We would like to thank Weiming Wang for the generated models obtained with Support-Free Hollowing [WLW*18].

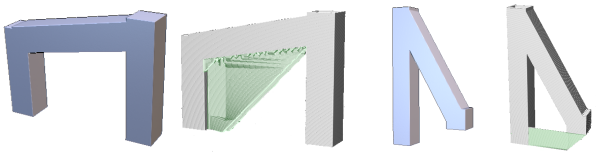

Figure 25: ***Left****: Low angle, flat overhang areas mislead our wall direction heuristics, generating suboptimal supports.* ***Right****: Overhang area located far away from the rest of the model and close to the bed. Our method connects the support for that overlang to the post on the left, using up a lot of material. A pillar would use much less material here.*

Model	Bridging the gap	Our method
Fawn	55cm (22min)	10cm (3min)
Nail	120cm (40min)	78cm (12min)
Bridge	46cm (16min)	53cm (11min)
Bunny	65cm (25min)	8cm (4min)
Female Knight	127cm (1h38min)	36cm (1h)
Minotaur	251cm (3h12min)	48cm (1h)

Table 6: *Material and printing time spent on external supports by our method compared to bridging the gap.*

Figure 26: *Demon dog printed using our method for external support.*

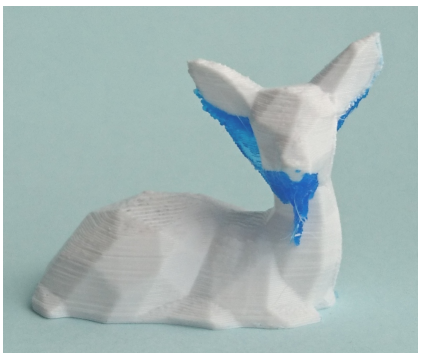

Figure 27: *Half-size fawn model, printed with our method for external support, using a blue filament color for supports.*

References

[DHL14] DUMAS J., HERGEL J., LEFEBVRE S.: Bridging the Gap: Automated Steady Scaffoldings for 3D Printing. *ACM Transactions on Graphics 33*, 4 (July 2014), 98:1 – 98:10. 4, 11

[HL18] HORNUS S., LEFEBVRE S.: Iterative Carving for Self-supporting 3D Printed Cavities. In *EG 2018 - Short Papers* (2018), Diamanti O., Vaxman A., (Eds.), The Eurographics Association. 3, 4, 9, 10, 11

[HLDC16] HORNUS S., LEFEBVRE S., DUMAS J., CLAUX F.: Tight Printable Enclosures and Support Structures for Additive Manufacturing. In *Eurographics Workshop on Graphics for Digital Fabrication* (2016), e Sa A. M., Pietroni N., Echavarria K. R., (Eds.), The Eurographics Association. 3, 5

[HYW*08] HUANG X., YE C., WU S., GUO K., MO J.: Sloping wall structure support generation for fused deposition modeling. *The International Journal of Advanced Manufacturing Technology 42*, 11 (2008), 1074. 4

[Ice] ICESL: Progressive infills. https://twitter.com/iceslapp/status/1000063188082282497. 3

[LEM*17] LIVESU M., ELLERO S., MARTÍNEZ J., LEFEBVRE S., ATTENE M.: From 3D models to 3D prints: an overview of the processing pipeline. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 537–564. 3

[LL17] LEE J., LEE K.: Block-based inner support structure generation algorithm for 3d printing using fused deposition modeling. *The International Journal of Advanced Manufacturing Technology 89*, 5 (Mar 2017), 2151–2163. 3

[LLL17] LEE T., LEE J., LEE K.: Extended block based infill generation. *The International Journal of Advanced Manufacturing Technology 93*, 1 (Oct 2017), 1415–1430. 3

[MSWS00] MCMAINS S., SMITH J., WANG J., SÉQUIN C.: Layered manufacturing of thin-walled parts. In *ASME Design Engineering Technical Conference, Baltimore, Maryland* (2000), Citeseer. 3

[Sli] SLIC3R: 3D infilling: faster, stronger, simpler . http://manual.slic3r.org/expert-mode/infill. 3

[SU14] SCHMIDT R., UMETANI N.: Branching Support Structures for 3D Printing. In *ACM SIGGRAPH 2014 Studio* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 9:1–9:1. 4

[Ult] ULTIMAKER: Gradual infill. https://ultimaker.com/en/resources/52670-infill. 3

[VGB14] VANEK J., GALICIA J. A. G., BENES B.: Clever support: Efficient support structure generation for digital fabrication. *Comput. Graph. Forum 33*, 5 (Aug. 2014), 117–125. 4

[WLW*18] WANG W., LIU Y., WU J., TIAN S., WANG C. C. L., LIU L., LIU X.: Support-free hollowing. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. 4, 9, 10, 11

[WWY*13] WANG W., WANG T. Y., YANG Z., LIU L., TONG X., TONG W., DENG J., CHEN F., LIU X.: Cost-effective printing of 3d objects with skin-frame structures. *ACM Transactions on Graphics 32*, 6 (2013), 177:1–177:10. 3

[WWZW16] WU J., WANG C. C., ZHANG X., WESTERMANN R.: Self-supporting rhombic infill structures for additive manufacturing. *Computer-Aided Design 80* (2016), 32 – 42. 3

[XC17] XIE Y., CHEN X.: Support-free interior carving for 3D printing. *Visual Informatics 1*, 1 (2017), 9 – 15. 3

[ZXW*15] ZHANG X., XIA Y., WANG J., YANG Z., TU C., WANG W.: Medial axis tree – an internal supporting structure for 3d printing. *Computer Aided Geometric Design 35-36* (2015), 149 – 162. Geometric Modeling and Processing 2015. 4