EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)

Volume 38 (2019), Number 2

Object Partitioning for Support-Free 3D-Printing

E. Karasik, R. Fattal, M. Werman

Hebrew University of Jerusalem, Israel

Figure 1: *An object requiring support when printed by an FDM printer. Two left images show the object with support structures and after their removal by a craft knife. Removal marks are clearly seen despite the elaborate effort put. Partitioning the object by a single plane of a precise location and angle results in two printable subparts. Gluing the two results in a cleaner surface, 20 minutes shorter printing-time, and saves 70cm of filament.*

Abstract

Fused deposition modeling based 3D-printing is becoming increasingly popular due to it's low-cost and simple operation and maintenance. While it produces rugged prints made from a wide range of materials, it suffers from an inherent printing limitation where it cannot produce overhanging surfaces of non-trivial size. This limitation can be handled by constructing temporary support-structures, however this solution involves additional material costs, longer print time, and often a fair amount of labor in removing it.

In this paper we present a new method for partitioning general solid objects into a small number of parts that can be printed with no support. The partitioning is computed by applying a sequence of cutting-planes that split the object recursively. Unlike existing algorithms, the planes are not chosen at random, rather they are derived from shape analysis routines that identify and resolve various commonly-found geometric configurations. In addition, we guide this search by a revised set of conditions that both ensure the objects' printability as well as realistically model the printing capabilities of the printer at hand.

Evaluation of the new method demonstrates its ability to efficiently obtain support-free partitionings typically containing fewer parts compared to existing methods that rely on support-structures.

CCS Concepts

• ***Computing methodologies*** → ***Shape analysis;*** *Mesh models;*

1. Introduction

Until the end of the last decade, 3D-printing was primarily used in the manufacturing industry, mainly for rapid prototyping of new products. The expiration of the Fused Deposition Modeling (FDM) printing process patents in 2009 led to a revolution in which this inexpensive technology reached the masses. By now FDM printers are an extremely popular tool among enthusiastic home users, makers, artists, small labs and production facilities where they are used for personalized production, architectural depiction, medical replacements, hobby recreation, and many other applications.

The FDM technology consists of extruding strands of a melted thermoplastic through a narrow nozzle and accurately depositing them side by side, and layer on top of layer. These deposits fuse together as they solidify and result in a 3D solid object. This printing technology can be implemented at very low cost, as low as a hundred US dollars, it supports a wide range of materials such as industrial ABS, environmentally-friendly Polylactic Acid, flexible nylon, hard polycarbonate, and wood- and metal-filled polymers, and it does not require additional materials (powders) or post-solidification processes. Consequently, FDM-based 3D-printers are by far the most popular non-industrial printers.

The FDM's layered operation results in a non-smooth object surface that reveals the printed layers. This anisotropy also leads to a structural weakness along the vertical (printing) axis. Nevertheless, the main shortcoming of this printing technology is its inability to form *overhangs*—portions of the object with no material supporting them from beneath, such as bridges and steep protrusions. While this can be partially solved by printing temporary support structures, this option is not entirely satisfactory as it involves specialized costly materials, generates excessive waste, leaves marks behind, increases the printing time, as well as increases the printer's complexity and cost when an additional hotend is used.

In practice users often avoid these problems by splitting the object into parts that do not contain overhangs, which are printed separately and glued together. This simple solution is quite effective and can be executed with minimal aesthetic and structural compromises. Nevertheless, breaking an object into printable subparts becomes an exponentially intricate task as the object complexity increases, requiring a good 3D geometrical imagination as well as fair knowledge in CAD software, which is not always the case as many users rely on existing art. Indeed, it was shown that this decomposition is NP-hard in [FM01] under a conservative definition of overhangs. Figure 1 shows the option of partitioning versus the use of support structures.

These difficulties motivated the development of automated object partitioning methods. Hu et al. [HLZCO14] decompose an object into approximately pyramidal subparts that can be printed with minimal support material. The restriction to pyramidal hulls leads to a higher number of subparts than needed as existing printers are capable of printing protrusions with a slope up to 150°. Wei et al. [WQZ*18] concentrate on shell models and derive a randomized skeleton-based algorithm that is applicable to locally-cylindrical objects. Yu et al. [YYT*17] operate on general meshes by also searching the partitioning via a randomized search. As we show random search requires longer running times and results in more parts than an object-analysis based search.

In this paper we describe a new method for partitioning general 3D models into a small number of subparts that can each be printed *with no* additional support structure. The algorithm identifies common non-printable geometric patterns in the object and separates them into printable or simpler subparts. The partitioning is based on a local and global object analysis that extracts geometry-specific separating planes efficiently. In order to fully decompose an object, a sequence of these partitioning steps is found by a stochastic search that minimizes the number of resulting subparts. This search is performed according to a new, complete and realistic printability criterion that takes into account the printing capabilities of a given printer. Evaluation of our algorithm demonstrates the effectiveness of our analysis-based partitioning to achieve optimal or near-optimal solutions faster than existing alternatives.

2. Previous Work

The recent growth in the popularity of 3D-printers gave rise to various computational fabrication problems that drew the attention of computer graphics researchers. We start with a brief review of this heterogeneous literature and then focus on works related to object partitioning.

Computational Fabrication. The trade-off between object strength and material cost has been the topic of several works. Stava et al. [SVB*12] describe a system that detects and corrects structural weaknesses by modifying the object's in-fill density and inserting struts. A simulation-aided shape editing system that integrates structural analysis and geometric design is described in [XXY*15]. Lu et al. [LSZ*14] introduce hollow honeycomb-cells to optimize the objects' strength-to-weight ratio. Wang et al. [WWY*13] also use struts in order to convert 3D solid objects into stable skin-frame approximate structures to reduce material cost. Similarly, Vanek et al. [VGB*14b] minimize both the material and printing time by converting the object into an outer shell and segmenting it so that an optimal packing with minimal support material is obtained. Zehnder et al. [ZCT16] describe a computational tool for designing structurally-sound and printable ornamental curve networks. A method for gradual construction of stable masonry model that reduces the material and construction costs is described in [DPW*14]. Zhang et al. [ZXW*15] propose a method for designing the internal supporting frame of 3D objects. Finally, Hornus and Lefebvre [HL17] describe a method for carving large self-supporting cavities inside an object.

Several works addressed the print-size limit of existing consumer FDM printers. Luo et al. [LBRM12] and Jadoon et al. [JWL*18] decompose large objects into smaller ones that fit into the printer's volume while considering their number, assemblability, and structural soundness. Song et al. [SFLF15] avoid the need for glue or special connectors and propose printing interlocking parts that allow repeated assembly and disassembly.

Additional works augment the object design process and its expressiveness. Prévost et al. [PWLSH13] introduce minimal deformations to the object volume and shape in order to properly balance it. Calì et al. [CCA*12] describe a framework for converting static models into ones with functional articulation containing joints with internal friction that do not require assembly.

Figure 2: *Left illustrations show an object with internal angles below* 135°. *While this object is printable via today's FDM printers, its pyramidal decomposition consists of three parts. Right image shows an actual print of a benchmark model designed for estimating the maximal printing angle supported. The test shows a successful operation at angles up to* 150°. *In this test we used a Printrbot Simple Metal with a E3D v6 hotend, printing a natural PLA at* 185° *Celsius through a 0.4mm nozzle.*

Unlike the conservative layer-by-layer support structures created by standard slicing software, several works take a geometric approach to construct more effective support structures. Dumas et al. [DHL14] exploit the fact that FDM printing has some tolerance to short bridges and incorporate a scaffolding structure composed of horizontal bridges and vertical pillars to support the object. Vanek et al. [VGB14a] describe a geometry-based approach that minimizes the support material by finding the orientation requiring the least amount of support area and minimize the number of points needed to support overhangs. Most recently, Dai et al. [DWW*18] propose a robotic printing system equipped with a multi-axis motion designed in order to reduce the need for supporting structures.

In FDM printing the object orientation affects the anisotropies it creates. Hildebrand et al. [HBA13] account for the anisotropies and find the best print direction for each part of the object in terms of build speed, tensile strength and strain. Zhang et al. [ZLP*15] determine the printing orientations such that the residual remains of the support structures are not in perceptually significant regions.

Object Partitioning for 3D Printing. Partitioning an object in order to achieve printability by existing FDM printers is the topic of several recent works. We mention here the ones that are more relevant to our settings, and refer the reader to Oh et al. [OZB18] for a more comprehensive survey of the topic.

Decomposing an object into geometrically restricted subparts is a valid approach to achieve printability. Indeed, both pyramidal object decompositions and convex decompositions result or are proxy to achieving printable parts. However, obtaining such exact decompositions is NP-hard to compute [TM84, Cha84, FM01]. Nevertheless, Hu et al. [HLZCO14] showed that the pyramidal decomposition is more likely to result in fewer printable parts, and consequently proposed an approximate pyramidal decomposition. Their method joins fine object subunits to form larger primitives through a series of clustering steps guided by the agreement on a printing base. While the resulting subparts are not guaranteed to be pyramidal shapes, being almost restricted to their pyramidal hull implies a saving in support material. As noted above, existing FDM printers tolerate non-trivial protrusions and hence the over conservative pyramidal condition often leads to an unnecessarily large number of subparts and post-printing labor. Figure 2 portrays such an example. The method we derive here is based on a more realistic printability condition taking into account the maximal protrusion angle the printer at hand supports.

Wang et al. [WZK16] propose a decomposition in which the surface normals are as perpendicular as possible to the printing direction in order to improve the surface quality and reduce the support needed.

Demir et al. [DAB18] also describe a method that decomposes the object into near-convex components by means of energy-minimization segmentation. The resulting subparts are then packed together to make the printing process more efficient in terms of material usage, printing times, and object fidelity. Similarly to pyramidal decomposition, the near-convex objective used in this method results in unnecessarily large number of parts and laborious assembly.

Wei et al. [WQZ*18] focus on locally-cylindrical shell-models which they model as a graph based on the object skeleton that they compute. Partitioning is then found through a stochastic search in which a random vertex in the graph is selected and a planar cut perpendicular to its edges is considered. The partitioning is finally obtained based on the skeleton's printability. Unlike this restricted geometric interpretation of the object as a collection of cylinders, we search and support multiple and more general geometric constellations. Moreover, our method ensures the meshes resulting from our partitioning are fully printable.

Finally, Yu et al. [YYT*17] do not make geometrical assumptions on the object shape and do not attempt to analyze it. Instead, they perform a randomized genetic search for a binary space partitioning tree that defines the object partitioning. The total overhanging area and the number of cuts define the objective function being minimized. Thus, some support material may be needed. In contrast to this fully-randomized search, our method analyses the object geometry and suggests cutting planes that resolve the non-printable portions of the object.

3. Printability Condition

The printability criterion plays a critical role in any method that computes a support-free decomposition. An over-stringent condition leads to too many subparts, whereas an incomplete set of criteria may lead to serious printing artifacts or even failure.

As noted above, the FDM printing process lays every layer on top of the previous one, which provides the support needed. However, the strong adhesion between successive layers and the rapid solidification of the newly deposited strands allow this process to tolerate a partial support where newly deposited layers are allowed to push the perimeter outwards to a certain extent. In other words, a reliable print of protruding faces is possible as long as the partial inter-layer support is sufficient. This extent is defined by the inner angle formed between the base and face planes [VGB14a]. We denote the maximal printing angle by θ_{crit} and refer to it as the *critical protrusion angle* (CPA). The CPA gives raise to a necessary printability condition that requires all the object faces to form an angle less than θ_{crit} with respect to a given base-plane.

We note that while θ_{crit} may slightly vary between printers and

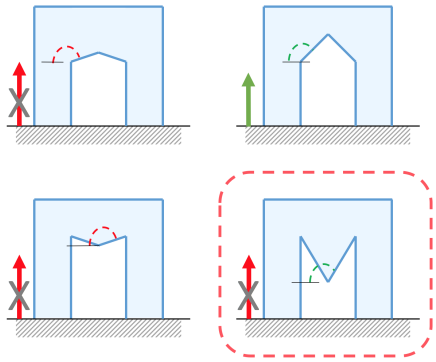


Figure 3: *Top-row shows illustrations of concave arches with faces that form an internal angle above the CPA (critical protrusion angle) on the left, and below the CPA on the right (internal angles are shown in dashed lines). In both cases, the CPA condition alone predicts correctly the printability. Bottom-row shows convex arches, where at the left the faces exceed the CPA, and at the right the faces angles fall below the CPA. Nevertheless, the latter configuration cannot be printed as the vertex from which these faces originate has no support.*

print settings, current consumer FDM printers can sustain up to $\theta_{crit} = 150^\circ$, see [JSXZ18]. The restriction to a pyramidal hull in [HLZCO14] corresponds to a CPA condition of $\theta_{crit} = 90^\circ$, which is considerably more stringent than what is needed in practice. The method in [WQZ*18] ensures that each skeletal edge meets the CPA condition, however, the mesh associated with the edge is made of a variety of faces, some of which may still exceed θ_{crit}. Yu et al. [YYT*17] also use a coarser representation of the mesh which may also miss CPA-violating faces.

The CPA condition applies on faces and ensures that every layer is properly supported by the previous one and, as noted above, was used to define printability in prior works. There are cases, however, where there is *no supporting geometry* in the previous layer while the CPA condition is *not violated*. A convex arch, as shown in Figure 3, is an example of a non-printable case whose faces satisfy the CPA condition. In this shape the faces around the lowest tip point of the ceiling are above it and do not support it. In order to avoid such singularities, we add the *supporting face* (SF) condition requiring that every vertex or open edge belonging to a face, pointing towards the base, must have support from bellow.

Claim: The CPA and SF conditions are necessary and sufficient for object printability. The following proof shows that these conditions correctly predict whether *every* point of the object has sufficient support or not.

Proof: *Sufficiency.* Let p be a non-printable point of the object with minimal height from the base. Any point in an object is either an inner point, inside an open face, inside an open edge, or a vertex. The point p cannot be an inner point of the solid object since it will have supporting matter below it. Similarly, p cannot be an inner point of a face since the CPA condition ensures the printability of all its points.

If p is on an open edge of a face, it is adjacent to two faces. Since it is non-printable, both of them must not have an inner point below p supporting it, which violates the SF condition. If the non-printable p is a vertex, it cannot belong to a face with points below it, which again, violates the SF condition.

We thus showed that a non-printable point in the object will necessarily be detected by either the CPA or SF conditions and hence the two provide a sufficient condition for printability.

Necessity. If there is a face not obeying the CPA condition, by its definition, the different levels (points) in this face do not provide a sufficient support for it to be printable. In case the SF condition is not met, the vertex or edge does not have support from below and can not be printed. An example of a violation of SF is portrayed in the final object in Figure 3 which is not printable.

The shape analysis procedures we use in our algorithm, and describe next, use these conditions as their printability criterion.

4. New Method

Fekete and Mitchell [FM01] show that the problem of deciding whether a polyhedra is decomposable into k pyramidal polyhedra is NP-hard. As we explain below, the restriction to pyramidal polyhedra corresponds to a special case in terms of the printability condition we use, namely a critical printing angle of 90°. The proof in [FM01] is rather elaborate but gives no reason to suspect this problem becomes any easier for higher, more realistic, critical angles that we use in practice.

In order to cope with this NP-hard problem we use a divide-and-conquer approach in which we recursively split the object into subparts as long as a non-printable part is encountered. Unlike previous approaches [WQZ*18, YYT*17] that perform a fully-randomized search of the cutting-planes, we apply shape analysis procedures that look for different geometric configurations in the object and suggest planes that extract printable or greatly simplified subparts. Randomness enters in the choice of the analysis procedure applied at each level of the recursion and its internal initialization. In Section 5 we show this analysis-based approach achieves lower numbers of subparts at lower running times.

We start by a brief description of each of the three shape analysis procedures that we use in this recursion.

- *Base Extraction.* Identifies planar surfaces of the object that can act as a printing-base for a large portion of its volume.
- *Tip Extraction.* Detects protruding narrow regions of the object, such as tips, whose faces are less likely to serve as useful bases. The procedure creates a new base within the object opposite to the tip that supports it.
- *T-Junction Splitting.* Is an object simplification procedure that attempts to separate between distinct primitives in the object.

The rest of the section provides a more detailed description of each of these procedures.

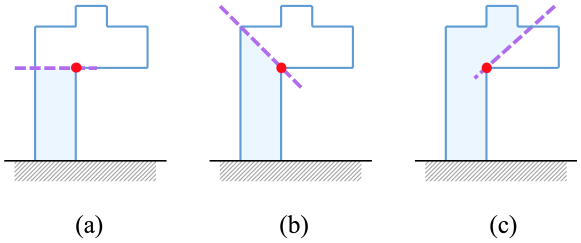


Figure 4: *Base Extraction steps. The first iteration (a) encounters a non-printable edge (with red). The second step (b) corrects the angle and increase the printable volume. The third step (c) increases the printable subpart up to the point where the cutting-plane reaches θ_{crit} and will produce non-printable faces if rotated any further.*

Algorithm 1 Base Extraction

procedure EXTRACTBASE(part, base)
  $\gamma = 15^\circ$
  $\vec{N} \leftarrow$ base.normal
  while $\angle\left(\text{base.normal}, \vec{N}\right) \leq \theta_{\text{crit}}$ and $\gamma \geq 0.1^\circ$ **do**
    PQ $\leftarrow \emptyset$ (priority by distance to base plane)
    PQ.add(base)
    while PQ $\neq \emptyset$ and Printable(PQ.top(), base) **do**
      top $\leftarrow$ PQ.pop()
      PQ.add(top.neighbors())
    if PQ = $\emptyset$ **then return** printable
    v $\leftarrow u \in$PQ.top() closest to base
    subpart $\leftarrow$ part $\cap$ HalfSpace(v, $\vec{N}$)
    c $\leftarrow$ avg(part $\cap$ Plane(v, $\vec{N}$))
    $\vec{N} \leftarrow$ RotateTowards($\vec{N}, c - v, \gamma$)
    $\gamma /= 2$
  return subpart, part $\setminus$ subpart

4.1. Base Extraction

Thingiverse and other online shared design repositories suggest that a large portion of 3D-printed objects are designed by CAD software. These objects as well as ones created by other means often contain large planar surfaces with a potential to serve as a print-base for a substantial portion of the object. Moreover, at each level of the recursion our partitioning algorithm performs a planar cut that adds planar surfaces to the object. The procedure we describe here detects these potential bases and greedily searches for the largest printable parts they can print.

Since planar surfaces may be tessellated into multiple faces, we start by detecting these surfaces using a breadth-first search (BFS) in which we group together adjacent faces with identical normals. Note that any two adjacent planes forming an angle greater than θ_{crit} between them violate the CPA condition with respect to one another and hence cannot serve as printing bases. Thus, we scan the list of potential bases for such pairs and remove them. In addition we exclude surfaces that cannot serve as bases since part of the object is below them. As a first and efficient step we exclude bases that have a neighbouring face beneath them. Cases with a more global obstruction are eliminated later. We re-generate this list of candidate base planes at each level of the recursion, allowing previously-eliminated planes to serve as a base later in the recursion.

To avoid considering the same surface multiple times at the same level of the recursion, we compute the list of potential bases once and store it in a stack from which every surface will be popped once. Since the area of a base correlates with the amount of volume it can support, we order the surfaces in the stack according to their area. However, the stacks of two consecutive levels are likely to share many surfaces in common and give rise to the exploration of similar sequences of bases. We widen this search space by adding some randomness to the sorting by area. In particular, we sample a uniform random variable $u_i \sim U[0,1]$, for each surface i, and sort the surfaces according to the order of $u_i^{a_{max}/a_i}$, where a_i are their areas.

Next, given a base popped from the stack we perform a two-step iterative optimization to find the largest printable subpart that can be isolated from the object by a planar cut. In the first step we assume the cutting-plane is parallel to the base-plane and construct a priority queue containing faces neighboring the base, i.e., that share an edge or a vertex. The faces are sorted according to their minimal distance from the base plane. We then extract the first face from the queue, verify that it can be printed from the base and if so, remove it and add its new neighbors to the queue. As this process advances, the ring of faces in the priority queue gets farther from the base in a synchronized manner and the minimal distance in the queue increases. Once an unprintable face is encountered, the process stops and the minimal distance found so far is the distance of the cutting-plane from the base-plane. Since the printability of all the faces below it was verified during this process, the subpart extracted by this cutting-plane is *printable*. The faces and the polyline resulting from this intersection are stored and used in the next step. The intersection of the object with a cutting-plane is restricted to the connected-component containing the feature we started from, the base surface in this case.

At the second step of this procedure, we attempt to increase the volume printable from the base. We find the center c of the cut's polyline, and v, the vertex closest to the base in the non-printable face encountered in the first step. We then rotate the cutting-plane's normal towards v around c. We repeat the optimization starting from this rotated cutting-plane.

If a new limiting face is encountered in the first step, we run another attempt with half the rotation angle previously used. This iterative process stops once the rotation angle falls below a threshold of 0.1° or the rotated cutting-plane itself reaches the CPA, θ_{crit}, where it will start generating non-printable faces. Algorithm 1 summarizes this base extraction procedure. Finally, since all the subparts extracted in this iterative scheme were tested for printability with respect to the *same base* and are all connected to one another, their union is also printable. Therefore this procedure outputs their union and the remaining portion of the object.

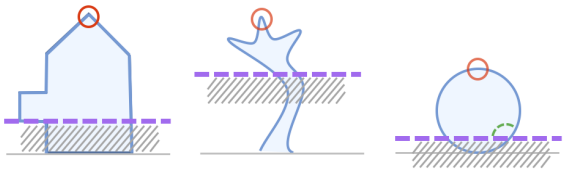


Figure 5: *Three examples showing the printable subpart found by the tip extraction procedure. Assuming $\theta_{crit} < 135^\circ$, each of these solutions are optimal, or appear in the optimal solution.*

4.2. Tip Extraction

Another class of geometric features commonly found in organic and CAD-based objects are protrusions which may be smooth, spiky or rectangular. We collectively refer to these features as *tips*. Being relatively narrow parts of the object, their faces are less likely to serve as useful bases, and in many cases the optimal approach is to print them from the opposite direction (beneath). Figure 5 shows several examples of objects containing tips and their optimal partitioning.

The procedure described here detects tips and searches for the farthest base-plane inside the object from which they can be printed. The detection is done by running a multi-scale tip template matching over a voxelized grid representation of the object (100^3 voxels in our implementation). We use 26, 3x3x3 binary templates where the center voxel and one other voxel are set to 1, and the rest to $-2/25$. These zero-sum templates do not respond to constant regions, i.e., away from object boundary.

When convolved with a voxelization of the object where object is 1 and empty-space is -1, local maxima are expected at very fine tips. Larger scale tips are detected by running an á trous multi-scale hierarchy in which 2^l zeros are inserted at the l-th level in between every pair of values of the templates [HT90]. These stretched templates respond to larger tips, yet contain the same number of non-zero elements and remain efficient to compute. We run a 3x3x3 non-maximum suppression over the 26 thresholded responses (1.6 in our implementation). The remaining non-zero points are potential tips. The orientations of the tips are obtained from the index of the specific template that showed a maximal response (recall that each template is associated with a different tip orientation).

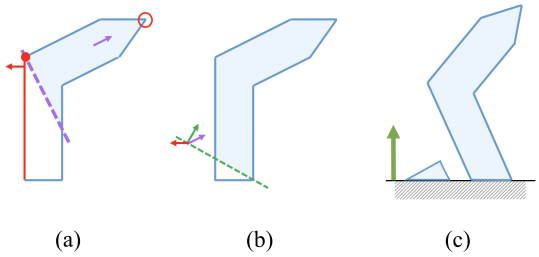
Similarly to the base-extraction procedure described above, we store the tips found in the current recursion step in a stack and do not pick the same tip twice at the same recursion level. However, they are randomly ordered in this case.

Given a suspected tip, we search for the cutting-plane, which is also the base-plane in this case, in two steps. In the first step, the tip's location and orientation are used to initialize this plane and the queue is fed with the face closest to the tip. The queue is prioritized according to the distance above this initial plane.

The repeated updates of the queue are identical to the ones performed in the base extraction and the ring of faces together with the base evolves away from the tip until it encounters a limiting non-printable face. The process is illustrated in Figure 6. This process has the capacity to extract multiple tips as long as they are printable from the same base, as shown in Figure 7.

Algorithm 2 Tip Extraction

procedure PRINTABLEPART(part, tip_face, $\vec{N}$)
  point ← centroid(tip_face)
  PQ ← ∅ (priority by distance to Plane(point,$\vec{N}$))
  PQ.add(tip_face)
  while PQ ≠ ∅ and Printable(PQ.top(), $\vec{N}$) **do**
    top ← PQ.pop()
    PQ.add(top.neighbors())
  subpart ← part ∩ HalfSpace(PQ.top(), $\vec{N}$)
  overhang_normal ← PQ.top().normal
  return subpart, overhang_normal
procedure EXTRACTTIP(part, tip_face)
  for $i \leftarrow 1,\ldots,30$ **do**
    $\vec{N}$ ← sample spherical cap(tip_face.normal, 60°)
    sub, $\vec{O}$ = PrintablePart(part, tip_face, $\vec{N}$)
  for 5 best results **do**
    for i = 1,...,5 **do**
      rotation ← $min\left(\angle\left(\vec{N},\vec{O}\right) - \theta_{\text{crit}}, 15^\circ\right)$
      $\vec{N}$ ← RotateTowards($\vec{N}$, $\vec{O}$, rotation)
      sub, $\vec{O}$ = PrintablePart(part, tip_face, $\vec{N}$)
  result ← sub with largest volume
  return result, part \ result



(a) (b) (c)

Figure 6: *Tip Extraction steps. The first iteration (a) encounters a non-printable face (marked in red). The second step (b) brings the cutting-plane's normal closer (as shown by green arrow) to that of the limiting face (red), and thereby increases the printable volume. The resulting parts are printable as shown in (c). For this example to be realistic assume $\theta_{crit} = 135^\circ$.*

The base-extraction procedure exploited the fact that portions of the object can be printed starting from its boundary faces. Here, we are creating a new base plane, and hence have the opportunity to explore additional plane orientations around the tip's direction. We carry this out by performing the search above multiple times, starting from 30 planes whose normals sample a spherical cap of 60° around the tip's orientation. We pick the five bases extracting the highest volume for further refinement.

The search of each of these bases encountered a limiting face, that forms a mutual angle ω greater than θ_{crit}. Nevertheless, a local refinement in the plane's orientation, towards this face, may enable its printability and extract a larger printing volume.

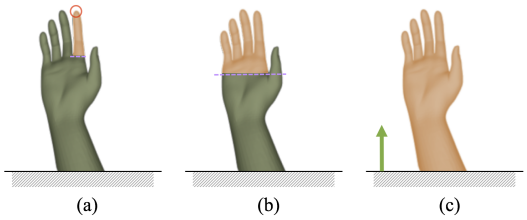


Figure 7: *Priority queue progression showing its ability to consider faces behind the cutting-plane once reached. The procedure was initiated from the tip shown by the red circle on the left. In the subsequent steps: (a) the priority queue visited the all faces of the index finger. Once it reaches the faces below this finger, (b) the queue processed all four fingers, and (c) the same took place once it reached the thumb.*

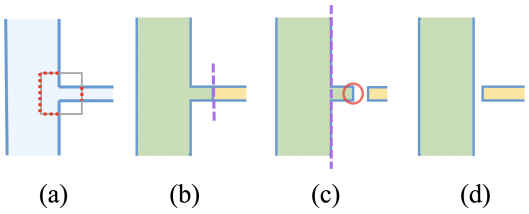


Figure 8: *T-junction Splitting steps. At first, (a) the center of the T-junction is detected, then (b) an initial cut is made at the narrow connected-component. Then, (c) a base extraction is applied on the subpart containing the wider component. This procedure removes the remaining narrow geometry it contains. This cut separates the two primitives and the initial cut is abandoned, as shown in (d). The detected connected-components at this regions are indicated by the dashed red lines. Note the example shape shown here is a part of a larger shape where the tip-extraction step is not applicable.*

We do this by rotating the plane by $\min\{\omega - \theta_{\text{crit}}, 15^\circ\}$ such that it either enables the printability of the limiting face, or moves towards this configuration when the angle is large. After changing the plane's orientation, we must recompute the extractable portion from the start, and repeat the process above starting from a queue containing the tip's closest face. Finally, we output the plane yielding the maximal volume. Algorithm 2 summarizes this tip extraction procedure.

4.3. T-junction Splitting

Many complicated and non-printable objects were created by joining simpler primitives, e.g., when using a union operation in CAD-based software. In such cases, simplifying the object by breaking it into not-necessarily-printable subparts can lead to a better solution compared to the greedy procedures described above. In this procedure we attempt to detect such cases by searching and splitting T-junctions in the object.

Locally, T-junctions can be identified as regions in which the intersection of the object with the surface of a cube contains two (or more) connected components, as depicted in Figure 8(a). We detect these regions by applying a graph-based detector over a graph constructed by voxelizing the object. Specifically, voxels inside the object are the vertices and neighbouring object voxels are connected by an edge.

We check whether a vertex is in the vicinity of a T-junction by analyzing the shape of the object around it. This is done by considering all the vertices of a certain distance k from this center vertex. As shown in Figure 8(a), when the center vertex is at the vicinity of T-junctions, this set is expected to consist of two or more connected-components. While multiple connected-components can be found in other shapes, such as a cylinder, T-junctions are characterized by having a large and one or more smaller connected-components. We search for such cases by gradually increasing k and halting either when a large ratio is found between the components (3 in our implementation) or when we reach a maximal distance of 20. We apply this search at a sub-sampled grid of the graph, at every fifth voxel along each axis.

Once a T-junction is found, we separate it into the two geometric entities with the following two steps. We start with an initial splitting biased towards the narrow component so that the wider side contains residual geometry from the narrow side. Specifically, we calculate the centers of masses of both the narrow c_n and wide c_w components and define the cutting-plane as the one perpendicular to their offset, $c_n - c_w$, and passing though c_n.

In the second step we execute the base extraction procedure over the subpart containing the wider component, with the base set to the newly-created plane produced by the initial cut. This procedure is expected to advance towards the wider component and separate it from the remaining geometry of the narrow primitive in this subpart. We then undo the initial cut done in at the first step and end up only with the parts produced by the last tip extraction step. A detailed depiction of these steps is given in Figure 8.

Note that when this procedure is successfully applied, the surface produced by the cut creates a viable potential tip at the subpart containing the narrow component. Similarly, in case the subpart containing the wider component was planar at the interface, with the narrow arm removed this separation may recover a potential base. Such cases, if they exist, are likely to be explored in the subsequent levels of the recursion.

4.4. Randomized Recursive Search

The randomized search for the best possible object partitioning uses these three splitting procedures recursively, over the remaining non-printable parts of the object. Once all the resulting parts are printable, a candidate solution is reached. The stopping criterion of this search is a predefined limit on the number of total cutting-plane operations n_{cuts} performed. This value controls the trade-off between optimality and running time of the algorithm. Algorithm 3 describes this recursive search.

In addition to the randomness taking place inside the splitting procedures, the search randomly picks which and how many procedures to use at each recursion level. As we show in Section 5, the probability of these different actions has a significant effect on overall performance. Therefore, we learn the probabilities to pick

the base extraction P_{base}, tip extraction P_{tip}, T-junction splitting P_T, procedures and the probability of calling an additional procedure within a level, $q^d P_{call}$, which decreases at rate q as a function of the recursion depth d. This is carried out for different stopping conditions n_{cuts}. In Section 5 we describe the results obtained by using different n_{cuts} values.

Additionally we do not allow the recursion depth to exceed $d_{\max}$, initialized to 15, and update it every time a fully-printable partition is found. Thus, in case of an "easy" object, $d_{\max}$ will decrease early in the search (even if the optimal solution is yet to be found).

Algorithm 3 Search

```
procedure RecursiveSearch(Parts, depth)
    Bases, Tips, Ts = DetectPrimitives(Parts)
    while n_cuts ≥ 0 and depth < d_max do
        part ← non-printable part from Parts
        NewParts ← Parts \ part
        r ← random(0,1)
        if r ≤ P_base then
            NewParts ∪ = ExtractBase(part, Bases.pop())
        else if r ≤ P_base + P_tip then
            NewParts ∪ = ExtractTip(part, Tips.pop())
        else
            NewParts ∪ = SplitT(part, Ts.pop())
        n_cuts ← n_cuts − 1
        if ∀part ∈ NewParts printable then
            if |NewParts| < best_n then
                best_n ← |NewParts|
                Result ← NewParts
                d_max ← depth
        else
            RecursiveSearch(NewParts, depth + 1)
        if random(0,1) > P_call · q^depth then break
procedure Search(object, n_cuts)
    Result ← ∅, best_n ← ∞, d_max ← 15
    RecursiveSearch({object}, 0)
    return Result
```

4.5. Post Processing

The printable partitions found by our recursive search may be over fragmented. To remain efficient, we do not search for alternative bases besides the ones found so far, and for a full solution simply check whether pairs of interfacing parts can be printed from one of their already found bases and if so merge them.

Finally, the greedy nature of the tip and base extraction steps may result in partitions containing excessively small subparts which are fragile and hard to glue, e.g., the small triangular residue in Figure 6. As a simple solution, we scan the cuts made in a reverse order and retract the extracted tip- and base-planes along their normal by a few millimeters ensuring sufficient material strength (2 millimeters in our implementation) as long as all the subparts remain printable.

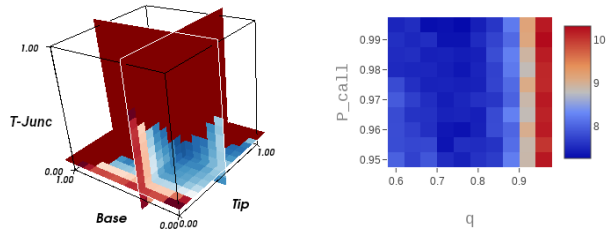


Figure 9: *Optimizing Probabilities. Left graph show the optimization landscape of P_{base}, P_{tip} and P_T, where $P_{rand} = 1 - P_{base} - P_{tip} - P_T$. A well defined global optimum can be observed in the optimal planes shown. Right graph shows the landscape of P_{call} and q, that shows a trade-off between the two. The colors represent the average number of subparts in the training set.*

These graphs were obtained for the case of $\theta_{\text{crit}} = 135^\circ$ with $1k$ cuts limit.

CPA	n_{cuts}	P_{call}	q	P_{Tip}	P_{Base}	P_T	P_{Rand}	AN
135	1k	0.995	0.68	0.5	0.3	0.1	0.1	7.24
135	3k	0.985	0.76	0.5	0.2	0.2	0.1	5.9
135	5k	0.995	0.8	0.6	0.3	0.1	0	5.54
150	1k	0.965	0.72	0.6	0.3	0.1	0	4.21
150	3k	0.99	0.68	0.7	0.2	0.1	0	3.93
150	5k	0.995	0.8	0.6	0.3	0.1	0	3.88

Table 1: *The optimal parameters found over the training set and the average number (AN) of subparts they achieve. The learning procedure was carried out for two different critical protrusion angles* 135° *and* 150°.

5. Results

We implemented our method in C++, using the Computational Geometry Algorithms Library (CGAL). Timings were measured on an Intel (R) i5-6600 3.5GHz CPU with 16GB of RAM. Our implementation assumes watertight closed mesh that represents the boundary of the object to be printed. Before discussing the evaluation and comparison of our method against the state-of-the-art methods, let us describe the parameter learning process and the conclusions it brings.

Learning the Probabilities. The algorithm presented in Section 4 contains a number of hyper-parameters, mostly ones that determine the number of operations done in each splitting procedure. While we used some trial and error to set these values, their influence is counterbalanced, to a large extent, by the number of times these procedures are executed, i.e., the more search effort is put inside a procedure, the less it needs be called at each level of the recursion, and vice versa. Consequently, we obtained the optimal balance between the algorithm operations by learning the probabilities to perform a certain action at each level of the recursion such that the lowest number of printable subparts per number of object-plane cutting operations is attained.

More specifically, we learned the probabilities to select each procedure, P_{base}, P_{tip}, P_T, as well as a fourth option of randomly sam-

pling a cutting-plane P_{rand}, and performing additional calls within a recursion level, P_{call} and q. This fourth option is added for the purpose of comparing our geometry-derived planes with the random planes current methods use [YYT*17, WQZ*18]. We make this sampling more cost-effective by limiting the distribution of the planes offset to the objects extents (while their orientation is uniformly sampled from the unit sphere).

The search alternated between finding the best P_{base}, P_{tip}, P_T, and P_{rand} given P_{call} and q in one step, and the other way around in the second. The search for P_{base}, P_{tip}, P_T, and P_{rand} was exhaustively done in a 0.1 grid spacing sampling $[0,1]^4$, whereas we used a spacing of 0.005 in $[0.95,1]$ for P_{call}, and 0.04 in $[0.6,1]$ for q. We used a heterogeneous training set containing 40 models with variable complexity (requiring 2-13 cuts), face count ($3k$ to $30k$), and sources (scanned organic objects and CAD-based models). The objects were from multiple sources; the Thingi10K dataset [ZJ16], AIM@SHAPE shape repository [SMKF04], and Stanford 3D scanning repository [TL96].

Figure 9 shows the optima found in the optimization landscape computed in this exhaustive search, and Table 1 reports the optimal values found. The procedure selection probabilities P_{base}, P_{tip}, P_T, and P_{rand} do not show dependence on the number of cuts permitted nor the critical protrusion angle. Based on these probabilities, it is clear that the optimal strategy favors the base- and tip-extraction procedures over the T-junction and random plane splitting. The low P_T can be explained by the sparsity of T-junctions in objects, however the low P_{rand} indicates that our approach of deriving the cutting-planes from geometric analyses is more beneficial than randomly sampling them, despite the higher in-procedure computational costs.

The table also shows that the parameters of the probability to perform additional procedure calls within a recursion level, P_{call} and q, increase as the number of permitted cut operations increases. This finding is expected since these parameters control the mean exit time of the uninterrupted recursion.

Finally, the table also indicates that our algorithm is sensitive to the specified CPA and results in more subparts when a more stringent angle of 135° is used, compared to the 150° case.

Evaluation. We compared our method with the methods in [HLZCO14, WQZ*18, YYT*17] which are the closest works to ours. However, each of these works defines its goals somewhat differently, and hence the comparison is done carefully, with these differences in mind.

The method in [HLZCO14] sets a stringent printing condition of $\theta_{\text{crit}} = 90^\circ$, however, its decomposition is approximate and hence its goal in practice is to minimize the amount of support-material needed. In contrast, our method searches for support-free decompositions. Table 2 shows that our method offers a significant speedup of x10 compared to this method, and although it generates fully support-free partitions it results in a small increase in the number of subparts it generates (15% on average). We believe this increase is justifiable since our approach altogether eliminates the need for expensive support-material or specialized hardware such as an additional extruder.

The method in [YYT*17] minimizes the support-material

Model	#Faces	135°	150°	160°	Comp.
					[WQZ*18], 160°
Airplane	31k	0:32, 2	0:25, 2	0:26, 2	1:57, 7
Armadillo	69k	10:52, 9	5:01, 6	6:40, 4	7:09, 10
Bearing	3k	0:17, 3	0:06, 3	0:17, 3	3:48, 10
Deer	18k	4:51, 4	0:42, 3	0:27, 2	1:45, 6
Gargoyle	50k	2:52, 5	2:46, 3	2:06, 2	4:18, 5
Knot	6k	$\emptyset$	0:30, 13	0:52, 8	1:39, 14
Octopus	2.6k	0:44, 8	0:23, 7	0:23, 5	1:48, 8
Sculpture	12k	0:49, 7	0:38, 4	0:20, 3	1:23, 10
Tree	23k	1:51, 3	0:23, 2	0:24, 2	3:22, 8
					[HLZCO14], 90°
Inuksuk	25k	3:10, 3	0:19, 2	0:25, 2	10:00, 3
Lamp	10k	1:52, 6	0:19, 4	0:16, 3	10:00, 4, RS
Helix	3.5k	0:57, 8	0:30, 5	0:17, 4	16:00, 5, RS
Hands	23k	2:25, 3	1:10, 2	1:02, 2	11:00, 5, RS
Genus-3	13k	0:37, 3	0:22, 3	0:31, 2	10:00, 3, RS
					[YYT*17], 135°
Dolphin	16.5k	0:57, 3	0:35, 2	0:26, 2	10:38, 3
Homer	10k	0:25, 2	0:07, 2	0:12, 2	1:12, 2
Horse	16k	0:32, 3	0:25, 2	0:23, 2	7:25, 3
Fertility	24k	4:39, 5	1:29, 4	1:13, 4	8:11, 2, RS

Table 2: *Comparison with other methods. Columns 3-5 show the running-time (minutes:seconds) and number of subparts produced by our method for different θ_{crit}. Last column shows the numbers produced by [WQZ*18] on an i7 3.6GHz machine with $\theta_{crit} = 160^\circ$, by [HLZCO14] on an i7 3.4GHz with $\theta_{crit} = 90^\circ$, and by [YYT*17] on an i7 3.4GHz machine with $\theta_{crit} = 135^\circ$. We labeled the non-printable results by Require Support (RS). The stop criteria was $3k$ cuts for $\theta_{crit} = 135^\circ$, and $1k$ cuts for for $\theta_{crit} = 150, 160^\circ$. Optimal probabilities were chosen according to the search above (for $\theta_{crit} = 160^\circ$, we use the same parameters as $\theta_{crit} = 150^\circ$). The maximum search depth was 15. Under these parameters, no result was found for the knot model at $\theta_{crit} = 135^\circ$.*

Model	Partitioned		Supported	
	Time	Material	Time	Material
Bearing	1:15	6335	1:20	6598
Sculpture	1:32	3284	1:54	3897
Hands	1:41	3924	1:58	4766
Fertility	2:02	6455	2:20	7104
Tree	1:20	3249	1:37	4318
Genus	2:32	8412	2:40	11066

Table 3: *Time and material added when printing the objects with a support material. The time is hour and minutes, and the filament length is in millimeters. We used the popular Slic3r object slicing software to generate the support. $\theta_{crit} = 150^\circ$*

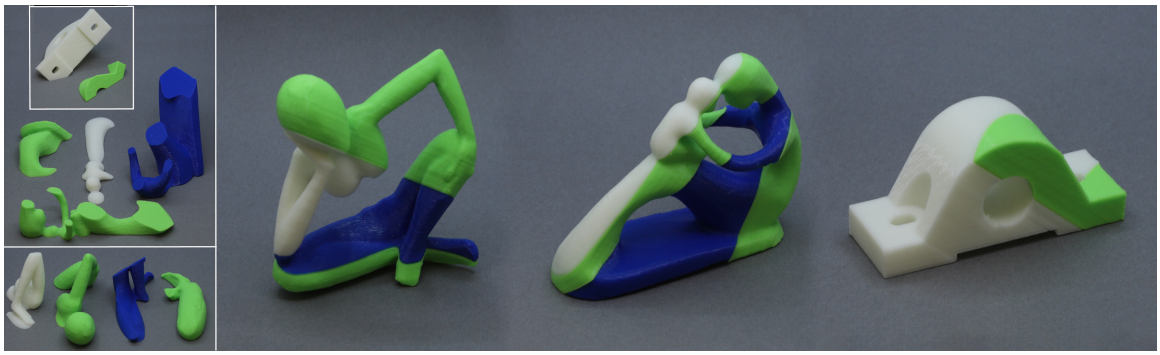


Figure 10: *3D printed objects. Different PLA colors are used for revealing the different subparts, which are shown before gluing on the right. A critical angle of* $\theta_{crit} = 150^{\circ}$ *was used in these examples.*

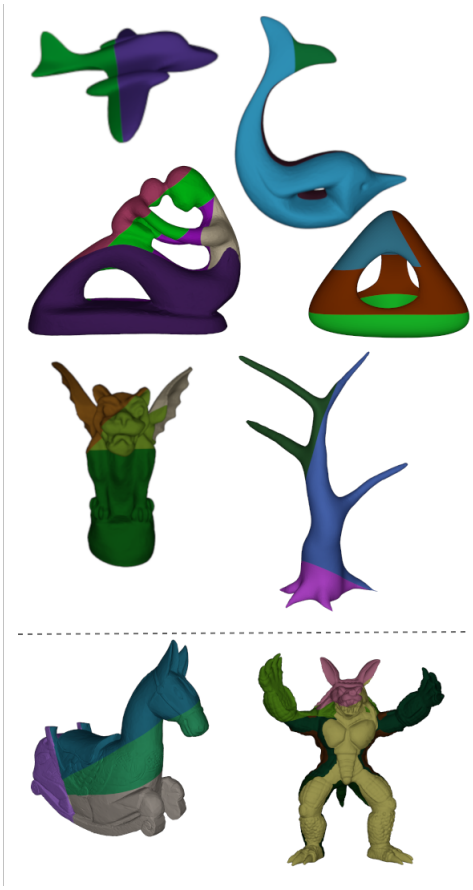


Figure 11: *Rendered images of partitions produced by our method. Objects on the top were partitioned with* $\theta_{crit} = 135^{\circ}$ *and the bottom ones with* $\theta_{crit} = 150$*. The different subparts are encoded with distinct colors.*

needed by conducting an evolutionary randomized search in which cutting-planes are generated randomly. This method often results in support-free object partitioning. Table 2 shows that, when it comes to fully-printable solutions, our method generates the same or a smaller number of subparts, as well as operates significantly faster. T

The method in [WQZ*18] guarantees the printability of the decomposed skeleton, however this method also focuses on printing shell models. The latter leads to an increase in the geometric complexity of the models, and hence this method produces a higher object count. Assuming the material saving in low-density in-fills is sufficient, our method offers both a lower number of subparts as well as operates faster.

In comparison to both these works, our method performs considerably less plane-cuts, which is the most time-consuming step. Wei et al. [WQZ*18] explore 8k skeletal partitions, and to evaluate each one, do many plane-cuts. Yu et al. [YYT*17] carry out 2.5k fitness evaluations, each consisting of a few object-plane cuts. Our method performs 1k-3k object-plane cuts, and reuses many of these operations during the recursion.

Figure 10 shows actual 3D-printed objects produced by our method. The subpart extracted from the right side of the Sitting Sculpture model has two separate contact areas with the base-plane. This resulting partitioning is unique to our method. Despite the particular cuts needed to minimize the number of subparts, no printing or gluing issues were encountered. Figure 11 shows additional object decompositions produced by our method.

Table 3 reports the printing-time as well as the amount of material consumed when printing the objects with support structures versus our partitioned solution. This tests shows a speedup of more than 15% and a saving of more than 10% filament on average. Figure 12 visually compares these two alternatives in terms of the surface quality they produce. The regions directly attached to the support structures show a noticeable amount of scratches and support leftovers despite our considerable effort in removing them using a sharp craft knife. The seams between the subparts in our results are apparent, but are somewhat less conspicuous thanks to the fact that the seams are typically aligned with the printing layers.

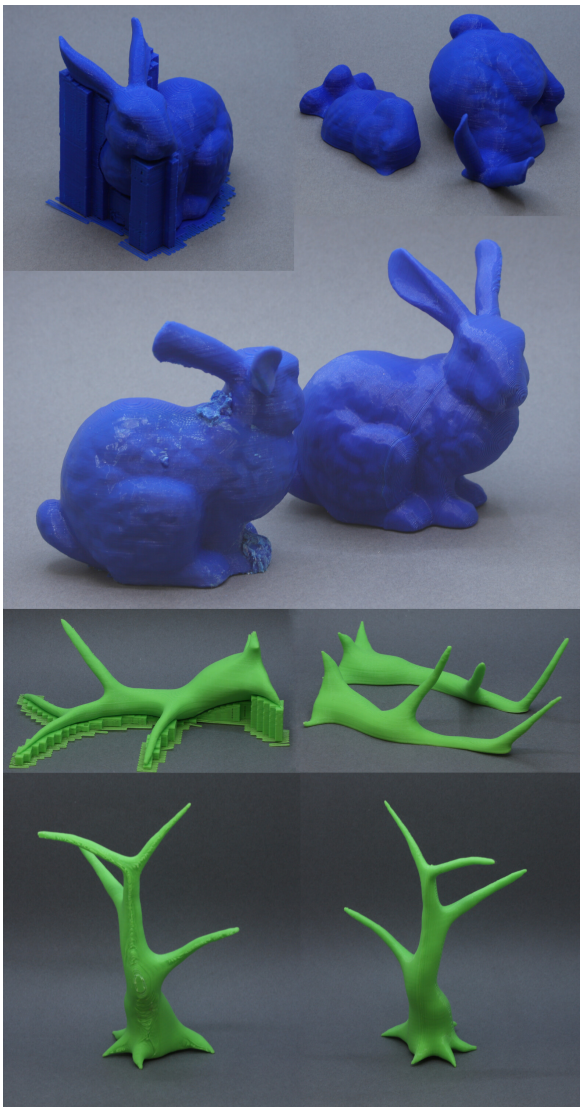


Figure 12: *A visual comparison showing the region where the support structures were removed, next to the glued seam appearing in our result.*

6. Discussion

We presented a new method for partitioning objects in order to enable their support-free 3D printing. The method searches various geometric constellations commonly found in objects and resolves them either by extracting a printable subpart, or by lowering the object's complexity. These object splitting procedures are carefully designed to cope with general geometry and operate efficiently. In addition, we defined a complete set of criteria for the printability of an object and showed its completeness. These criteria take into account the specific printer's ability to cope with sloped surfaces. Evaluation of the method shows that it results in similar or lower number of subparts and runs faster than state-of-the-art methods.

Unlike several existing methods, our method is geared towards generating fully support-free partitions. This strategy alleviates the need for specialized support-material or additional printing and post-printing hardware.

The main limitation of our method stems from the NP-hardness of the problem it tackles. Indeed, the solutions it computes may have more parts than the optimal solution, and in case of very large and complicated shapes it may require long running times in order to come-up with a solution. As shown in the the previous section, it offers attractive running times and number of parts on benchmark objects.

7. Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedback. This work was partially funded by the European Research Council, ERC Starting Grant 337383 "Fast-Filtering".

References

[CCA*12] CALÌ J., CALIAN D. A., AMATI C., KLEINBERGER R., STEED A., KAUTZ J., WEYRICH T.: 3d-printing of non-assembly, articulated models. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 130:1–130:8.

[Cha84] CHAZELLE B.: Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. *SIAM J. Comput. 13*, 3 (1984), 488–507.

[DAB18] DEMIR I., ALIAGA D. G., BENES B.: Near-convex decomposition and layering for efficient 3d printing. *Additive Manufacturing 21* (2018), 383 – 394.

[DHL14] DUMAS J., HERGEL J., LEFEBVRE S.: Bridging the gap: Automated steady scaffoldings for 3d printing. *ACM Trans. Graph. 33*, 4 (July 2014), 98:1–98:10.

[DPW*14] DEUSS M., PANOZZO D., WHITING E., LIU Y., BLOCK P., SORKINE-HORNUNG O., PAULY M.: Assembling self-supporting structures. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 214:1–214:10.

[DWW*18] DAI C., WANG C. C. L., WU C., LEFEBVRE S., FANG G., LIU Y.-J.: Support-free volume printing by multi-axis motion. *ACM Trans. Graph. 37*, 4 (July 2018), 134:1–134:14.

[FM01] FEKETE S. P., MITCHELL J. S. B.: Terrain decomposition and layered manufacturing. *International Journal of Computational Geometry & Applications 11*, 06 (2001), 647–668.

[HBA13] HILDEBRAND K., BICKEL B., ALEXA M.: Smi 2013: Orthogonal slicing for additive manufacturing. *Comput. Graph. 37*, 6 (Oct. 2013), 669–675.

[HL17] HORNUS S., LEFEBVRE S.: *Iterative carving for self-supporting 3D printed cavities*. Research Report RR-9083, Inria Nancy - Grand Est, July 2017.

[HLZCO14] HU R., LI H., ZHANG H., COHEN-OR D.: Approximate pyramidal shape decomposition. *ACM Transactions on Graphics, (Proc. of SIGGRAPH Asia 2014) 33*, 6 (Nov. 2014), 213:1–213:12.

[HT90] HOLSCHNEIDER M., TCHAMITCHIAN P.: Les ondelettes en 1989. *P.G. Lemarie ÌĄ (Ed.)* (1990).

[JSXZ18] JIANG J., STRINGER J., XU X., ZHONG R. Y.: Investigation of printable threshold overhang angle in extrusion-based additive manufacturing for reducing support waste. *International Journal of Computer Integrated Manufacturing 31*, 10 (2018), 961–969.

[JWL*18] JADOON A. K., WU C., LIU Y., HE Y., WANG C. C. L.: Interactive partitioning of 3d models into printable parts. *IEEE Computer Graphics and Applications* (2018), 1–1.

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: Partitioning models into 3d-printable parts. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 129:1–129:9.

[LSZ*14] LU L., SHARF A., ZHAO H., WEI Y., FAN Q., CHEN X., SAVOYE Y., TU C., COHEN-OR D., CHEN B.: Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph. 33*, 4 (July 2014), 97:1–97:10.

[OZB18] OH Y., ZHOU C., BEHDAD S.: Part decomposition and assembly-based (re) design for additive manufacturing: A review. *Additive Manufacturing 22* (2018), 230 – 242.

[PWLSH13] PRÉVOST R., WHITING E., LEFEBVRE S., SORKINE-HORNUNG O.: Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4 (July 2013), 81:1–81:10.

[SFLF15] SONG P., FU Z., LIU L., FU C.-W.: Printing 3d objects with interlocking parts. *Comput. Aided Geom. Des. 35*, C (May 2015), 137–148.

[SMKF04] SHILANE P., MIN P., KAZHDAN M., FUNKHOUSER T.: The Princeton shape benchmark. *Shape Modeling International* (June 2004), 167–178.

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph. 31*, 4 (July 2012), 48:1–48:11.

[TL96] TURK G., LEVOY M.: The stanford 3d scanning repository, 1996.

[TM84] TOR S. B., MIDDLEDITCH A. E.: Convex decomposition of simple polygons. *ACM Trans. Graph. 3*, 4 (Oct. 1984), 244–265.

[VGB14a] 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.

[VGB*14b] VANEK J., GALICIA J. A. G., BENES B., MĚCH R., CARR N., STAVA O., MILLER G. S.: Packmerger: A 3d print volume optimizer. *Comput. Graph. Forum 33*, 6 (Sept. 2014), 322–332.

[WQZ*18] WEI X., QIU S., ZHU L., FENG R., TIAN Y., XI J., ZHENG Y.: Toward support-free 3d printing: A skeletal approach for partitioning models. *IEEE Transactions on Visualization and Computer Graphics 24*, 10 (Oct 2018), 2799–2812.

[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 Trans. Graph. 32*, 6 (Nov. 2013), 177:1–177:10.

[WZK16] WANG W., ZANNI C., KOBBELT L.: Improved Surface Quality in 3D Printing by Optimizing the Printing Direction. *Computer Graphics Forum* (2016).

[XXY*15] XIE Y., XU W., YANG Y., GUO X., ZHOU K.: Agile structural analysis for fabrication-aware shape editing. *Comput. Aided Geom. Des. 35*, C (May 2015), 163–179.

[YYT*17] YU E. A., YEOM J., TUTUM C. C., VOUGA E., MIIKKULAINEN R.: Evolutionary decomposition for 3d printing. In *Proceedings of the Genetic and Evolutionary Computation Conference* (New York, NY, USA, 2017), GECCO '17, ACM, pp. 1272–1279.

[ZCT16] ZEHNDER J., COROS S., THOMASZEWSKI B.: Designing structurally-sound ornamental curve networks. *ACM Trans. Graph. 35*, 4 (July 2016), 99:1–99:10.

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10k: A dataset of 10, 000 3d-printing models. *CoRR abs/1605.04797* (2016).

[ZLP*15] ZHANG X., LE X., PANOTOPOULOU A., WHITING E., WANG C. C. L.: Perceptual models of preference in 3d printing direction. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 215:1–215:12.

[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.