Additive Manufacturing 21 (2018) 383–394

Contents lists available at ScienceDirect

Additive Manufacturing

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

Near-convex decomposition and layering for efficient 3D printing

İlke Demir*, Daniel G. Aliaga, Bedrich Benes

Purdue University, 305 N. University St., West Lafayette, IN, United States

ARTICLE INFO

Keywords:
3D printing
Convex decomposition
Model segmentation
Computational geometry
Optimization

ABSTRACT

We introduce a novel divide-and-conquer approach for 3D printing, which provides automatic decomposition and configuration of an input object into print-ready components. Our method improves 3D printing by reducing material consumption, decreasing printing time, and improving fidelity of printed models. An input object is decomposed into a set of components obtained by a near-convex segmentation that minimizes an energy function. Then the configuration phase provides a robust algorithm to pack the components for an efficient print job. Our approach has been tested on both simulated models and real-world printed objects. Our results show that the framework can reduce print time by up to 65% (fused deposition modeling, or FDM) and 36% (stereolithography, or SLA) on average and diminish material consumption by up to 35% (FDM) and 10% (SLA) on consumer printers, while also providing more accurate objects.

1. Introduction

Designing and 3D printing objects is a rapidly growing area. It enables experts and casual users to build a large variety of custom objects. Moreover, manufacturing technologies have progressed tremendously enabling multi-color, multi-material, and even multi-function printing. The most common approach for 3D printing is by gradually putting down thin layers that build up to form the 3D object (e.g., stereolithography (SLA) or fused-deposition modeling (FDM)). However, this process requires additional support structures for overhangs. Moreover, at inclined angles the printed surface is irregular and not smooth (i.e., the stair-stepping effect). Also, the size of the object and the printing speed are limited by the characteristics of the printer.

3D printers have both limitations and advantages depending on the coherency between the printer features and the model geometry. Instead of relying only on improvements of the 3D printing technology, we provide a solution that optimizes the model in order to maximize that coherence by segmenting the model into easily printable components.

In addition to the previous work on converting synthetic models to 3D printable objects [1–5], researchers pursue developing 3D manufacturing approaches based on segmenting the to-be-printed objects (e.g., [6–11]), and on using clever support structure methods (e.g., [12,13]). Our key idea is to simultaneously reduce printing time, decrease material consumption, and increase fidelity by decomposing an object into homogeneous material components each of which obeys a set of shape criteria and by configuring them on a printing layer (Fig. 1).

Our input is a polygonal model as shown in the overview in Fig. 2. During the *decomposition phase*, our parameterized approximate convex decomposition algorithm partitions the initial clusters into an optimal set of components. Our method seeks a low number of near-convex components with no near-horizontal faces (i.e., faces are either horizontal or have an angle more than a printer-defined threshold from the horizontal). Firstly, imposing exact convexity makes the problem NP-hard and creates unnecessarily many pieces (see Section 2 and Figs. 4f and 5b). Instead we pursue near-convex components which relaxes the convexity constraint and results in fewer components that are still self-supporting. Further, it reduces the consumption of support material. Secondly, having faces as either horizontal or exceeding the threshold angle enables the printer to create smooth exterior surfaces for the given object. Moreover, making the faces to be coherent with the printing directions improves the quality of segments. After the decomposition, we prepare the components for printing in a *configuration phase*. The components are laid out for 3D printing using as little vertical printer head displacement as possible. Since significant amount of the printing time is spent for moving the printing bed down (or the printer head up) and given that multiple components can be printed simultaneously, this configuration step reduces the total printing time.

We demonstrate the results of our automatic method on a variety of objects. An example in Fig. 1 (also used in [8,11]) shows the overall quality improvement by 15% on near-horizontal surfaces. For that model, our approach conserves 49.4% of the material and reduces the printing time by 50.3%. We also applied our method to different objects

* Corresponding author.
E-mail address: idemir@purdue.edu (İ. Demir).

https://doi.org/10.1016/j.addma.2018.03.008
Received 24 February 2017; Received in revised form 15 February 2018; Accepted 7 March 2018
Available online 09 March 2018

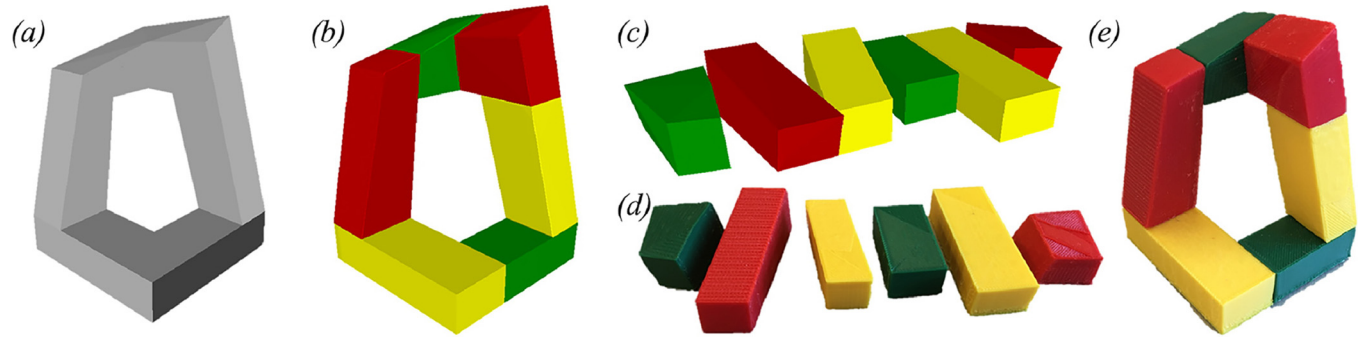

Fig. 1. Decomposition for 3D printing: Input model (a), our automatic near-convex decomposition (b), configuration that will be printed (c), individual printed components (d), and the final printed and assembled object (e).

and Table 2 shows the resulting time and material reductions.

Our main contributions include:

- a near-convex decomposition method for 3D models that improves the quality of a printing process by increasing smoothness of angled surfaces, reducing material consumption, and decreasing the printing time,
- an automatic configuration method to pack a number of printable components on the printing surface so as to reduce print time and support consumption, and
- an implementation of a fully automatic pipeline that processes an input 3D model for an optimal printing process by segmenting the model and creating a configuration of the segmented parts.

2. Previous work

Divide-and-conquer has been an important approach in fabrication for decades. Many designs are composed of parts with different materials and/or colors that are manufactured separately and then assembled with different techniques (welding, gluing, riveting, bolts and nuts, etc.). This is also the case in recent studies on 3D printing of complex designs [5].

2.1. Decomposition, conversion, and support

Segmentation is commonly used to print heterogeneous objects with conventional 3D printers. Prevost et al. [2] carve inside objects to modify the position of the center of gravity. To print them, they divide the objects into small sections depending on the number of voids inside the objects. Considering the layer-by-layer printing approach, some objects, depending on their structural properties [14], cannot be printed without support material. To overcome this limitation, Hu et al. [8] decompose shapes into pyramidal parts to eliminate usage of support material and to decrease printing time. Zhou et al. [9] also segment into primitives, but use cylindrical assumptions. In another study, Dumas et al. [15] decrease the amount of support material by printing scaffolds composed of bridges and columns instead of printing supporting volumes with lower densities. The approach of Vanek et al. [12] generates automatic support structures for a 3D object. In a different approach, Wang et al. [3] convert the mesh into skin-frame structures to reduce the cost of printing.

2.2. Puzzles and packing

In recent years, segmentation and 3D printing have been frequently employed to generate customized 3D puzzles. Lo et al. [16] propose a design approach to create puzzles from given shapes composed of 3D polyominoes and Xin et al. [17] follow a similar approach to create Burr puzzles. In another study, Song et al. [18] focus on the recursive property of interlocking puzzles. Although these methods focus on 3D printing, their segmentation does not consider printing time nor material consumption. Rather, the focus is on creating an assembly procedure that follows a specific order of increasing difficulty. Recently, Attene [13] proposes a disassemble and pack approach for 3D printing. However, this approach only considers convex objects.

2.3. Segment and pack

Luo et al. [6] segment real-world objects for easy manufacturability. Close to our approach is the work of Vanek et al. [7] who also segment and pack, employing an optimization for the packing stage. Their method first converts the input to shells and then uses a closed loop of merging and packing to obtain a maximum tightness. In contrast, our method works with the full volume of the input, employs no assumptions about the mesh, and eliminates the support material. In a follow-up work, Yao et al. [10] also segment and pack the components to reduce the support structure used, but they do not consider angled surfaces (i.e., rotation) as a part of their optimization. Also, their packed form may still need support structures due to thin and overhanging structures. Finally, their number of partitions is fixed during the optimization, which limits the quality of segments. Recently, Chen et al. [11] also come up with a coupled optimization approach that segments and packs. However, they assume pyramidal structures as well, which restricts the possible improvements. Also, their voxelization dependency impacts the appearance of surface details and overall fidelity. Ezair et al. [19] analyze the orientation of pieces for 3D printing and Zhou et al [20] transform the models into boxes and looks for a continuous folding sequence. While the latter reduces the required printing volume, fidelity is lost.

2.4. Convex decomposition

It is the problem of computing a decomposition of a 3D model T by partitioning it into a minimal set of convex sub-surfaces [21]. Chazelle et al. [22] prove that computing such a decomposition is an NP-hard problem and propose various heuristics to solve it practically. Later, Lien et al. [23] point out that the proposed algorithms are impractical due to the high number of clusters. In order to provide a tractable solution, they propose to relax the exact convexity constraint and consider instead the problem of computing an approximate convex decomposition of T. They alter algorithm parameters so as to generate a partitioning $\Pi = \{\pi_1, \pi_2, \ldots, \pi_K\}$ of T with a minimal number of clusters K and verifying that each cluster has concavity lower than a threshold ϵ.

The approximate convex decomposition problem, as addressed in the previous paper, requires a sophisticated analysis of the model features, and uses plane-based bisections leading to poor decompositions. In order to improve such decompositions, Mamou and Ghorbel [24] show a volumetric solution as a novel hierarchical segmentation approach for 3D meshes. The approach starts with computing a dual graph of the mesh and clustering its vertices iteratively by successively

applying topological decimation operations, while minimizing a cost function related to the concavity and the aspect ratio of the produced clusters.

Overall, convex models are more printing-friendly, however exact convex decomposition is NP-hard, divides the model into many tiny little pieces, and thus it is inappropriate for printing. Concave packing solutions create complex structures that may be difficult to assemble. On one hand, there are solutions that require convex objects and do not provide tight packing [11,13]. On the other hand there is a solution that provides concave packing [7], but it requires converting the input object into shells. Thus, we provide a middle-ground solution as a near-convex decomposition optimization.

Unlike previous work, we guide our decomposition for less resource consumption and for less surface deviation error even when using low resolution printing. We use a near-convex decomposition of the input model followed by a reconfiguration of the parts that collectively reduces print time, improves quality, and diminishes material waste. The Approximate Convex Decomposition (ACD) [23] builds an approximation of the original shape using near-convex decomposition but it also introduces a significant deformation of the input object. Pyramidal Decomposition (PD) [8] creates a particular type of near-convex decomposition (e.g., vertical convexity), putting emphasis on base polygon selection, which is not optimized. This may still cause some wasted material in pyramidal deficit regions. Also, it is not suitable for all objects such as those with thin structures, hollow objects, and ball-and-stick figures. We highlight that ACD and PD will not yield results of similar quality to ours even with parameter tuning. Segmentation results from different approaches are demonstrated later in Fig. 5. Our approach follows the intentions of Hu et al. [8] and Zhou et al. [9], however takes into consideration accuracy of angled surfaces, flexes the constraints of pyramidal/cylindrical structures, and reduces printing resources.

3. Overview

The input to our method is an unlabeled triangle soup model and the output is a printable configuration of the model components (Fig. 2). We define a metric to evaluate how well a set of properties for 3D printing is achieved; those properties include a tight volumetric approximation, small number of components, no support material, faster print time, and less angled-surfaces leading to higher quality. Then, we describe and demonstrate an optimization process that yields a careful balance of the desired properties. After we obtain the components, we find a configuration to layer them for efficient fabrication.

3.1. Decomposition

Our algorithm takes the input triangle set $T = \{t_0, \ldots, t_n\}$ and partitions the set into disjoint triangle clusters $Z = \{z_0, \ldots, z_{N_Z}\}$, where $\cup_{a=1}^{N_Z} z_a = T$. We call each triangle cluster a *search subspace* because we found it much more efficient to subsequently decompose each subspace into printable components.

We define and formulate several properties in order to segment each subspace into printing-friendly components (Section 4.2.1). We approach segmentation as an energy minimization $E(C)$ over the components $C = \{c_0, \ldots, c_{N_C}\}$ that seeks segmentation parameters x best satisfying the component properties. After the optimization, the model is placed into a grid of voxels and the computed parameters are fed to a volumetric decomposition algorithm in order to separate the model into components.

3.2. Configuration

The segmented components are oriented and positioned for efficient 3D printing. We implement a greedy approach to lay the components on the printing rack on their largest faces and separated just far enough to prevent being printed as a single piece.

4. Decomposition

Our decomposition algorithm has two steps: search subspace creation and segmentation.

4.1. Search subspace creation

A naïve implementation to find the aforementioned component set C would iterate through all possible cuts of the voxelized volume and possibly merge unrelated parts. Such a solution would be very time demanding and would possibly diverge from a good solution in terms of fidelity. Instead, we use heuristics to compute a set of triangle clusters that contain partially similar parts of the model, if any (i.e., considering relative granularity and inter-similarity of parts, see Fig. 2 clusters). Then, finding optimum printing components within each search subspaces is a much simpler task. This partitioning does not reduce the amount of geometry. Rather, it diminishes the size of the search space by eliminating the combination of components spanning unrelated parts. In addition to benefiting our minimization, the identified clusters define a grouping beneficial for assigning color and material properties to partially similar structures within the model for multi-color/multi-material printing.

The first step of search subspace creation is to define an initial set of clusters containing similarly-shaped triangles. Inspired by [25], the dissimilarity of triangles is defined by a shape dissimilarity metric $S_{t_i t_j}$, consisting of the differences of their areas, edges and normals. If $S_{t_i t_j} \leq \tau_S$, the triangles are included in the same cluster. Based on initial experiments, we found $\tau_S = 0.6$ to work well for all of our models.

The second step of search subspace creation is to iteratively merge and split the clusters from the previous step so as to balance spatial similarity with shape similarity. For this objective, we define a spatial similarity metric between clusters z_a and z_b. If two clusters have enough ($\geq \tau_N = 0.5$) neighboring triangles, those clusters are merged, and non-neighboring triangles are split into a new cluster. Intuitively, the spatial similarity metric tells us the percentage of triangles in cluster z_a having neighbors in cluster z_b. During each iteration of this step, we compare cluster-by-cluster, mark similar clusters, and merge-split at the end of each iteration, until convergence. We also highlight that our method uses the same threshold parameter values for all models.

4.2. Segmentation

After we obtain the search subspaces $Z = \{z_0, \ldots, z_{N_Z}\}$, we segment each subspace z_i into components $C = \{c_0, \ldots, c_{N_C}\}$. We start this section by formulating a set of component properties to minimize and then we define the optimization's energy function.

4.2.1. Component properties

To improve printing, the components should exhibit the following properties. We formulate these properties so that a minimization over the possible segmentations reveals components that satisfy the properties. The motivation behind each desired property is shown in Fig. 3.

- *Concavity:* To guide the optimization to produce near-convex, or convex, components, we define a concavity measure $V(T_*)$ for any triangle set T_* that uses a function $P(x)$ which projects triangle vertices onto the set's convex hull. We wish to minimize this measure because such components tend to use less (or no) support material and they are easier to print [8] (Fig. 3a).

$$V(T_*) = \arg\max_{x \in T_*} ||x - P(x)|| \quad (1)$$

- *Surface angles:* We seek components that have faces (i.e., triangles)

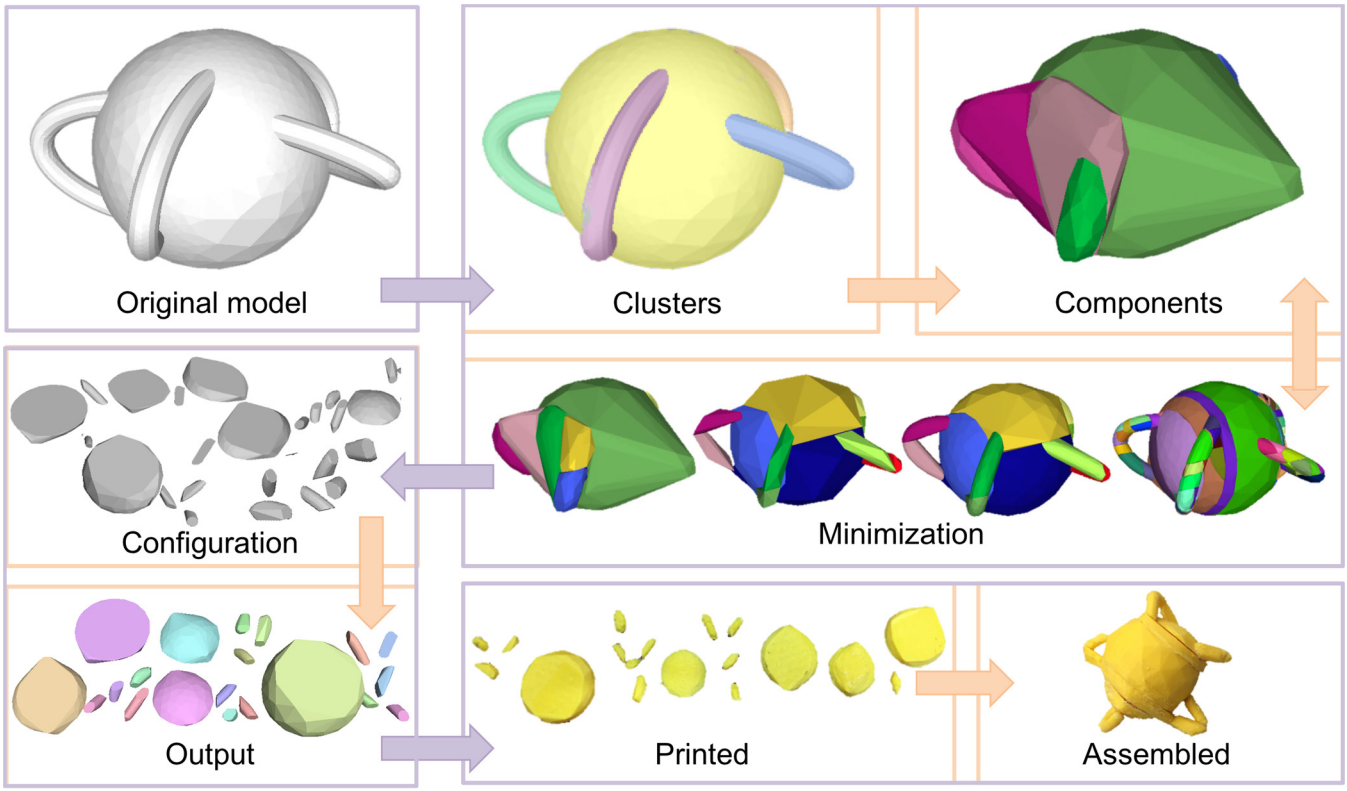

Fig. 2. System pipeline: A 3D mesh is first decomposed into clusters and then optimized for optimal components. Afterwards, the components are configured for an efficient layout. Finally, printed and assembled to produce the final physical object.

that are parallel to a base plane or angled more than a threshold from the base plane. Without loss of generality, we generally assume the base plane to be horizontal because most printers provide higher resolution in the *X*–*Y* (horizontal) axes versus in the *Z* (up) axis. Avoiding faces that are "near horizontal" improves the surface fidelity by reducing the stair stepping effect (see Figs. 13 and 3b) that results from the resolution characteristics of typical printers.

For this property, we define a metric $\Delta(T_*)$ that accumulates the difference of angles between face *f*'s and base plane *B*. A zero value for the metric implies all faces are parallel to the base plane or are angled beyond a threshold $\tau_\angle$ from the base plane. Otherwise, the metric provides a measure of how near the triangles are to a base plane orientation. The metric uses $\angle(\cdot)$ to denote the angle of a face with respect to the horizontal and the binary function $NH(\cdot)$ to determine if a face should partake in the metric. The function $NH(\cdot) = 1$ only when the face is close to the orientation of the base plane. Note that we put the largest face as the base polygon, which the optimization already assumes when calculating the surface normal angles. $\angle(B)$ is needed, because the components are not rotated until the packing phase starts.

$$\Delta(T_*) = \sum_{f_i \in T_*} \mathrm{NH}(f_i)\left|\angle(f_i) - \angle(B)\right| \tag{2}$$

$$\mathrm{NH}(f) = \begin{cases} 1, & \text{if } \angle(f) \le \tau_\angle \\ 0, & \text{otherwise} \end{cases}$$

- *Sizes and numbers:* With our decomposition, we also seek for a reasonable and automatically-determined balance between size and number of components (see the discussion at the end of Section 2), to prohibit under/over segmentation (Fig. 3c). Also, the components should be printed in compliance with the size and accuracy of the printer (i.e., too small parts are not accurately printed, or a component with a large size in at least one dimension cannot fit in the printing bed). As one solution, we want to minimize the inter-cluster size variance, where $s(x) = [height_x, width_x, depth_x]$ and μ_s is the mean size of all components in that cluster.

$$S(T_*) = \|s(T_*) - \mu_s\| \tag{3}$$

- *Deviation:* The components should collectively represent the overall object, thus the deviation from the original model should be as small as possible to improve the model fidelity (Fig. 3d). For this purpose we employ a cost function $A(T_1, T_2)$, similar to the one in [24]. It uses the concavity and aspect ratio of the newly computed triangle set T_1 relative to the original triangle set T_2, which is obtained by the unification of the vertices of the graph representation. In essence, the metric provides an estimate of the deviation of the new model from the original model, thus we want this cost to be as small as possible to preserve fidelity.

4.2.2. Energy function

Our approach treats segmentation as an energy minimization over the components. Our method defines an objective function $E(C)$ as a weighted sum of component concavity $V(c_i)$ (Eq. (1)), surface angle sum $\Delta(c_i)$ (Eq. (2)), size variance $S(c_i)$ (Eq. (3)), and shape deviation $A(c_i, T)$ relative to original model *T*. It is defined over the components $C = \{c_0, \ldots, c_n\}$ where each subspace $z_k \in Z$ contains mutually exclusive components($c_i \subset z_k$) with the aforementioned properties.

$$E(C) = \frac{1}{N_C}\sum_{i=0}^{N_C} w_a A(c_i, T) + w_v V(c_i) + w_\delta \Delta(c_i) + w_s S(c_i) \tag{4}$$

Then, we minimize the objective function (4) by changing the input parameters *x* of our segmentation algorithm $\Omega(\cdot)$ to solve the following problem:

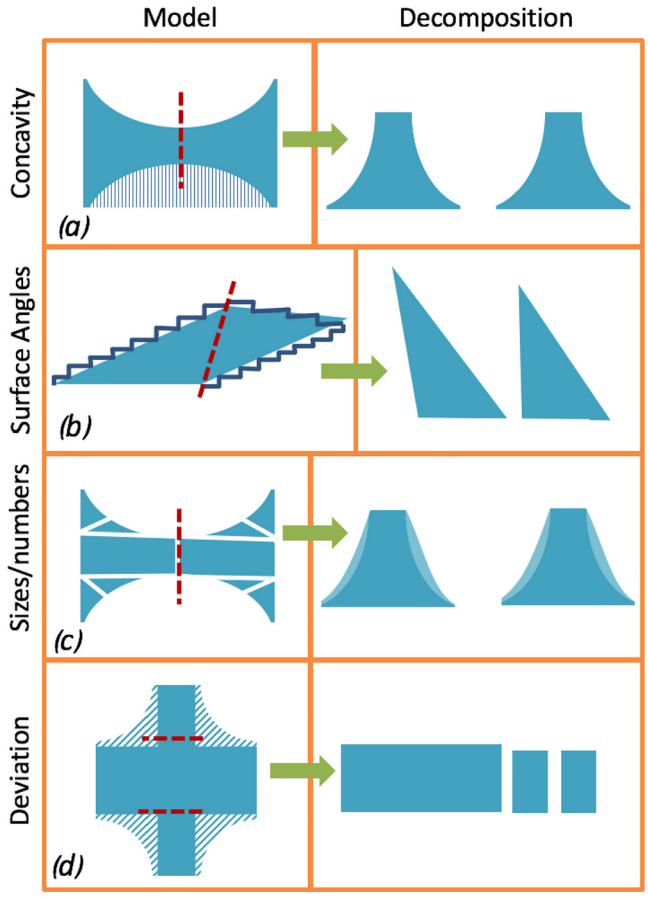

Fig. 3. Component properties: Convex components need less support material (a). Better surface quality can be achieved by avoiding near-horizontal angles (b). Balancing convexity and size/number of components prevent over-segmenting (c). Minimizing deviation increases model fidelity (d). The red dashed lines indicate the cut line. The combed area in (a) indicates the support structure, and the combed areas in (c and d) indicate the model deviation.

$$\begin{aligned} x &= \underset{C\in\Omega(x,Z)}{\operatorname{argmin}} E(C) \\ \text{such that } T &= \sum_{k=0}^{NZ}\Omega(x, z_k) = \sum_{k=0}^{NZ}\sum_{j=0}^{NC_k} c_j, \\ \text{and} \quad & x_l \le x \le x_u \end{aligned} \tag{5}$$

The second part of Eq. (5) states that the segmentation algorithm Ω uses input parameters x to decompose each search subspace z_k into a set of non-overlapping components C (e.g., the union of all components equals T) and the input parameters x are constrained to the range $[x_l, x_u]$.

The next section explains our segmentation algorithm Ω and Section 5 provides implementation details about input normalization and the weights in Eq. (4).

We highlight that balancing the aforementioned component properties is key for having the most suitable components for printing. Fig. 4 shows a partitioning of the object Fig. 4g into progressively more components. The partitioning ranges from a few components shaping the convex hull (Fig. 4a) to an almost exact convex decomposition (Fig. 4f). Our minimization seeks to minimize all the aforementioned properties which results in a more desirable solution such as Fig. 4e. Following those decompositions, Volumetric Hierarchical Approximate Convex Decomposition (VHACD) [24] creates a solution space ranging from the convex hull of the object (with maximum object deformation and minimum number of pieces) to the exact convex decomposition of the object (with minimum object deformation and maximum number of pieces) showing that the solution we are searching for lies in between, which encourages us to use VHACD as a tool to obtain the cuts through the model.

4.2.3. Near-convex decomposition

Our segmentation method minimizes the energy term (Eq. (4)) over the components. The model is processed and the volumetric cuts are realized by VHACD algorithm of Mamou and Ghorbel [24]. Even though the approach of Mamou and Ghorbel is more robust than previous work for approximate convex decomposition, it needs a number of parameters to be tuned per model and per use-case (i.e., resolution, number of points per convex hull, plane downsampling, etc.). We optimize the decomposition by automatically finding a set of parameters that satisfy the component properties mentioned in Section 4.2.1. We treat the parameters of VHACD as the input vector of the minimization. As the solution converges, the parameters produced as an output of the minimization give an optimum segmentation for the model. Thus, the user does not need to adjust any parameters or segment manually.

In our problem definition, the objective function is a black box; thus, we cannot provide explicit first or second derivatives of the energy function. Hence, we use the BOBYQA algorithm of Powell [26] to explore the parameter space of VHACD. The algorithm works by guessing a trust region from the bounded parameter space, and then exploring different input vectors for the minimization by interpolating the values within the search space. Since the parameter space of VHACD is well-bounded by the algorithm itself, this makes it easier to define the energy function in the implementation of the minimization.

Fig. 5 demonstrates components from different convex decomposition algorithms on an input model (Fig. 5a). Exact convex decomposition over-segments the model (Fig. 5b), ACD [23] significantly deviates from the input model (Fig. 5c), VHACD [24] over-smooths corners for convexity (Fig. 5d), and PD [8] creates near-horizontal surfaces that decreases accuracy of printed surfaces (Fig. 5e). In contrast, our approach (Fig. 5f) balances convexity, sizes, deviation, and accuracy in order to obtain the optimum set of components for printing.

4.3. Configuration

Segmentation divides the model into components that follow the properties from Section 4.2.1. However, the components need to be

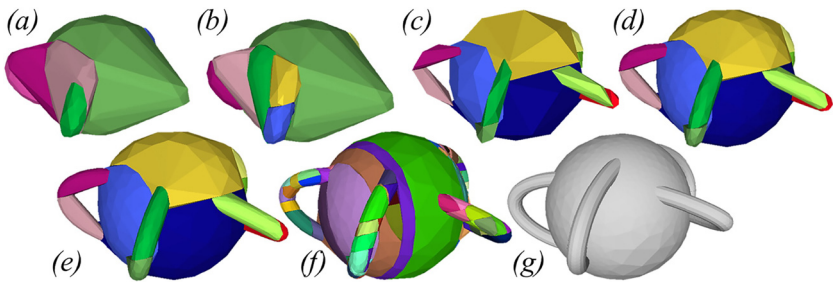

Fig. 4. Progressive partitioning: An object (g) is partitioned from few (a) to many (f) components. Our method enables to find optimal solutions such as (e) which is practical for our printing objectives.

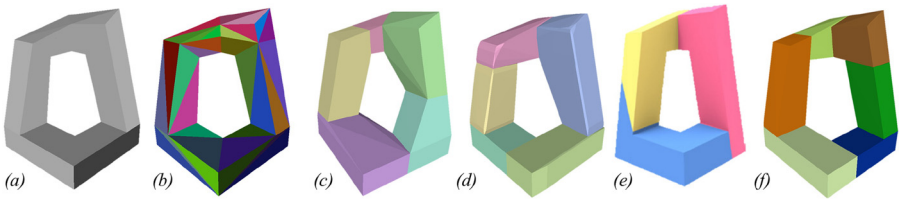

Fig. 5. Comparison: We demonstrate the segmentation results with other convex decomposition approaches. The original model (a), and the segmentations of exact convex decomposition (b), ACD [23] (c), VHACD [24] (d), PD [8] (e), and our approach (f).

Table 1
Processing times: We compare the pre- and post-processing times of our approach versus the cleaning time for models printed by an FDM printer.

Model	Segmented			Original	
	Computation	Assembly		Cleaning	
Cctv	2.5	< 1	6	10	50.3
Block	6.8	2	20	12	19.1
Chair	5.2	2	16	9	34.7
Ball	18	5	20	30	12.6
Inukshuk	3.2	< 1	7	8	31.9
	Time (min)	Time (min)	# of parts	Time (min)	% of material

adequately positioned and oriented on the printing bed, in order to produce an efficient configuration for printing. The configuration must be carefully set up because of the following reasons.

- Most of the printers have different accuracies in different axes, which makes printing efficiency rotationally dependent. Thus, small angles in low accuracy axes reduce the model fidelity (e.g., the staircase effect).
- The extrusion speed in different axes also vary, and that effects the time-efficiency of the print job. Thus, the slowest axis should be used as less frequent as possible.
- The components must be as self-supporting as possible to reduce the amount of support material consumed. Thus, the components should be placed on their largest face in accordance with other constraints.

We implement a configuration algorithm that places the components on the printer tray. For each component, we first find its oriented bounding box (OBB), and then rotate it so that the smallest dimension of the bounding box is oriented along the z-axis (i.e., the slowest extrusion speed axis). Then, we orient the object so that the largest face is lying horizontally on the print bed. We also find the second dominant axis of each component's OBB, and set that to be placed along the y axis by further rotating the component to have a better packing. Then, we use a greedy packing algorithm to place all components close enough to be efficiently printed, but far enough not to merge the components while printing. For this purpose, we iterate over the components and do a pairwise spatial comparison of their OBB's. If they overlap, we set $min_y(c_i) = max_y(c_j) + d_n$, where $min_y(c_i)$ and $max_y(c_i)$ denote the minimum and maximum y values of the OBBs. In other words we translate component c_i to the right of c_j separated by a threshold of d_n. The use of OBBs for placement is actually a good approximation in our case, since our components are approximately convex.

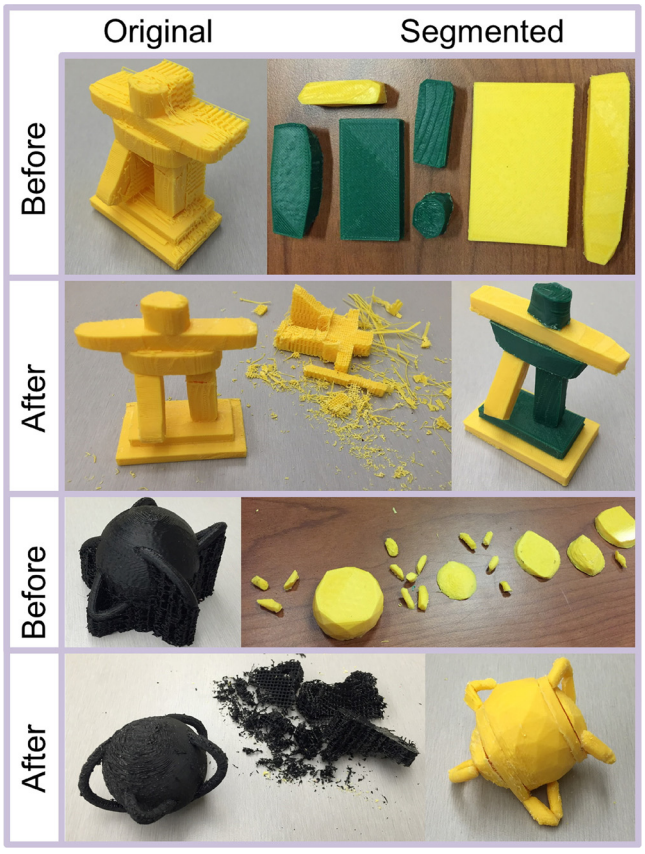

Fig. 7. Original vs. segmented models: We show the original and segmented forms of the model, before and after post-processing (removing support material and assembling, respectively).

The described method produces non-overlapping placement of components organized in a compact fashion for printing. Because the minimization algorithm keeps the number of components low, a greedy approach is not too expensive computationally. If the number of

Fig. 6. Example objects: We show side by side the printed results of the original and the segmented models.

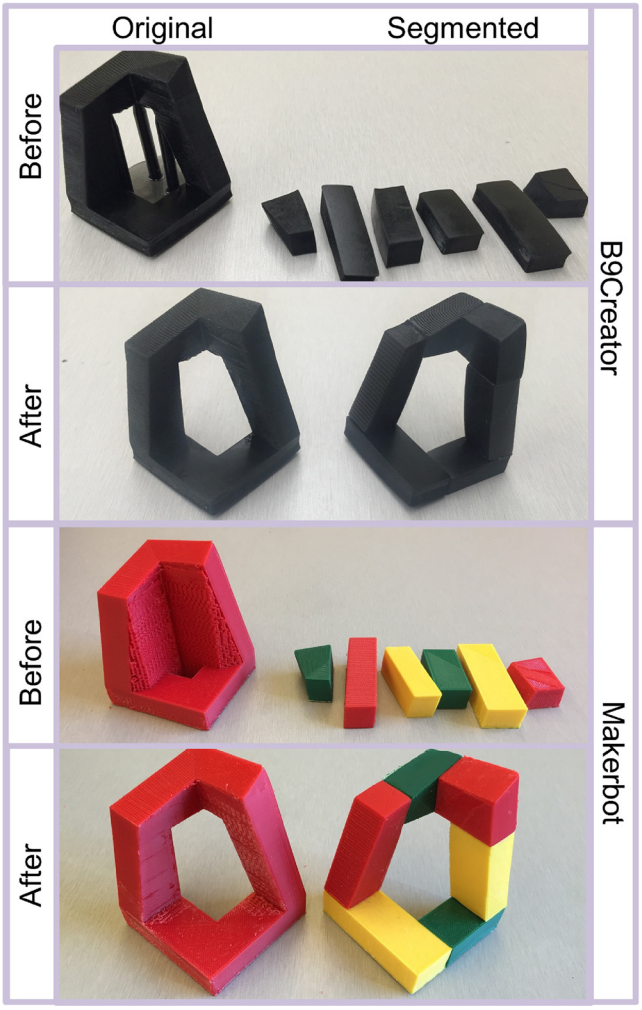

Fig. 8. Printer comparison: The original and segmented forms of the model, before and after post-processing, with B9Creator and MakerBot printers.

components would have been a large number, an alternative packing algorithm, like [7,13], would be needed. Although it is true that simple packing with OBBs can leave some redundant space, (i) we do not have many pieces to configure and (ii) the configuration is 2D; thus we leave the optimization of 3D packing as a future work.

5. Implementation

Our framework is implemented in C++ using Qt, OpenGL, and `dlib-ml` library and is executed on an Intel i7 Desktop PC with NVIDIA GTX 680 graphics card. The input to our approach is a triangle mesh model and we use `obj` file format. We use an extended version of the `glm` library to process and store the model information, which includes pre-computed attributes like triangle size and neighbor information.

5.1. Input

In order to support a general decomposition approach, we choose an unstructured collection of triangles, or "triangle soup", as input. The input may be a well-connected mesh but such is not necessary. If the input consists of non-triangular polygons, we perform a triangulation. Our formulation is done based on triangles, and the optimization is done on the volumetric representation. Hence, it would be trivial to process non-polygonal models by feeding that representation directly to a relaxed version of VHACD, to get an initial polygonal model. Note that this option skips the clustering step.

5.2. Optimization

The optimization of E (Eq. (4)) is implemented using the `find_min` function in the `dlib` library, implemented in `c++`. The elements in the energy function are weighted to normalize the effect of all properties. Based on the approximate intervals of the parameters, we normalize by using $w_z = 1.0$, $w_a = 0.2$, $w_\omega = 0.0055$, and $w_s = 0.1$. Note that these weights are same for all models and there are no user-specified terms in any part of our approach.

Although VHACD has a number of parameters to be fine-tuned, we have concluded by experiment that the crucial parameters are the resolution of voxelization during segmentation, the minimum volume per convex hull estimation, and the maximum number of vertices to be included in each convex hull. Since the ideal boundaries and step sizes of VHACD parameters are different, we have normalized resolution by 0.00001, minimum volume per convex hull by 10,000, and maximum numbers per convex hull by 0.25. After the normalization, the lower and upper limit vectors become $x_l = [1.0, 0.0, 1.0]$ and $x_u = [200.0, 100.0, 256.0]$ respectively. Finally, VHACD provides the opportunity to exploit an adjustable bias towards cutting the model along the dominant axis which gives us the chance to automatically adjust for the best cut direction for slicing. Thus, having a bias for cuts in the direction of fabrication (e.g., as in orthogonal slicing directions of [27]), makes it easier to match and glue the components.

5.3. Printers

For the real-world objects in the results section, we have fabricated some of the test objects using two different types of printers: a MakerBot Replicator 2X (an FDM printer) and a B9Creator (an SLA

Table 2
Evaluation: Comparison of the original and the segmented models, their printing times and material consumption, per model and per printer type.

Model	FDM (time: min, material: g)				SLA (time: min, material: ml)			
	Original		Segmented		Original		Segmented	
	Time	Mat	Time	Mat	Time	Mat	Time	Mat
Cctv	130	29.75	64	14.99	70	21.52	21	20.67
Block	100	23.97	89	19.38	36	3.96	11	3.79
Chair	97	21.44	63	14.00	24	3.5	9	3.48
Ball	111	19.48	73	17.02	21	2.00	11	1.79
Cup	125	30.19	84	17.71	20	0.67	6	0.58
Deer	78	11.98	29	5.25	74	4.14	14	3.61
Sledge	182	41.88	112	27.81	65	6.33	7	6.01
Homer	162	37.85	120	27.81	28	8.01	22	8.76
Toy	96	22.40	86	18.62	30	7.38	17	8.04
Inukshuk	148	35.37	98	24.98	52	23.82	14	21.92

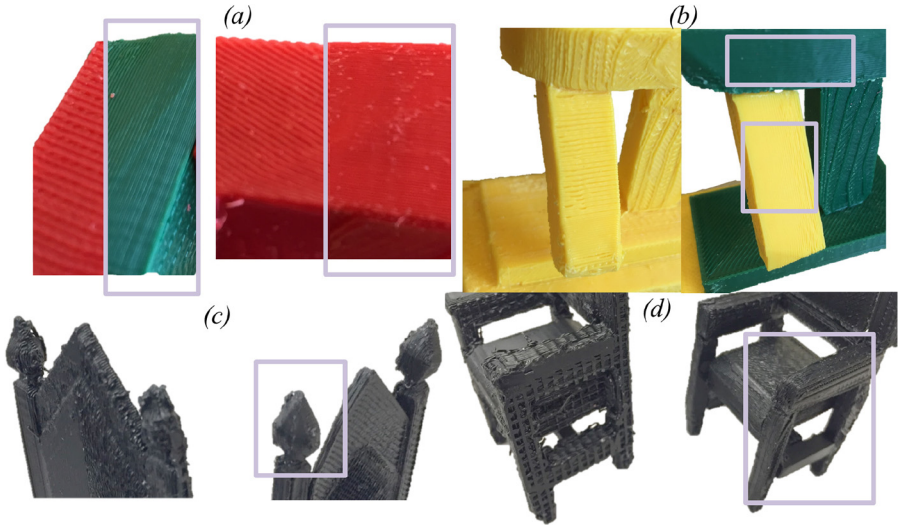

Fig. 9. Improvements: Our results are highlighted within boxes. The avoidance of angled surfaces improves surface fidelity (a and b), having no support material protects the deterioration of the object (c), convexity gets rid of the support material (and its scars) from the inside and outside of the objects (d).

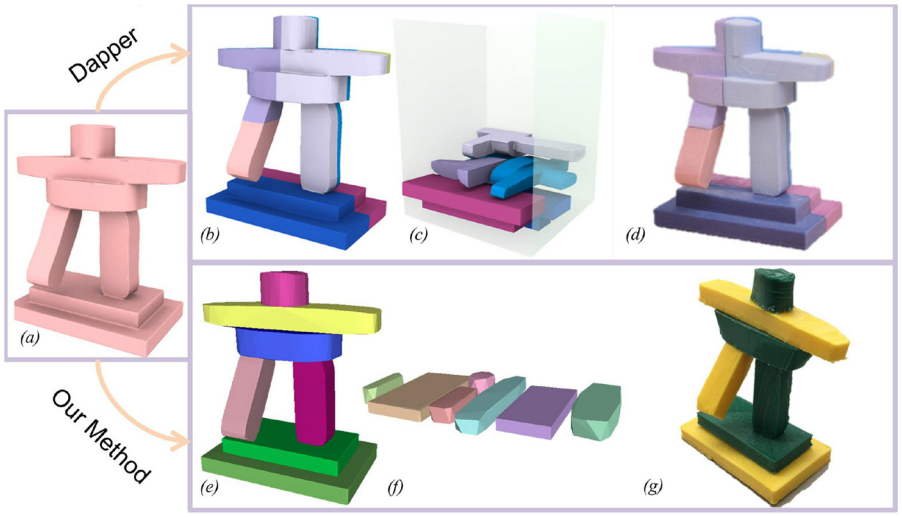

Fig. 10. Comparison with Dapper [11]: Our segmentation is more coherent with the printing direction, is able to provide better fidelity on the leg, and eliminates support material.
Images a–d courtesy of [11].

printer). In our FDM printer, ABS material is used and printer settings are: infill: 20%, number of shells: 2, resolution: 0.2 mm. In our SLA printer, we used the thickness 200 μm with B9R-2-Black resin, to show that even the fastest configuration produces higher fidelity results than the printed non-processed models.

5.4. Assembly

For assembling the segmented parts, we color-code the segments in the layout and glue the corresponding faces of the printed segments. Overall, the assembly process of our method took much less time than cleaning the support material from original model (Table 1). We also show the amount of material to be cleaned (in terms of percentage of the original mass) and the number of pieces to be assembled per model. We highlight that having reasonable assembly times implies that the models are not over-segmented, as one would expect in an exact convex decomposition case. Table 1 shows computation time to process a model, assembly time of the printed segmented model, compared to cleaning time of the printed original model.

We also make sure that gluing areas are compatible with each other, because (i) the convex decomposition does not neglect any volume, and (ii) the deviation from the original model is minimized by the energy function. The former ensures that there is no missing volume between the components (so that the gluing areas are compatible surfaces) and the latter ensures that the edges of the gluing areas follow the actual bounding volume of the object. Since the gluing surfaces are known, another option is to embed interlocking pieces (lego-like structures) to these areas for assembly. This easy extension is not currently implemented and it is left as future work.

5.5. Simulation

To simulate the layered printing process for our analysis (Fig. 13), we used MakePrintable services to convert the GCodes produced by our Makerbot 2X printer device driver into a 3D `.stl` model. Then, we computed the Hausdorff distance (in Meshlab) for comparison and visualization of the 3D simulated printed model and the original model.

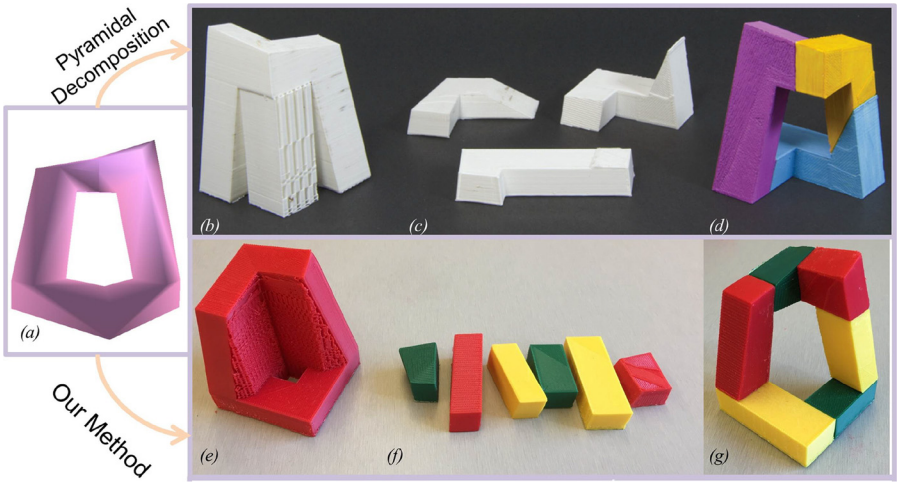

Fig. 11. Comparison with PD [8]: Our segmentation is more coherent with the printing direction, is able to provide better fidelity on the upper surface of the building, and eliminates support material.
Images b–d courtesy of [8].

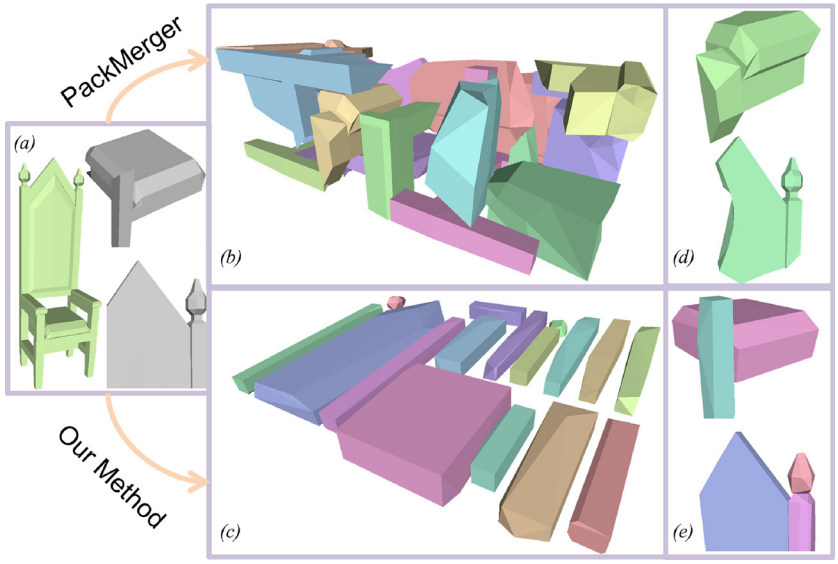

Fig. 12. Comparison with PackMerger [7]: Ready-to-print version of the chair (a) segmented by PackMerger (b) and our method (c). The zoom ins compare the original model (cropped for visibility), PackMerger (d), and our components (e).

6. Results and applications

We applied our framework to 3D models obtained from AIM@SHAPE and Archive3D, as well as manually modeled objects. The complexity of our 20+ models varies from 200 polygons to 250 K polygons with 23.9 K polygons on the average. Our approach works on both solid and shell forms with no assumptions, since we work on convex hulls of the shapes. We state the time and material consumption statistics for the two printing machines mentioned above. The typical preprocessing time to automatically segment and configure a model for printing is 15 min for a medium complexity model. We also demonstrated some of our results in our supplementary video.

Printed examples. We compare printed versions of the original models and segmented models in Fig. 6; with better approximated surfaces, and multi-color support. We also show real-world printed examples of some models, in their original form before and after removing the support structure, and as a segmented model before and after assembly (Fig. 7). As seen in those figures, our approach prevents wasting material, and provides higher fidelity objects, with multi-material support. Note that, even if the approximated surface is highly curved (as in Fig. 7, bottom), our decomposition finds segments that connect well, even after printing with accumulated printing errors.

Printer comparison. We show the results of the same object from different printers in Fig. 8. Our approach decreases printing time for both types of printers, but the drops in SLA are considerably higher (see Table 2). SLA print time reductions are 56.5% for the pyramidal segmentation [8] and 64% for our approach. FDM print time reduction averages to 34% with our method. Regarding the support material, when segmentation is employed, support structures not printed in FDM yield a 32% reduction in material on average. SLA printers do not automatically calculate the needed support structure. Hence, for our experiments we have manually added minimum estimated support structure for the original models, ending up in 10% less material for the segmented models.

Improvements. We show close ups of improved models. Fig. 9a and b shows the increased fidelity of near-horizontal surfaces (our green and red colored result in the highlighted boxes is side by side to the original one). Fig. 9c shows overcoming the blending of support structure with fine details of the object. Fig. 9d shows a better approximation and printing for interior structures. In all figures, we highlight our

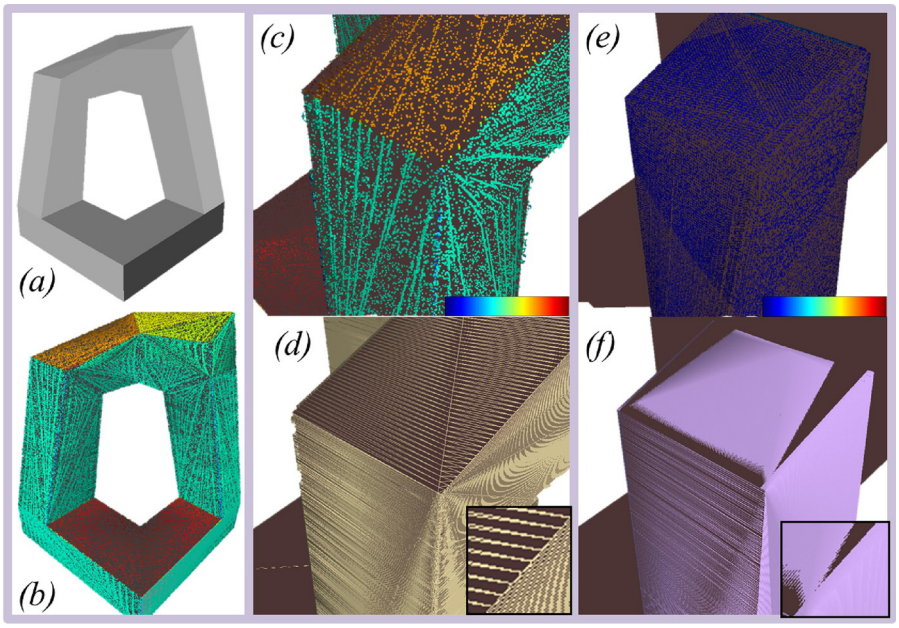

Fig. 13. Model fidelity: Printed versions of the original model (a), a color-coded representation of the Hausdorff distance between the simulated printed model and the original model (b), close up of (b) in point cloud (c) and mesh forms (d), and close up of simulated printed segmented model in point cloud (e) and mesh forms (f). Our method can increase the fidelity on near-horizontal surfaces.

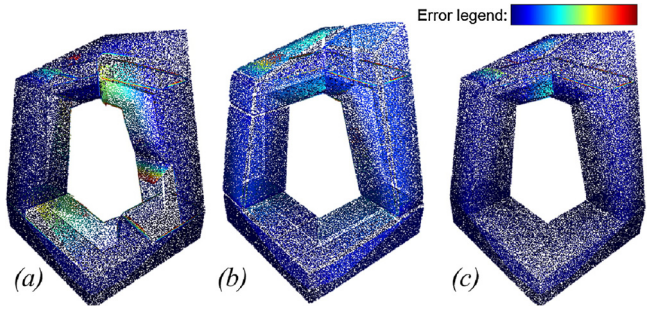

Fig. 14. Additional comparison: We also calculated the deviation from the original model of different segmentation approaches: ACD [23] (a), VHACD [24] (b), our approach (c). As color-coded, our model has the highest fidelity.

improvements with boxes surrounding corresponding areas.

Comparison to previous work. 3D printing research is continually improving. Dapper [11] is a previous work that uses a prior 3D printing segmentation approach [8] (i.e., it uses their code; see Section 5 of Dapper) and improves upon it. Moreover, they compare to PackMerger [7]. Dapper and PackMerger, in fact, have similar overall segmentation and packing objectives. Contrary to previous work (Dapper, PackMerger), our work uses a new-to-3D printing volumetric segmentation method (i.e., VHACD) and furthers such a methodology.

We compare to related work with a similar aim, in particular to the example model of Dapper [11], pyramidal decomposition [8], and PackMerger [7], using the same challenging representative objects in Dapper and Pyramidal Decomposition and similar objects to those in PackMerger. First, observe that our segmentation results in providing higher fidelity on the front of the right leg of the Inukshuk model (Fig. 10 and zoom-in in Fig. 9b) and on the top of the CCTV building (Fig. 11 and zoom-in in Fig. 9a). Second, a more semantically meaningful decomposition (observe the colors) is obtained. Third, a set of components coherent with the printing direction as discussed in Section 5.2 is obtained. This improvement can be seen overall for each piece of the model in Fig. 10, on the slanted cut of the building in Fig. 11, and in the zoom-ins for the chair model in Fig. 12. Fourth, Dapper and PackMerger require support material and tend to build the volume in the z-direction that is slower to print. And finally, some of those methods need topologically clean meshes, however we do not put any assumptions on the meshes. Also, even though PackMerger includes the gluing size during optimization, the glued areas are not smooth (Fig. 12b and d), thus the printing errors make it harder to match those docking areas. For the model in Fig. 11, our decomposition provided 6 parts, reduced the printing time by 50.6%, and material by 49.7%. Note that all of our "material savings" are 100% as compared to [8], since we eliminate support material. However the results of [8] actually state that there is 1–44% waste in their printed objects (see Table 1 of the paper) due to the pyramidal deficit regions. For the model in Fig. 10, our decomposition provided 70% height reduction, and printed in 33% less time and with 30% less material. Their paper does not provide all statistics about that model, but note that their height minimization and packing is just to reduce the print time, and they do not aim for eliminating support material. Dapper further reduces height by 17% and support materials by 34% as compared to PackMerger (with a total reduction of 32% and 53%). Their solution as published requires support material to hold-in-place their multiple layers of components, which is measured by "gaps" in their paper (same for PackMerger). Our decomposition enabled printing in 64.9% of the original time with 65.3% of the original material in Fig. 12, where PackMerger [7] overall reduces printing time by 18.8% and material by 29.3%. Comparing all those qualitative and quantitative results, we can conclude that our approach significantly improves the efficiency of fabrication.

The accuracy of the printed segmented models and printed original models is compared in Fig. 13. First, we simulate segmented printed objects and original printed objects, in both cases exporting the resulting objects in a layered form. Then, we compute the distances between the simulated printed model and the original model, and between the simulated segmented printed model and the original model. We encode the error by coloring the point cloud by Hausdorff distance rendered on the original model. Observe that the printed model does not approximate the original model (Fig. 13c) as well as our segmented printed model (Fig. 13e). We also superimpose the printed versions in wireframe to demonstrate the density and accuracy of our approximation in Fig. 13f, compared to Fig. 13d, to prove that better approximations are possible with the same printer (the dark triangular area shows the overlapping surface, instead of the stepping effect). The coloring in the point cloud version indicates that our algorithm decreased the overall error more than 35% based on the Hausdorff distance of sampled surface points. We have not evaluated based on a measurement of the real printed models, because parameters contributing to this surface error is more constrained in simulation space

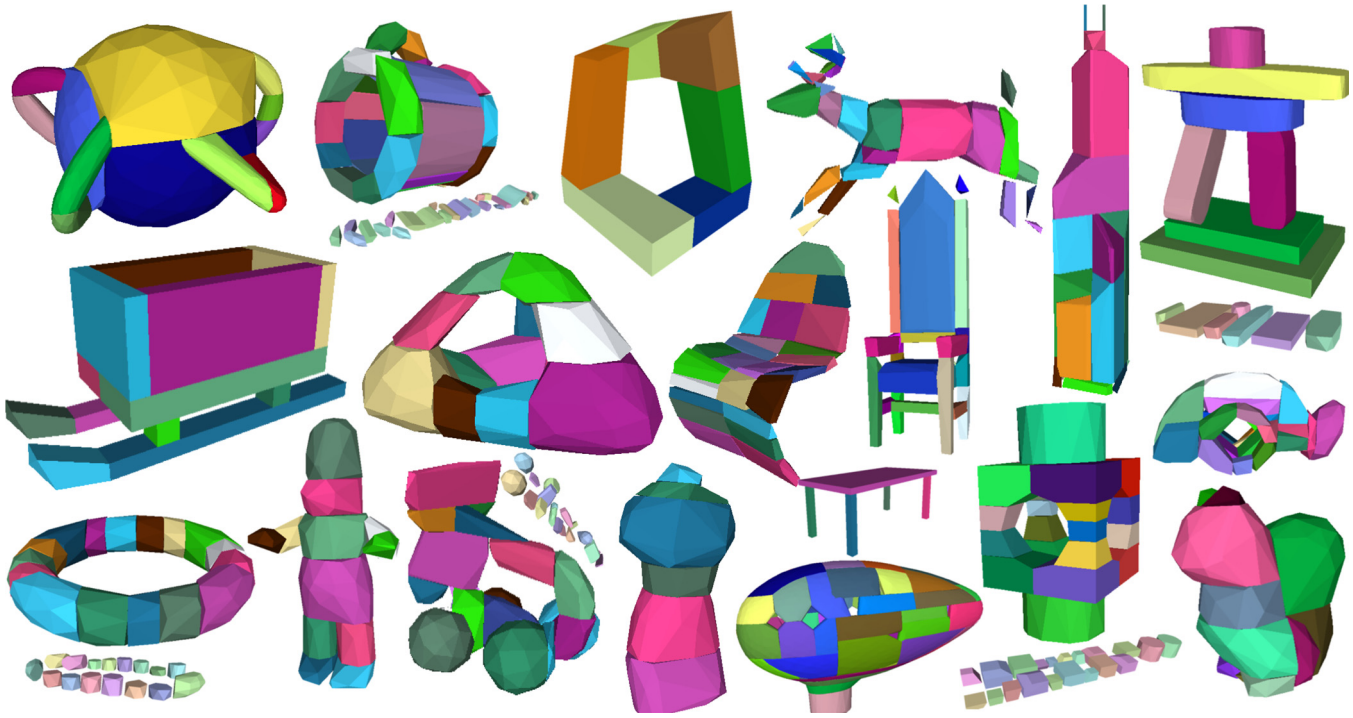

Fig. 15. Database: A subset of our model database in the segmented form. Some configurations are also depicted in insets.

than the printing-measuring space, ending up in more consistent evaluations.

We calculated the deviation of different approximate decompositions (ACD [23], VHACD [24] with typical parameter values, and our approach) from the original model. Our result is closer to the original model by at least an additional 15% accuracy (Fig. 14) over the other methods.

We summarize the results of our approach on various models in Table 2. We show the printing time and material consumption reductions. On average, our approach can reduce the material consumption by 35% and print time by 64% compared to the original model. We have also demonstrated a subset of our model database and some example corresponding segmentations in Fig. 15.

Limitations. With regard to limitations, if the model is thin and curvy our algorithm may not provide good results even for the best approximation due to printer limitations. Another limitation is the limited size of the printing bed. In that case the configuration can have additional layers instead of one layer, and support material between the layers or sequential print jobs would be needed.

7. Conclusion and future work

We present an automatic framework for improving printing of 3D models by a decomposition and configuration approach. The generated examples are compared to original models by simulations and by printing, demonstrating that our approach reduces the consumption of printing resources as well as it improves the printing quality by constructing better approximations of the original mesh. We used our approach on a variety of models and documented the qualitative and quantitative improvements on the final printed results.

As future work, we believe that our algorithm is flexible to fit any segmentation target. The approach is adaptive and can be used to exploit any printer-specific property, or any other segmentation task that puts some conditions on the desired components. We believe that one future work is to make this process more computationally efficient and to carry our implementation to the 3D printer drivers in order to exploit our automatic divide and print approach. Our implementation uses multi-core processing for the segmentation part but we did not optimize for GPU computing. We also believe that a better minimization implementation that records and stores some previous evaluations would reduce the processing time. Another improvement is to add connecting docks (e.g., lego-like keys) to our segments. Finally, while glue-products today yield extremely strong bonds (e.g., epoxy glues are even used inside car engines), we leave analyzing the effect of overall object strength to future work.

Acknowledgments

This research was funded in part by National Science Foundation grants “MRI:Development of a Next-Generation 3-D Printer for Smart Product Design - Purdue PolymerMakers” IIS 1726865, “CDS&E: STRONG Cities - Simulation Technologies for the Realization of Next Generation Cities” CBET 1250232, and “CGV: Medium: Collaborative Research: A Heterogeneous Inference Framework for 3D Modeling and Rendering of Sites” IIS 1302172. We would like to thank Jorge Garcia Galicia for running PackMerger [7] on the example models for comparison, and Ulas Yaman for the initial motivations of the project.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.addma.2018.03.008.

References

[1] M. Eigensatz, M. Kilian, A. Schiftner, N.J. Mitra, H. Pottmann, M. Pauly, Paneling architectural freeform surfaces, ACM Trans. Graph 29 (4) (2010) 45:1–45:10.
[2] R. Prévost, E. Whiting, S. Lefebvre, O. Sorkine-Hornung, Make it stand: balancing shapes for 3D fabrication, ACM Trans. Graph. 32 (4) (2013) 81:1–81:10.
[3] W. Wang, T.Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, X. Liu, Cost-effective printing of 3D objects with skin-frame structures, ACM Trans. Graph. 32 (6) (2013) 177:1–177:10.
[4] D. Yamanaka, H. Suzuki, Y. Ohtake, Density aware shape modeling to control mass properties of 3D printed objects, SIGGRAPH Asia 2014 Technical Briefs, SA ’14, ACM, New York, NY, USA, 2014 pp. 7:1–7:4.
[5] J. Hergel, S. Lefebvre, 3D fabrication of 2D mechanisms, Comput. Graph. Forum 34 (2) (2015) 229–238.

[6] L. Luo, I. Baran, S. Rusinkiewicz, W. Matusik, Chopper: partitioning models into 3D-printable parts, ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31 (6) (2012).
[7] J. Vanek, J.A.G. Galicia, B. Benes, R. Měch, N. Carr, O. Stava, G.S. Miller, Packmerger: a 3D print volume optimizer, Comput. Graph. Forum 33 (6) (2014) 322–332.
[8] R. Hu, H. Li, H. Zhang, D. Cohen-Or, Approximate pyramidal shape decomposition, ACM Trans. Graph. 33 (6) (2014) 213:1–213:12.
[9] Y. Zhou, K. Yin, H. Huang, H. Zhang, M. Gong, D. Cohen-Or, Generalized cylinder decomposition, ACM Trans. Graph. 34 (6) (2015) 171:1–171:14.
[10] M. Yao, Z. Chen, L. Luo, R. Wang, H. Wang, Level-set-based partitioning and packing optimization of a printable model, ACM Trans. Graph. 34 (6) (2015) 214:1–214:11.
[11] X. Chen, H. Zhang, J. Lin, R. Hu, L. Lu, Q. Huang, B. Benes, D. Cohen-Or, B. Chen, Dapper: decompose-and-pack for 3D printing, ACM Trans. Graph. 34 (6) (2015) 213:1–213:12.
[12] J. Vanek, J.A.G. Galicia, B. Benes, Clever support: efficient support structure generation for digital fabrication, Comput. Graph. Forum 33 (5) (2014) 117–125.
[13] M. Attene, Shapes in a box: disassembling 3D objects for efficient packing and fabrication, Comput. Graph. Forum 34 (8) (2015) 64–76.
[14] O. Stava, J. Vanek, B. Benes, N. Carr, R. Měch, Stress relief: improving structural strength of 3D printable objects, ACM Trans. Graph. 31 (4) (2012) 48:1–48:11.
[15] J. Dumas, J. Hergel, S. Lefebvre, Bridging the gap: automated steady scaffoldings for 3D printing, ACM Trans. Graph. 33 (4) (2014) 98:1–98:10.
[16] K.-Y. Lo, C.-W. Fu, H. Li, 3D Polyomino puzzle, ACM Trans. Graph. (SIGGRAPH Asia) 28 (5) (2009) article 157.
[17] S. Xin, C.-F. Lai, C.-W. Fu, T.-T. Wong, Y. He, D. Cohen-Or, Making burr puzzles from 3D models, ACM SIGGRAPH 2011 Papers, SIGGRAPH '11, ACM, New York, NY, USA, 2011 pp. 97:1–97:8.
[18] P. Song, Z. Fu, L. Liu, C.-W. Fu, Printing 3D objects with interlocking parts, Comput. Aided Geom. Des. (Proc. of GMP 2015) 35–36 (2015) 137–148.
[19] B. Ezair, F. Massarwi, G. Elber, Orientation analysis of 3D objects toward minimal support volume in 3D-printing, Comput. Graph. 51 (C) (2015) 117–124.
[20] Y. Zhou, S. Sueda, W. Matusik, A. Shamir, Boxelization: folding 3D objects into boxes, ACM Trans. Graph. 33 (4) (2014) 71:1–71:8.
[21] B.M. Chazelle, Convex decompositions of polyhedra, Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing, STOC '81, ACM, New York, NY, USA, 1981, pp. 70–79.
[22] B. Chazelle, D.P. Dobkin, N. Shouraboura, A. Tal, Strategies for polyhedral surface decomposition: an experimental study, Proceedings of the Eleventh Annual Symposium on Computational Geometry, SCG '95, ACM, New York, NY, USA, 1995, pp. 297–305.
[23] J.-M. Lien, N.M. Amato, Approximate convex decomposition of polyhedra, Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling, SPM '07, ACM, New York, NY, USA, 2007, pp. 121–131.
[24] K. Mamou, F. Ghorbel, A simple and efficient approach for 3D mesh approximate convex decomposition, 2009 16th IEEE International Conference on Image Processing (ICIP) (2009) 3501–3504.
[25] I. Demir, D.G. Aliaga, B. Benes, Coupled segmentation and similarity detection for architectural models, ACM Trans. Graph. 34 (4) (2015) 104:1–104:11.
[26] M.J. Powell, The BOBYQA Algorithm for Bound Constrained Optimization Without Derivatives, Cambridge NA Report NA2009/06, University of Cambridge, Cambridge, 2009.
[27] K. Hildebrand, B. Bickel, M. Alexa, SMI 2013: orthogonal slicing for additive manufacturing, Comput. Graph. 37 (6) (2013) 669–675.