Axis-Aligned Height-Field Block Decomposition of 3D Shapes

ALESSANDRO MUNTONI, University of Cagliari and New York University
MARCO LIVESU, CNR IMATI
RICCARDO SCATENI, University of Cagliari
ALLA SHEFFER, University of British Columbia
DANIELE PANOZZO, New York University

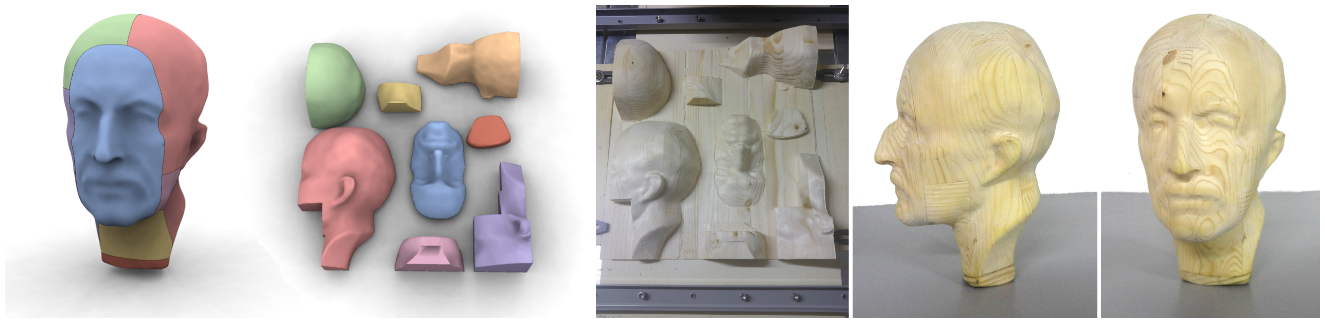


Fig. 1. We decompose general 3D geometries into height-field blocks (left), enabling their fabrication using single pass 3-axis CNC machining (right).

We propose a novel algorithm for decomposing general 3D geometries into a small set of overlap-free *height-field blocks*, volumes enclosed by a flat base and a height-field surface defined with respect to this base. This decomposition is useful for fabrication methodologies such as 3-axis CNC milling, where a single milling pass can only carve a single height-field surface defined with respect to the machine tray, but can also benefit other fabrication settings. Computing our desired decomposition requires solving a highly constrained discrete optimization problem, variants of which are known to be NP-hard. We effectively compute a high-quality decomposition by using a two-step process that leverages the unique characteristics of our setup. Specifically, we notice that if the height-field directions are constrained to the major axes we can always produce a valid decomposition starting from a suitable surface segmentation. Our method first produces a compact set of large, possibly overlapping, height-field blocks that jointly cover the model surface by recasting this discrete constrained optimization problem as an unconstrained optimization of a continuous function, which allows for an efficient solution. We then cast the computation of an overlap-free, final decomposition as an ordering problem on a graph, and solve it via a combination of cycle elimination and topological sorting. The combined algorithm produces a compact set of height-field blocks that jointly describe the input model within a user given tolerance. We demonstrate our method on a range of inputs, and showcase a number of real life models manufactured using our technique.

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh geometry models**;

Additional Key Words and Phrases: shape decomposition, fabrication

ACM Reference format:
Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. 2018. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. *ACM Trans. Graph.* 1, 1, Article 1 (January 2018), 15 pages.
https://doi.org/10.1145/3204458

1 INTRODUCTION

The advent of digital manufacturing has opened the doors to broad-based bespoke 3D object fabrication, while simultaneously introducing numerous new geometry processing challenges. Fabrication often requires decomposing the processed shapes into blocks that satisfy different sets of geometric requirements [Livesu et al. 2017; Medeiros e Sá et al. 2016]. Our work addresses one of the most challenging decomposition problems that arise in fabrication settings: *height-field block* decomposition. Height-field blocks are solids bounded by a flat base and a height-field surface defined along a direction orthogonal to, and located strictly above, this base. For fabrication purposes the resulting blocks are required to cover the outer surface of input model, but are not required to cover its entire volume. We propose the first algorithm to partition a general 3D shape into a set of height-field blocks, leaving an axis aligned void inside.

The most significant application of height-field block decomposition is 3-axis CNC milling (Figure 1). In its most common and easiest to automate setting, automatic single-pass 3-axis CNC machining is limited to fabricating height-field blocks whose base is placed on the machining tray and whose axis is orthogonal to this tray (see further discussion in Sections 2, 6). CNC machining methods can be used to carve shapes from non-layerable materials such as wood or stone. They operate across a much wider range of scales and provide higher accuracy than 3D printing. Our decomposition algorithm allows fabrication of general 3D shapes using this methodology.

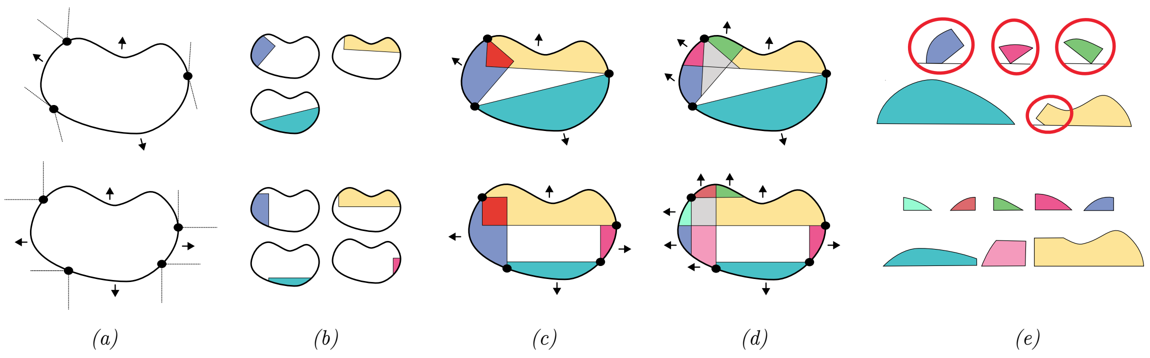
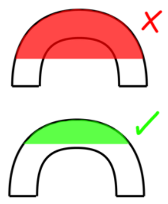


Fig. 2. From height-field segmentations to height-field solid blocks: (a) two alternative height-field segmentations defined on the boundary of a simple shape. Top one uses arbitrary directions, the bottom is constrained to the global axes; (b) the resulting minimal solid blocks with flat base defined by each height-field segment. Note that raising any of the bases would leave a portion of the boundary uncovered; (c) some blocks overlap in the interior of the shape (red extent); (d) overlaps are resolved by splitting the blocks along their supporting lines, thus increasing the number of blocks; (e) the split operation produces 4 invalid blocks out of 5 for the non axis aligned case (top right, red ovals). Inner blocks, in grey, are ignored as they are not strictly necessary to reproduce the input boundary. Constraining the height-field directions to the global axes we can always guarantee a valid solution (proof in Appendix B)

Height-block decomposition can also benefit fabrication techniques such as 3D printing, which requires supports when printing models with large overhangs. Eliminating supports reduces printing time and increases surface quality [Hu et al. 2014]. Decomposing a model into height-field blocks and printing these blocks separately allows for support-free 3D printing. Note that both applications only require the resulting blocks to cover the surface of the input model, and can often benefit from a decomposition that results in an interior voids due to obtained material savings.

Problem Statement. A height-field block decomposition of an input mesh should assign each point on the surface to a corresponding height-field block, *covering* the input surface. Blocks should never overlap and should remain strictly inside the input surface (see inset).



Additionally, to reduce manufacturing time and to facilitate easy assembly the number of blocks should be small and tiny blocks should be avoided. Based on these requirements, our algorithm's goal can be formulated as computing a decomposition that satisfies the constraints above while simultaneously minimizing the number of blocks and maximizing the size of the smallest block. While this specific problem setting has not been investigated before, closely related problems such as minimal pyramidal decomposition, or covering a volume by non-overlapping height-field blocks [Hu et al. 2014; Fekete and Mitchell 2001], have been shown to be NP-hard and have no known exact or approximate polynomial time solutions. To obtain a pyramidal decomposition within an acceptable computation time Hu et al. [2014] significantly relax the height-field constraints and consider only a finite set of possible height-field orientations. In our setting such unbounded relaxation is not possible, since the identified constraints are critical for manufacturing. Our major observation is that we can always guarantee a valid solution by restricting the set of possible block base and side face orientations to the major axis directions. We use this observation to develop an algorithm that is guaranteed to obtain valid decompositions that satisfy the height-block constraints exactly or up to a specified tolerance and can be computed within a feasible time frame. As shown by our results, our algorithm robustly produces decompositions with acceptable, even if sub-optimal, block counts for a range of complex geometries. Our framework supports additional fabrication-motivated constraints: users can limit block sizes to ensure that each block fits into the machining chamber, or limit block height to reduce local thickness and introduce inner cavities to save material (Figure 6).

Contribution. Our core contribution is a computational solution to both exact and controlled height-field decomposition. We demonstrate the practical applicability of our algorithm on 5 fabricated results, four milled (Figures 1 and 12) and one 3D printed (Figure 14), and compare our method against potential alternatives (Figures 3 and 15). To ensure replicability of our results and to accelerate adoption of our technique, we provide a reference open-source implementation of our algorithm in the supplemental material.

2 RELATED WORK

Our work fits into the highly active domain of geometry processing for digital fabrication [Livesu et al. 2017; Medeiros e Sá et al. 2016]. Many of these techniques require decomposing models prior fabrication based on different criteria. We complement these methods by providing a new decomposition technique that strictly satisfies constraints that arise in contexts such as 3-axis CNC milling and overhang-free printing. Below we review the related decomposition technologies.

Surface Segmentation. Numerous method had been proposed for segmenting a surface model into charts that meet some prescribed requirement [Shamir 2008]. The general frameworks they employ are focused on surface features, and do not consider volumetric constraints. While they can potentially be modified to use height-field approximation as a desired chart property, they allow for no obvious extension to address our volumetric constraints such as block overlap avoidance.

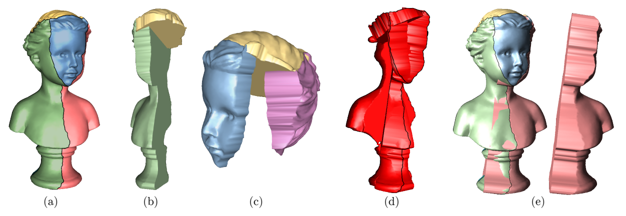


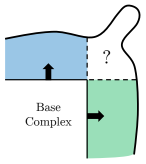
Fig. 3. Segmenting the input surface into height-field charts (a) [Herholz et al. 2015] does not account for possible intersections between the height blocks they induce. Resulting pairwise intersections visualized in (b,c). Combined intersection volume shown in (d,e).

Height-Field Surface Segmentation. Starting with the early work by Cook et al [1984], a number of height-field based surface segmentation techniques were proposed for efficient support of normal and displacement mapping during rendering. While adequate for this task they are poorly suited for our needs. First, they tend to segment shapes into large numbers of charts, e.g. Doget et al. [2000] use over 1500 height-fields to represent a human head. Such counts make fabrication unfeasible. More important, like general surface segmentation techniques these methods have no obvious extension to the volumetric setup, i.e. no obvious way to avoid or resolve overlaps between resulting blocks (Figure 2). The problem is particularly acute in 3D since generic segment boundaries are rarely planar and thus most induced blocks would have large interior overlaps, (Figure 3).

Decomposition for 3D Printing. Multiple methods have been proposed for decomposing shapes into parts to ensure that each component is small enough to fit into the printing chamber during 3D printing [Song et al. 2016, 2015; Yao et al. 2015; Alemanno et al. 2014; Luo et al. 2012; Hao et al. 2011; Medellín et al. 2007]. Shape decomposition is also used to ensure quality prints in terms of surface finish [Wang et al. 2016], to minimize the amount of material used [Vanek et al. 2014], and to achieve better mechanical properties [Hildebrand et al. 2013]. Our problem setting is distinct from those addressed by these methods, with only minimal overlap in problem setting or methodology.

Volumetric Decomposition. Computational geometry research has addressed a number of problems which bear strong similarities to our setting. Any convex volumetric decomposition can clearly be converted into a height-field block decomposition by splitting convex parts into two along an equator plane, separating faces with up and down pointing normals. Unfortunately, computing a minimal size exact convex decomposition is known to be NP-hard [Chazelle 1984; Tor and Middleditch 1984]. While practical surface-based approximate convex decomposition methods exist [Kraevoy et al. 2007], they do not produce a convex volume decomposition, as they do not prevent the convex hulls of the computed charts from overlapping, and do not prevent the height-field blocks induced by the charts from intersecting. Approximate volumetric convex decomposition, e.g. [Attene et al. 2008; Lien and Amato 2007], relaxes the convexity requirements to obtain a smaller number of parts; separating these parts along the equator may result in non-height-field blocks. Thus neither method is suitable for our needs as we require strict height-field constraint enforcement. Pyramidal decomposition aims to decompose an entire volume into height-field blocks, or pyramids. Fekete and Mitchell [2001] proved that both the 3D version of this problem and the 2D version on polygons with holes are NP-hard. To the best of our knowledge no known exact polynomial time solutions exist.

Height-field Decomposition for Fabrication. A number of papers specifically address height-field decomposition for fabrication.



Alemanno et al. [2014] propose a user assisted method for decomposing 3D shapes into height-field blocks. Their method is driven by a manually crafted inner structure, which describes the bases and the orientations of each block, fully defining the block decomposition. Overlaps between pairs of blocks, are resolved using an interlocking zipper pattern, where regions shared by multiple blocks are expected to satisfy the height-field requirement for both blocks. This assumption does not hold unless special care is taken in the construction of the inner base structure (see inset). Our approach algorithmically computes the inner structure and automatically resolves such configurations if and when they occur (Section 4.3).

Hu et al. [2014] propose an algorithm for approximate pyramidal decomposition and advocate using it for 3D printing. As they observe, a height-field block (or pyramid) can be printed standing on its base, thus maximizing its stability and ensuring that no support structures are needed to sustain it during fabrication. Since, as observed earlier, exact pyramidal decomposition is NP-hard to compute, they opt for only weak enforcement of height-field constrains and have no direct control on how far the results deviate from a desired approximation accuracy (Figure 15). The method is therefore unsuited for settings, such as 3-axis milling, where the height-field constraint needs to be strictly satisfied. Our algorithm can enforce both strict and fixed accuracy height-field constraints (Section 5).

Herholz et al. [2015] decompose free-form shapes into a set of approximate height-field surface charts for milling and molding. Candidate height-field directions are sampled from the Gaussian sphere with a saliency-based approach. As-rigid-as-possible deformation is used to enforce the height-field condition on the charts when violated. The method produces segmentations that induce overlapping height-field blocks (Figure 3). In their milling examples the authors resort to a manual process to hollow-out the back sides of each part to produce overlap-free shell parts. By using height-field blocks we remove the need for such manual backside processing and guarantee overlap avoidance.

Gao et al. [2015] propose a multi-directional 3D printing system that allows to fabricate an object around a cuboidal shell, using its six facets as printing beds. The method is only suitable for genus zero objects which can be segmented into six axis aligned approximate height-field blocks. While the algorithm seeks for a solution that minimizes the overhang angle it cannot guarantee that the resulting angles will be below any specific threshold. Our framework can provide both strictly height-field blocks and blocks with strictly constrained overhang angles regardless of topology (Figures 14, 13).

Construction Sequences/Stability. Our formulation of overlap resolution (Section 4.3) and inner void generation (Section 5) are inspired by earlier works on construction and assembly sequences, e.g. [Wu et al. 2016; Attene 2015; Hildebrand et al. 2012; Schwartzburg and Pauly 2013; Cignoni et al. 2014; Skouras et al. 2015; Lo et al. 2009; Xin et al. 2011; Song et al. 2012; Deuss et al. 2014; Zhang et al. 2016] and static/dynamic equilibrium [Bächer et al. 2014; Prévost et al. 2013; Musialski et al. 2015; Wang and Whiting 2016; Musialski et al. 2016], respectively. However, the constraints and optimization goals we address are distinctly different, with none of these approaches directly applicable to our needs.

3 PROBLEM SETTING AND OVERVIEW

Formal Problem Statement. We can formulate height-block decomposition as a semi-volumetric partition of an input geometry into blocks that satisfy the following requirements:

(1) *Base*: each block B has a flat polygonal base b
(2) *Axis*: each block has an *axis* orthogonal to its base;
(3) *Height-Field*: each block has a height-field geometry with respect to its axis - i.e. for any point p inside the block B, the line segment between p and the perpendicular projection p' of p onto the base b lies entirely inside B. Moreover the block is located strictly to one side of its base. The block is bounded by its top surface, the base and optional *side* faces orthogonal to the base.
(4) *Size Limit*: the size of each block is smaller than the build volume of the machine used to fabricate the components;
(5) *Coverage*: the top, or height-field, surfaces of the blocks jointly cover the input surface;
(6) *Non-overlapping*: blocks do not overlap;
(7) *Complexity*: the overall number of blocks is small, and each block contains as few thin features as possible.

We jointly refer to our first four constraints as *block-fabrication* constraints. They define the properties each individual block should satisfy. The block size can be further restricted along the height-field axis, as the block's *height* affects the amount of material used, and reducing it shortens fabrication time and saves costs. The coverage and non-overlap conditions are necessary to assemble the target model from these blocks. We refer to a block decomposition which satisfies all six conditions as *valid*. The last criterion, while not mandatory, is important for real-life fabrication since an excessive number of blocks would make assembly too cumbersome to attempt, and thin features make the blocks fragile.

Algorithm. To obtain the desired decomposition we need to solve a highly constrained discrete-continuous optimization problem over a very large search space, the variables of which are: the number of blocks, the direction associated with each block, and the location and geometry of their bases. The key observation behind our framework is that if we initialize our height-field blocks via intersections of the input model with axis aligned boxes, then we can always produce a valid decomposition via a finite set of boolean operations (proof in Appendix B). Specifically, given such a set of blocks that jointly cover the input surface, we can split them along the planes of their bounding boxes (Figure 2, bottom). This splitting process allows for trivial overlap elimination via duplicated block removal. The resulting set of sub-blocks satisfies all our requirements. In particular, each sub-block is an intersection of a rectangular box with a section of the input surface a priori constrained to be a height-field, i.e. a height-field block. Note that using the same process on non-axis aligned blocks would not produce the desired result (Figure 2, top).

This hypothetical splitting process provides a robust height-field decomposition of the input, but will clearly generate a large number of blocks. In practice, rather than performing all such splits at once, we perform a more restricted set of Boolean operations that use a subset of the box bounding planes, and seek to minimize the number of blocks produced. Our process preserves the height-field property of each individual block and terminates once all overlaps are removed. Thus, in the worst case it produces the same block set as the basic splitting algorithm, but in practice its outputs are drastically more compact.

We initialize and constrain the height-blocks to be inside the model by using a volume-aware block growth process. We first compute a dense set of maximal size valid height-field blocks that jointly cover the surface of the input model without intersecting it (Figure 4b). We avoid redundant and costly intersection tests by reformulating the computation of each block as an unconstrained continuous optimization problem that we efficiently solve. We extract from this set a minimal subset that covers the entire input surface while keeping the overlaps between the selected blocks small (Figure 4c). Given this compact set of blocks, we perform a sequence of Boolean operations that remove all overlaps and jointly ensure that the resulting blocks retain the height-field property and can be assembled to form the desired output (Section 4.3). In computing the sequence we seek to minimize the number of blocks produced and to maximize the smallest feature size as much as possible, to avoid the creation of fragile components that might break during fabrication. The combined algorithm strictly enforces all manufacturing constraints, while producing decompositions into small number of blocks and preserving the input surface geometry.

4 METHOD

4.1 Initialization

The orientation of the input may impact the height-field block decomposition. We fix this degree of freedom by maximizing the alignment between the surface normals and the global axes, that is computing the rotation matrix $\mathbf{R}$ that minimizes:

$$\arg\min_{\mathbf{R}} \frac{\sum_{f \in \mathcal{F}} \mathcal{A}_f \|\mathbf{R}(\mathbf{n}_f)\|_1}{\sum_{f \in \mathcal{F}} \mathcal{A}_f},$$

with $\mathbf{n}_f$ denoting face normals, and $\mathcal{A}_f$ face areas. Similarly to [Gao et al. 2015], we solve this problem with a RANSAC approach, sampling the Gauss sphere with spherical Fibonacci and selecting the orientation that performs best. Although not optimal (Section 6), this initialization significantly impacts the final results. Across all the models shown in the paper, when compared with a random initialization, obtaining up to 8 times less blocks.

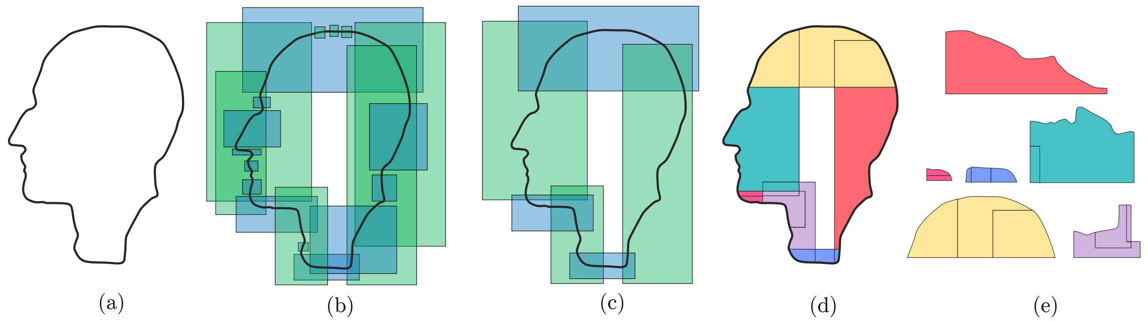


Fig. 4. Overview: (a) input; (b) dense set of axis aligned maximal height height-field blocks (height-field blocks along the horizontal direction in green, blocks along the vertical direction in blue); (c) minimal block covering of the shape; (d) final blocks after overlap removal (with boundaries of original intersections demarcated); (e) resulting blocks, laid for fabrication.

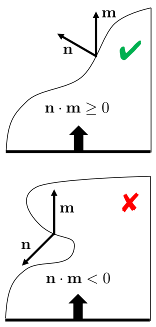
4.2 Partition into Overlapping Height-Field Blocks

The goal of this stage is to compute a set of individually valid height-field blocks that jointly cover the input surface, where each block is constrained to be an intersection of the input model and an axis aligned rectangular box. To produce a compact final decomposition we seek to minimize the number of blocks in this covering set. A greedy approach to generating such a set would be to start from a seed block, maximally grow it until it cannot be further extended without violating our constraints, and then add more blocks using a similar process until coverage is achieved. This approach is heavily dependent on the strategy used to compute seed blocks, and can result in drastically larger numbers of blocks than necessary. Instead, we use a more conservative, if more time consuming, strategy where we first compute a large set of maximal blocks that cannot be further extended without violating our block-fabrication constraints, and then select a minimal subset of them that satisfies our coverage constraint. We avoid time-consuming brute-force evaluation of fabrication constraints by precomputing valid solution spaces for block extension and constraining block computation to these spaces. This two stages process provides a suitable starting point for our overlap resolution system (Figure 4b). An advantage of this approach is that the initial maximal blocks can be computed entirely in parallel, allowing for a trivial speedup. While other strategies could be used to grow the initial maximal blocks, we found our solution to be simple to implement and robust.

4.2.1 *Maximal Height-Field Block Set.*

We desire a set of maximal size axis aligned bounding boxes that provide a good starting point for selecting a compact subset that covers the entire model.

Problem Setting. The input to this stage is a closed, intersection-free, triangle mesh $\mathcal{S} = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V}$ is the set of its vertices and $\mathcal{F}$ the set of faces. The output of this stage is a set $\mathcal{B}$ of axis aligned boxes, where the geometry of each $\mathbf{b} \in \mathcal{B}$ is encoded via the positions of its extrema corners (ones with the smallest and largest coordinate values). A box $\mathbf{b}$ is *valid* if its intersection with the input shape is a valid height-field block. We note that for closed volumes, this requirement can be recast as requiring the angle between the outward pointing normal of any input shape triangle fully or partially inside the box and the milling direction associated with the block to be acute. Evaluating this condition explicitly and repeatedly during maximal box computation can be prohibitively expensive. Below we describe an efficient way to sidestep such explicit evaluations.


m
n
✓
$\mathbf{n} \cdot \mathbf{m} \geq 0$
m
✗
n
$\mathbf{n} \cdot \mathbf{m} < 0$

Initialization. To ensure complete coverage, we initialize the set $\mathcal{B}$ with the bounding boxes of all the mesh triangles. Since each triangle can be part of at most three outward oriented height-field surfaces, we create seed bounding boxes associated with only these three orientations. In general normal directions on a surface change gradually, thus we expect each of these boxes to be valid, i.e. only overlap triangles which satisfy the acute angle constrain vis a vis our three initial axis directions. Section 4.5 discusses a pre-process which can be applied to the models to enforce this condition, if not satisfied *a priori*.

Expansion. We seek to maximize the coverage provided by each box while satisfying validity conditions. The validity testing can be reduced to two conditions. First, and most important, we need to test whether the expanded box overlaps with any triangles whose normals point in the opposite direction to the box's axis. Second, we must at all times ensure that the dimensions of the box does not exceed the dimensions of the milling machine processing volume. A naive approach to block computation would be to grow each box using small steps, e.g. adding one triangle at a time, terminating growth if and when the validity constraints are violated. However, the first test in particular can be quite time consuming, and repeatedly performing it for each box at each expansion step can be prohibitively computationally expensive for large models. Instead of testing constraints directly, we define a valid solution space for box expansion and constrain our maximization problem to this space. The solution spaces are defined independently for each of our six orientations and are reused for all blocks which share this orientation. They are formulated so as to simultaneously prevent constraint violation and enable easy continuous optimization of coverage maximization.

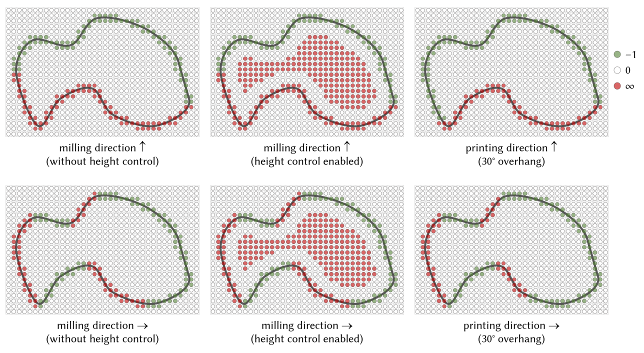


Fig. 5. Example scalar fields for maximal height-field block computation. Left column: The field is infinite next to triangles with normal opposite to the selected axis direction, negative in the areas whose coverage we seek to maximize, and zero elsewhere. Middle column: Defining maximal block height (distance from surface to block base) translates into specifying an infeasible (infinite field value) region inside the model, where the distance to the surface is above this maximal height value. Note that the infeasible inner region is computed using an isotropic distance field, hence it is computed once and used for all axis directions. Right column: by relaxing the height-field criterion (i.e., testing against 90°+ overhang instead of 90°) we can generate quasi height-field blocks with controlled overhang angles, useful for 3D printing.

Solution space. We define a set of volumetric scalar fields, one for each milling direction $\mathbf{m}$, that smoothly encode an energy function we seek to minimize for all corresponding boxes. We represent each field using a regular grid defined over a bounding cube of the input model (we set grid spacing to the average mesh edge length). The field is designed to be infinite outside the valid region with respect to the axis of interest, negative in the areas whose coverage we seek to maximize, and zero elsewhere (see Figure 5), and is specified as follows:

- ∞ on vertices of grid cells that contain triangles whose normals form obtuse angles with the milling direction;
- -1 on vertices of grid cells that contain only triangles whose normals form acute angles with the milling direction;
- 0 on all other vertices.
- optional: ∞ on grid vertices that are further from the boundary than our maximal height threshold

In case of conflicts (i.e., grid vertices having incident cells containing both obtuse and acute angles with the milling direction) we break ties by considering the lowest value we obtained. We assign continuous values within each cell by using tricubic interpolation. The energy of a box is then defined as the integral of the scalar field s inside the box:

$$\mathbf{E}(\mathbf{b}) = \iiint_{\mathbf{b}} s \, dV$$

which can be represented as the sum of the integrals over all cells of the regular grid that intersect the box. Each one of these integrals can be evaluated in closed form; we provide the derivation of this integral and of its derivatives in Appendix A. Note that any box $\mathbf{b}$ with $\mathbf{E}(\mathbf{b}) \neq \infty$ by construction satisfies our block-fabrication constants, with the exception of maximal size which is discussed in the next paragraph.

Optimization. We simultaneously grow all boxes to cover as much of the input surface as possible, while still keeping them valid. The boxes are expanded by minimizing $\mathbf{E}$, subject to additional hard constraints that limit the size of the boxes to prevent them from growing beyond our maximal size threshold, and constrain the initial seed triangle associated with each box to remain inside this box. Both sets of constraints can be expressed as linear inequalities with respect to the position and dimension of the box $\mathbf{Cb} \leq \mathbf{d}$, leading to the following non-linear optimization with linear inequality constraints:

$$\arg\min_{\mathbf{b}} \mathbf{E}(\mathbf{b}) \tag{1}$$

$$s.t. \, \mathbf{Cb} \leq \mathbf{d} \tag{2}$$

We convert this constrained optimization into an unconstrained one using logarithmic barriers, and minimize it using BFGS with bisection line search. The optimization is stopped when the energy at the current iteration is smaller than ∞ and its difference to the energy at the previous iteration is smaller than 10^{-6}.

Heuristic Pruning. The collection of maximal height blocks extracted using this basic procedure is highly redundant. To improve performance we reduce the set of processed boxes as follows. First, instead of considering all three valid directions for each seed triangle, we only use the milling direction closest to its normal. Second, we seed (and grow) boxes at random triangles, evenly distributed over the surface, and stop seeding new boxes as soon as the entire surface is completely covered, i.e. as soon as all the triangles of the surface have been assigned to at least one height block. While this heuristic could in theory lead to inferior results, they work well in practice, as we demonstrate in Figure 6 (bottom part), where we compare the results obtained with and without pruning. The difference in quality is negligible, but the heuristics reduce the computation cost from 47 to 2 minutes.

4.2.2 *Minimal Covering.*

After maximally expanding all block boxes, we compute a minimal subset of them which entirely covers the surface. Computing such a set amounts to solving the classical minimal set cover problem, known to be NP-complete [Cormen et al. 2001]. We obtain a solution by casting it as an integer linear programming problem:

$$\arg\min \mathbf{1}^T \mathbf{x} \tag{3}$$

$$s.t. \sum_i x_i \, \mathbf{a}_i \geq \mathbf{1} \tag{4}$$

$$x_i \in \{0, 1\} \tag{5}$$

where x_i is the i-th entry of $\mathbf{x}$, a vector of binary variables that indicates if the box $\mathbf{a}_i$ is kept in the minimal covering. $\mathbf{a}_i$ is a binary vector with as many entries as the faces of the input, and where a value of 1 indicates that the corresponding face is contained in the box. We use an off-the-shelf solver (http://www.gurobi.com/) to obtain a solution. While in theory the runtime for this step can be exponential, in practice the solver converges to a solution in minutes (Table 1).

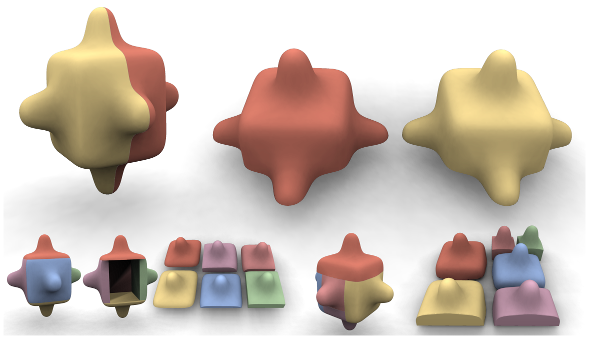


Fig. 6. Users can balance block count with the amount of material necessary for fabrication. If height is unconstrained (top) blocks are free to expand and cover the whole volume. If height is constrained (bottom-left) blocks become thinner, leaving a void in the interior and requiring less time and material to be fabricated. Specifically, the 2 height-field blocks on top require 35% more material than the 6 at the bottom-left model on bottom left to be fabricated. Bottom-Right: a decomposition with height constraints and with no heuristic pruning. The difference in quality is negligible, but the pruning reduces the computation cost from 47 to 2 minutes.

4.3 Overlap Resolution

Given the set of height-field blocks produced by the minimal covering, we seek to resolve the overlaps between them with a minimal increase in the number of blocks and without introducing thin fragile features.

Single Pair. Before addressing the general case, we consider overlap resolution on an individual pair of blocks $\mathbf{b}_1$ and $\mathbf{b}_2$ (Figure 7). The overlap between the blocks can always be eliminated by subtracting one block from another, e.g. $\mathbf{b}_2$ from $\mathbf{b}_1$. In all but some special cases, which we will discuss later, such a subtraction keeps the number of blocks constant. Post-subtraction, the block $\mathbf{b}_1 \setminus \mathbf{b}_2$ may no longer satisfy the validity constraints (Figure 7a). Depending on the configuration, reversing the order and computing $\mathbf{b}_2 \setminus \mathbf{b}_1$ may result in two valid blocks, but it is not guaranteed. In many instances, no order can produce a valid result (Figure 7c). Invalid blocks created by subtraction have multiple bases - i.e. polygons orthogonal to the milling direction. Each such block $\mathbf{b}_i \setminus \mathbf{b}_j$ can be therefore converted into a set of valid blocks by splitting it along one or more of the planes it shares with block b_j, such that each such base defines a separate block. While this solution is guaranteed to work, we want to minimize the number of such splitting operations. The basic overlap resolution method for two blocks $\mathbf{b}_1$ and $\mathbf{b}_2$ can be hence formulated as follows: (1) if only one of the two differences is valid use this difference to form a solution (Figure 7a); (2) if both $\mathbf{b}_1 \setminus \mathbf{b}_2$ and $\mathbf{b}_1 \setminus \mathbf{b}_2$ are valid blocks, perform the subtraction operation that maximizes the smallest output block (Figure 7b); (3) otherwise, split one of the difference blocks to obtain valid sub-blocks, selecting the refinement that maximizes the smallest output block (Figure 7c).

Multi-Block. We extend this framework to the multi-block scenario by casting overlap resolution as finding a sequence of subtraction operations that minimizes the number of splits required to ensure output validity. When splitting is unavoidable, we prioritize split operations that avoid producing very small blocks.

We represent the relation between adjacent height-field blocks using a directed graph whose vertices represent blocks; each graph edge represents a subtraction order dependency between its end vertices. More formally, we introduce an edge from $\mathbf{b}_1$ to $\mathbf{b}_2$ if and only if the difference $\mathbf{b}_2 \setminus \mathbf{b}_1$ is not a valid height block. Note that it is possible to have two opposite edges connecting the same pair of vertices if both subtraction orders produce invalid blocks (Figure 7c). Cycles in this graph exactly correspond to scenarios where splitting cannot be avoided. To obtain the desired subtraction order, if the initial graph contains cycles, we first transform it into a directed acyclic graph (DAG) by breaking all the cycles (and splitting the associated blocks). We then produce a valid subtraction order by computing an optimal topological order on this acyclic graph. The overall complexity of this step is $O(|s|(n + e)(c + 1))$, where $|s|$ is the number of splits, n is the number of vertices, e is the number of edges, and c is the number of cycles.

Reduction to a DAG. To break cycles in the graph we iteratively select an edge $\mathbf{b}_1 \rightarrow \mathbf{b}_2$ on each cycle and split $\mathbf{b}_2 \setminus \mathbf{b}_1$ along the height-field direction of $\mathbf{b}_2$, generating two or more non-overlapping valid blocks. We are sure that they are valid relying on a simple assumption: cutting a block with any plane containing the milling direction keeps requirements *Axis*, *Base*, and *Height-Field* in both sub-blocks. The vertices corresponding to the generated sub-blocks, denoted $\mathbf{b}_{2,1}, \ldots, \mathbf{b}_{2,n}$, are then added to the graph. Note that each sub-block's vertex can at most inherit the edges of its parent, excluding $\mathbf{b}_1 \rightarrow \mathbf{b}_2$ (since there is no intersection between $\mathbf{b}_{2,1}$ and $\mathbf{b}_{2,2}$ and neither of them intersects $\mathbf{b}_1$). As a consequence, if the block $\mathbf{b}_2$ participated in n cycles, its sub-blocks can participate at most in $n - 1$ cycles. This observation guarantees that each refinement step reduces the total number of cycles that graph vertices participate in, and consequently ensures that the reduction process terminates in a finite number of steps, producing a DAG. We detect all the cycles in the graph using Johnson's algorithm [Cormen et al. 2001]. Among all the edges participating in a loop, we give priority to the one that maximizes the size of the smallest height block created. The splitting algorithm stops when a DAG is obtained. Note that while it is guaranteed to succeed, this splitting strategy may produce sub-optimal height-field decompositions with more blocks than necessary (see Figure 8).

Topological Sorting. We produce a valid subtraction sequence by computing an optimal topological order on the resulting DAG. Producing a linear ordering of a DAG's vertices such that for every directed edge $\mathbf{b}_i \rightarrow \mathbf{b}_j$, $\mathbf{b}_i$ comes before $\mathbf{b}_j$ is a classical problem in graph theory. Note that our directed edges encode pathological splitting orders, we therefore aim to find an *inverse* topological sorting of the DAG vertices (i.e., for every directed edge $\mathbf{b}_i \rightarrow \mathbf{b}_j$, $\mathbf{b}_j$ should come before $\mathbf{b}_i$). We pre-process the DAG by inverting the orientation of each directed edge (i.e., transforming the roots in leaves, and vice versa) and performing topological sorting. Among all the possible orderings, we favor the one that maximizes the size of the smallest height-field block. To do so, we use Kahn's iterative algorithm [Kahn 1962], prioritizing the vertices associated with the smallest blocks. In short, the algorithm works as follows: at each

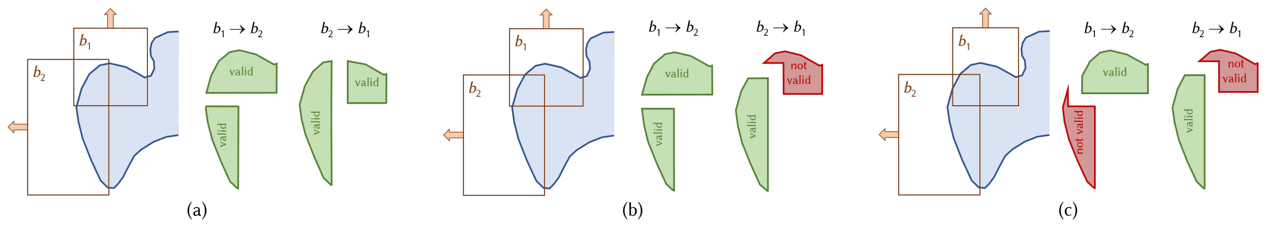


Fig. 7. Different block processing orders result in different decompositions: (a) independently on the processing order, two valid height blocks are produced; (b) processing the block b_1 before the block b_2 produces two valid height-fields, inverting the order one block is not a height-field; (c) no valid solution exists, independently on the processing order. In the latter case, splitting one of the boxes allows for a valid processing order.

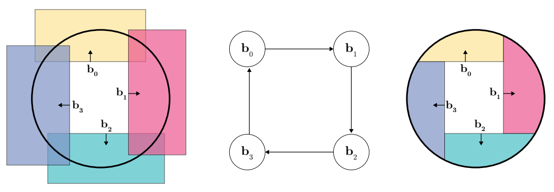


Fig. 8. Our cycle splitting strategy may produce sub-optimal height-field decompositions containing more pieces than necessary. Left: a set of axis-aligned boxes jointly covering the surface. Middle: the resulting conflict graph encoding pathological subtraction orders. A loop is present, meaning that at least one box will be split to produce a DAG (Section 4.3). Right: a valid decomposition into four height-field blocks. This solution does not correspond to any box subtraction order, and therefore cannot be produced by our algorithm.

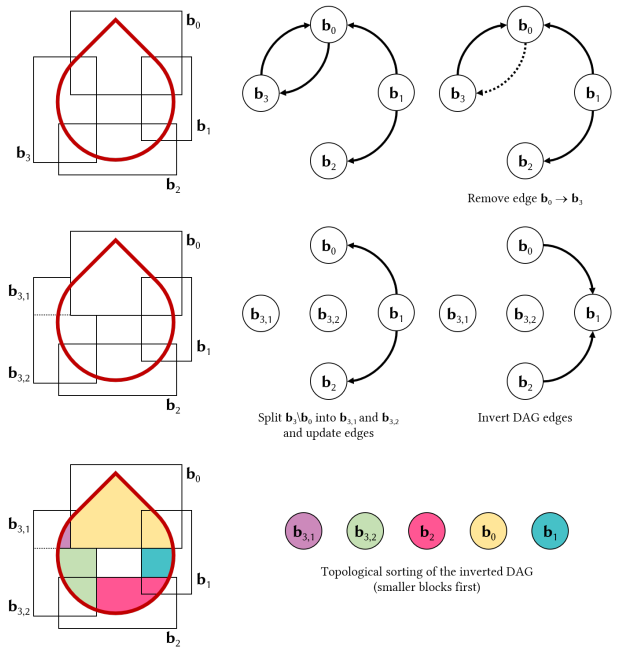

$\mathbf{b}_0$
$\mathbf{b}_1$
$\mathbf{b}_2$
$\mathbf{b}_3$
Remove edge $\mathbf{b}_0 \to \mathbf{b}_3$
$\mathbf{b}_{3,1}$
$\mathbf{b}_{3,2}$
Split $\mathbf{b}_3 \backslash \mathbf{b}_0$ into $\mathbf{b}_{3,1}$ and $\mathbf{b}_{3,2}$ and update edges
Invert DAG edges
Topological sorting of the inverted DAG (smaller blocks first)

Fig. 9. Conflicts between intersecting blocks are encoded in a directed graph. An edge $\mathbf{b}_i \to \mathbf{b}_j$ means that $\mathbf{b}_j \setminus \mathbf{b}_i$ is not a height-field. If the graph is acyclic we are guaranteed that, without splitting any box, subtracting blocks using an inverse topological order would produce a valid height-field block decomposition. If the graph contains cycles, we reduce it to a DAG by iteratively removing edges (and splitting the blocks accordingly). Among all the possible inverse topological orders, we select the one that maximizes the size of the smallest height-field block.

iteration we find the roots of the graph (vertices with no incoming edges — if the graph is a DAG at least one root always exists); we order them from the smallest to the biggest block, and use this order to perform the subtraction, removing their corresponding vertices from the DAG. We repeat the process, iteratively looking for new roots until all the vertices in the DAG have been processed (Figure 9). Note that the inverse of the resulting topological sorting can also be used to generate illustrated instructions that describe a valid assembly sequence, though we do not guarantee that a globally valid assembly sequence always exists (Section 6).

4.4 Improving Blocks Size and Shape

A shortcoming of the method described so far is that it does not explicitly prevent the generation of tiny blocks or blocks with narrow protruding features, which could potentially break during fabrication (due to the stress induced by the milling tip) or during assembly. Such features are usually generated when performing Boolean operations between blocks with close-by faces with similar orientation. We describe here a greedy twofold strategy that has no theoretical guarantees but that in our experiments successfully removes narrow features, leading to the formation of well shaped height blocks.

Block Snapping and Shrinking. We process the blocks selected by the minimal covering (Section 4.2.2), aiming to minimize the number of intersections between blocks before starting the overlap resolution (Section 4.3). First, we consider all pairs of face-adjacent blocks, that is blocks with same orientation faces for which the distance between these faces is less than a fixed amount (the default is one grid unit, the user can choose to change it). We sort all the candidate pairs according to these distances, and adjust their dimensions reducing the distance to zero, making them perfectly face-adjacent. We then consider the remaining set of intersecting blocks, and try to shrink them in order to avoid overlaps. Note that shrinking blocks may leave some portion of the surface uncovered, we therefore apply block shrinking if and only if complete surface covering is preserved. These steps result in a conflict graph with less arcs and typically less cycles, thus reducing the number of splits

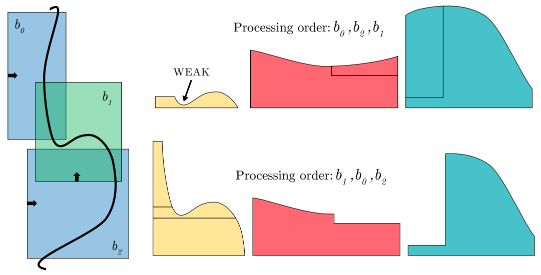

b_0
b_1
b_2
WEAK
Processing order: b_0, b_2, b_1
Processing order: b_1, b_0, b_2

Fig. 10. Left: a portion of surface is covered with three blocks: b_0, b_1, b_2. Note that any processing order would produce three valid height blocks. Right: processing b_1 after b_0, b_2 produces a narrow feature (top). We detect such configurations and locally modify the processing order: giving higher priority to b_1 produces a better height block decomposition (bottom).

necessary to reduce it to a DAG. On average, using this strategy we decreased the number of box splits by 50%.

Modified Processing Order. If, by processing the blocks in the order computed in Section 4.3, we generate a tiny height-block or a narrow feature, we rollback the operation and modify the processing order, giving the current block higher priority w.r.t. all the blocks it overlaps (Figure 10). This maximizes the size of the sub-blocks derived from such height-block and typically reduces the number of narrow features produced. We automatically detect small blocks by measuring their volume [Zhang and Chen 2001], and detect tiny features by measuring the distance between pairs of block side and base edges. More advanced approaches [Zhou et al. 2013] could be used instead but in our experiments this was not necessary since all the thin features were detected by the two criteria above.

Post Processing. The boolean processing may result in adjacent blocks with same height-field direction. To reduce block count we merge them into a single block, either by raising the base of the lower block along the height-field direction, or by lowering the base of the higher block. The condition for performing the first operation is that there are no surface triangles in between the old and the new base. The condition for performing the second operation is more complex: we need to make sure that the new block doesn't intersect any other block in the decomposition, and that it still satisfies the height-field constraint. After moving one of the two bases, we merge the two blocks. This post-processing typically merges one or two pairs of blocks.

4.5 Faithfulness vs Complexity

Enforcing strict fabrication constraints on highly detailed models often results in an excessive number of height-field blocks. For practical applications, exact fidelity to the input can often be sacrificed to reduce block count and facilitate easier fabrication. We provide an optional mechanism that allows users to reduce reconstruction accuracy, or faithfulness, in exchange for a lower block count. We note that smoother, or less detailed, models typically require significantly fewer height-field blocks to reconstruct than their detailed counterparts. We consequently achieve our target using a two step procedure that removes high-frequency surface details before the decomposition, and reintroduces the removed details into each block subject to preserving the fabrication constraints (Figure 11). To remove high-frequency details from the input shape we use the low-pass filter proposed in [Taubin 1995]. After computing the height-field block decomposition we reintroduce the high-frequency details using a variation of the Laplacian surface reconstruction framework [Sorkine 2006], enriched with height-field constraints that ensure that the vertices assigned to each block remain above its base with respect to the milling direction and that no triangle flips its orientation. Specifically, after the block decomposition of the smooth geometry is computed, every vertex $\mathbf{v}_i$ is assigned to a block $\mathbf{b}_{\mathbf{v}_i}$, which has a milling direction $\mathbf{m}_{\mathbf{v}_i}$. We then reintroduce the details by minimizing the following energy:

$$\arg\min \|\Delta\mathbf{v} - \delta\|^2 s.t. \tag{6}$$

$$\mathbf{v}_i \in \mathbf{b}_{\mathbf{v}_i} \tag{7}$$

$$\mathbf{n}_t(\mathbf{v}) \cdot \mathbf{m}_t \geq 0 \tag{8}$$

where $\delta_i = \frac{1}{|N_i|}\sum_{v_j \in N_i}(v_i - v_j)$ are the differential coordinates of the original mesh [Sorkine 2006], and $\mathbf{n}_t$ is the normal of the triangle t. Equation 7 ensures that every vertex is constrained to stay above the block's base and can be modeled with a set of linear inequality constraints. Equation 8 prevents triangle normals from flipping and is a quadratic condition on the vertex positions. We minimize this energy using coordinate descent, by optimizing one vertex at a time and freezing the others. The per-vertex optimization is solved with Newton iterations and it typically converges within 5 iterations, recovering most of the details of the input meshes (Figure 11). Specifically, the average distance between the two models, measured with Metro [Cignoni et al. 1998], is less than 1×10^{-4} times the bounding box diagonal.

Initialization Constraints. Given an input mesh, we assume that bounding boxes of individual triangles define valid height blocks. This condition can be violated in two scenarios. In the first case the top or outer surface defined by the intersection of the box and the input model may not be a height-field. This situation can only occur on meshes with high-frequency features, and it is solved using our high-frequency removal preprocess. Initial blocks can, in theory, intersect the input surface. This situation can only occur if the mesh triangle size is larger than the local feature size. Such situations can always be avoided by refining the mesh using one or more rounds of one-to-four triangle subdivision.

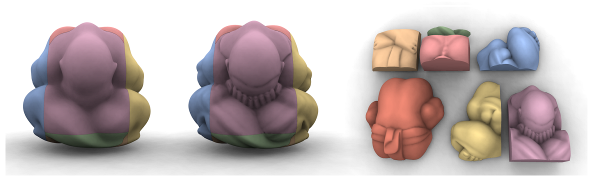


Fig. 11. To keep the number of height-field blocks low we trade faithfulness for complexity. Specifically, given a detailed model we run our method on a pre-filtered version with no high frequencies (left). We then maximally restore the details while preserving the height-field property everywhere (middle, right). This strategy allowed us to reduce the number of height-field blocks from 14 to 7.

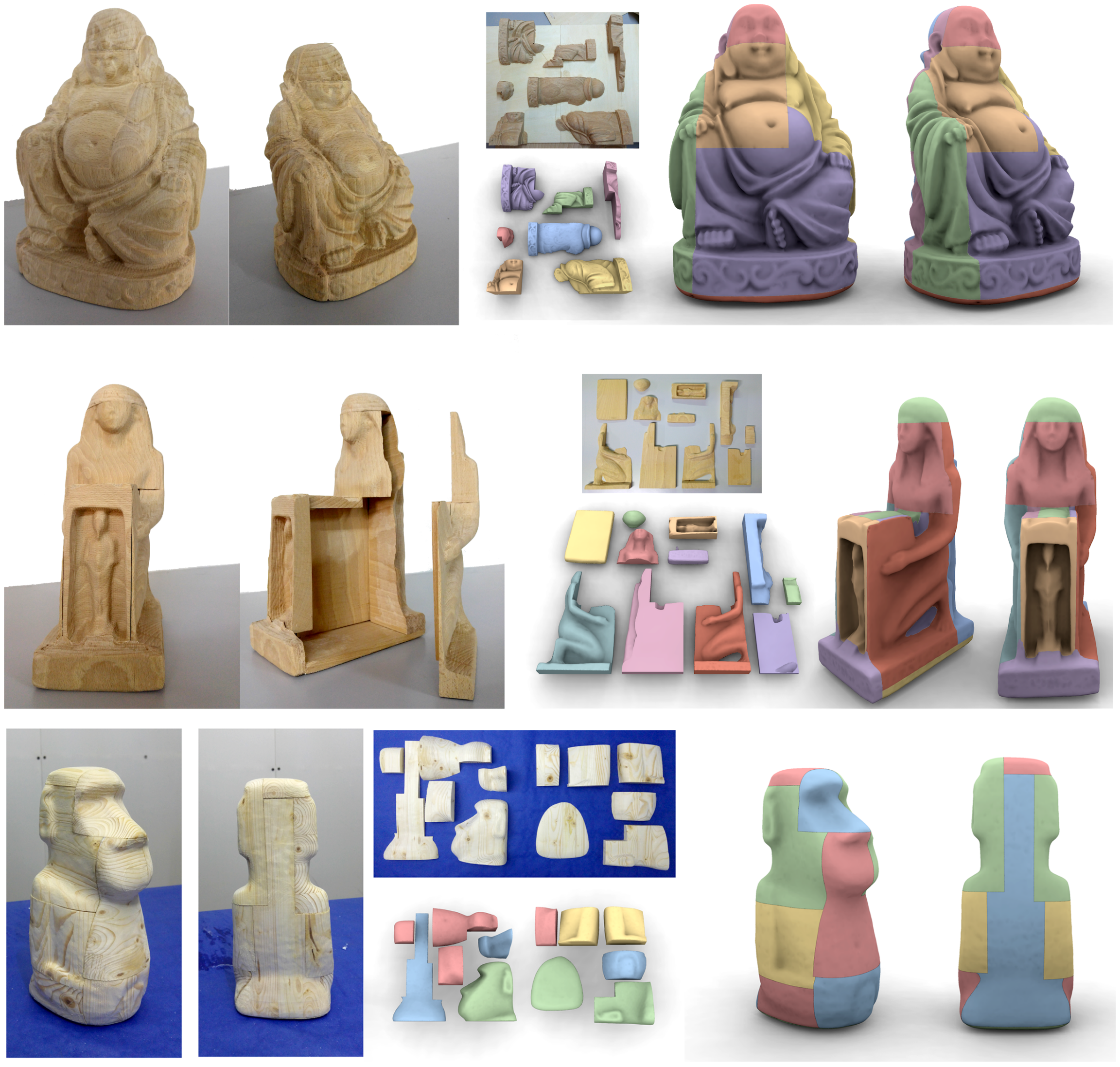


Fig. 12. A gallery of decompositions computed with our algorithm and fabricated in wood using a 3-axis milling machine.

5 RESULTS

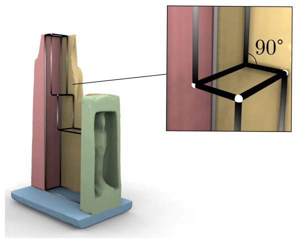
Throughout the paper we demonstrate a range of models decomposed into height-field blocks using our approach, ranging from relatively simple (Spiky cube) to highly complex (Chinese lion, Lion vase) and from relatively smooth (Kitten) to highly detailed (Buddha, Bimba). All our output decompositions are fabrication ready and strictly satisfy all validity conditions. The number of blocks in our decompositions varies from single digits (Moai, Max Planck) to 63 for Fertility.

Milled Results. We milled four objects from solid blocks of two different woods: pinewood and beechwood; the former is softer and easier to mill while producing less detailed results, the latter is harder and needs a longer milling time but leads to more detailed models. We milled *Moai* (Figure 12) and *MaxPlanck* (Figure 1) from the pinewood blocks; *Buddha* and *Egyptian Statue* from the beechwood blocks (Figure 12). *Moai* was milled with a Roland Modela MDX40, while *MaxPlanck*, *Buddha* and *Egyptian statue* have been milled with a Stepcraft-2/840 Desktop CNC System. The milling paths have been automatically generated from our height block decompositions using *Autodesk Fusion 360* (https://www.autodesk.com/products/fusion-360/overview). *Moai* is assembled from 12 blocks and is 27

centimeters tall; *MaxPlanck* from 8 blocks and is 22 centimeters tall; *Buddha* from 8 blocks and is 19 centimeters tall (we skipped the curved base since the model does not stay straight with it); the *Egyptian Statue* from 11 blocks and is 32 centimeters tall. They have been assembled using wood glue, as shown in the video sequences provided as additional material. After assembly, the seams on each model have been covered with wood putty and sanded with fine grain sandpaper. This procedure hides the seams, which are only visible in the discontinuities of the wood pattern. We tested an alternative procedure on *Moai* statuette: we covered it with water-based enamel and then polished it. The final appearance is shown in the inset.

Height Control. By controlling the maximal height of the produced blocks users can control the amount of material necessary to fabricate the object (Figure 6), trading material savings for increased block count.

Internal Framework. Restricting our milling directions to the six major axis does not only simplify the assembly of small models, where the large overlaps can be easily covered with glue or kept together using metal L-shaped connectors, but it also drastically simplifies the construction of large architectural-scale objects. In this scenario, the interior of the shape can be realized with a supporting framework made of metal beams, which follows the internal edges, and is kept together with a single type of joints: since the height blocks are axis aligned, the only possible intersections of the beams are multiples of 90 degrees. The required joints are simple to fabricate and reusable. A virtual example for the *Egyptian Statue* is in the inset.



High-Frequency Models. While our method can directly generate brute-force decompositions for high-frequency inputs, such decomposition would inevitably lead to high block counts, due to the very restrictive fabrication constraints inherent in 3-axis CNC milling. Our optional high-frequency filtering algorithm (Section 4.5) leads to much simpler decompositions, while losing minimal surface details, as shown in Figure 11.

Comparison with [Hu et al. 2014]. As noted earlier, in contrast to our framework the method of Hu et. al [2014] has no direct control on how far its outputs deviate from the height-field or pyramidality constraints, and demonstrate results where these constraints are far from satisfied. To compare the two approaches, we repeat the experiment proposed in Figure 23 of [Hu et al. 2014] and show the results in Figure 15. Our method introduces less blocks to decompose the model, and, more importantly, our blocks are millable, while the

Model	Timing				# Blocks	Height
	MHFBC	MC	OR	B		
Airplane	1′41″	1″	1″	9″	11	∞
Batman	33′44″	20″	1″	54″	8	0.15
Bimba	25′23″	3″	0.2″	39″	16	∞
BU (orientation)	17′40″	9″	2″	27′24″	16	0
BU (no orientation)	71′3″	9″	1″	23″	9	0
Buddha	61′37″	15″	1″	33″	8 (7 milled)	0.125
Chinese Lion	7′38″	3″	2″	59″	27	0.125
	2′35″	14″	0.1″	6″	2	∞
Cube Spike	1′19″	4″	0.1″	9″	6	0.4
	47′21″†	6′48″	0.1″	13″	6	0.4
David	26′27″	9″	0.2″	18″	7	0.075
Dea	31′33″	20″	0.2″	21″	7	0.075
Egea	15′56″	6″	0.1″	7″	6	0.1
Egyptian Statue	16′7″	6″	5″	21″	11	0.05
Eros	39′33″	10″	1″	1′1″	10	0.125
Fertility	15′32″	11″	32″	10′20″	63	∞
Gentildonna	36′23″	9″	0.1″	20″	10	0.125
Kitten	21′25″	6″	2″	59″	25	0.05
Lincoln	8′23″	6″	0.6″	1′41″	14	0.125
Lion Vase	29′24″	8″	0.4″	14″	10	0.175
Max Plank	15′59″	10″	0.3″	14″	8	∞
Moai	12′47″	4″	0.2″	9″	12	0.225
Pensatore	11′4″	9″	0.1″	29″	7	0.1

† No pruning

Table 1. Model statistics: computation time (split into Maximal Height-Field Block Computation (MHFBC), Minimal Covering (MC), Overlap Resolution (OR), CSG Operations (B)); number of blocks; and the block height maximum used (as percentage of model diagonal, ∞ means no height limit).

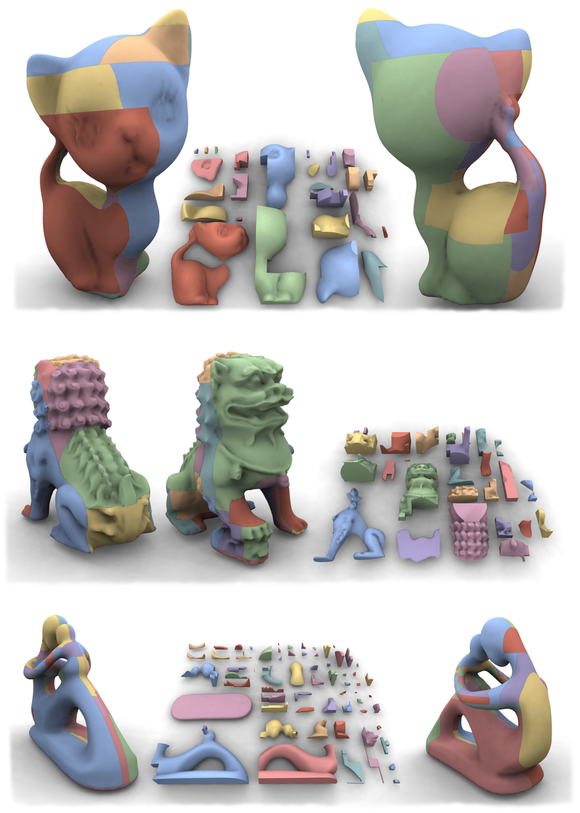


Fig. 13. Our method produces valid, compact decompositions for complex models containing: non-trivial topology (e.g., Fertility), thin features (e.g., the ears of the Kitten) and large portions not aligned with the global frame (e.g., the Chinese Lion's body) for which manual decomposition is highly challenging to compute, resulting in 27 blocks for the Chinese Lion, 25 for Kitten and 63 for Fertility.

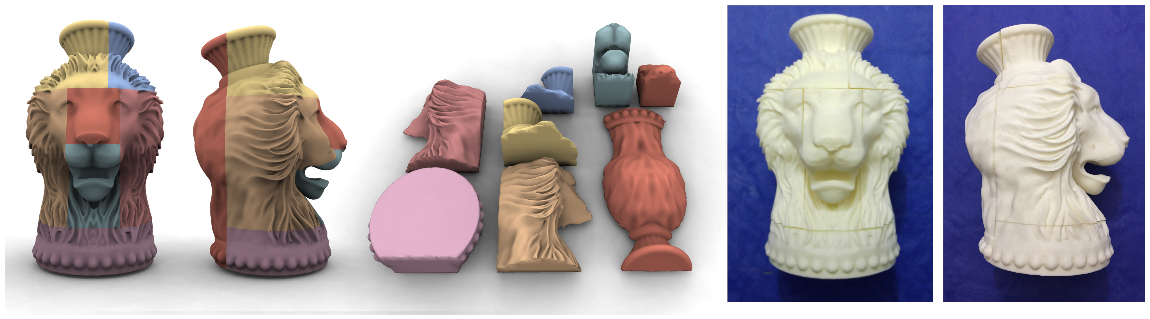


Fig. 14. By relaxing the height-field constraint we can decompose an object into blocks whose top surface contains overhangs that are strictly smaller than a printer-specific threshold. Here we show an example of a decomposition with a maximal overhang of 30 degrees for the Lion Vase dataset (left, middle). Notice from the side view that the lion mouth contains a considerable amount of overhangs. Since the threshold we set is compatible with that of common FDM printers, we safely printed each block without using support structures, saving time and material, and achieving better surface quality (right).

outputs of Hu et al. do not satisfy the height-field property (highlighted with red ovals in Figure 15) and thus can only be fabricated with a 3D printer.

3D Printing. Our pipeline is designed to produce height-field blocks, but, with a minor modification, becomes a powerful tool to produce decompositions tailored for FDM 3D printers. These printers require support material to sustain overhanging parts with angles larger than 35-40 degrees. The support does not only increases printing time and wastes material [Vanek et al. 2014], but also lowers surface quality [Zhang et al. 2015]. Our method can be used to decompose an object into blocks whose top surface contains overhang that are strictly smaller than a printer-specific threshold by simply relaxing the height-field condition in Section 4.2.1, assigning an infinite value to the scalar field only when triangles have a larger overhang. We show an example of a decomposition with a maximal overhang of 30 degrees in Figure 14 (note that this is the only result with overhangs, all the other results are decomposed in height-field blocks).

Implementation Details. We implemented our algorithm in C++, using *Eigen* [Guennebaud et al. 2010] for linear algebra routines, *Gurobi* (http://www.gurobi.com) for branch and bound, and *libigl* for mesh booleans [Zhou et al. 2016; Jacobson et al. 2016]. We run all our experiments on a workstation with a 4-cores Intel i7-4790K processor clocked at 4.0 Ghz and 16 Gb of memory. Our method takes under one hour on even the most complex model (*Buddha*) with the runtime dominated by the initial maximal size block computation. A summary of the timings and number of height-field blocks for all our experiments is shown in Table 1 and we attach all our results (input/output) in obj format as additional material.

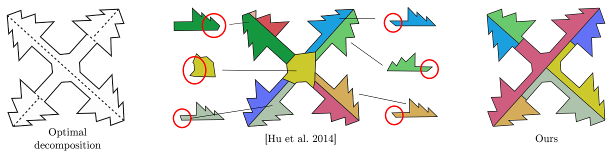


Fig. 15. Comparison with [Hu et al. 2014]: pyramidal decomposition (middle) produces ten blocks, six of which violate the height-field condition (see red ovals). Our method (right) decomposes the shape into seven valid height-field blocks, one more than the optimal decomposition (left). Note that we re-oriented the model before running our method, according to the strategy described in Section 4. For the sake of better visual comparison all the models are shown in the same position.

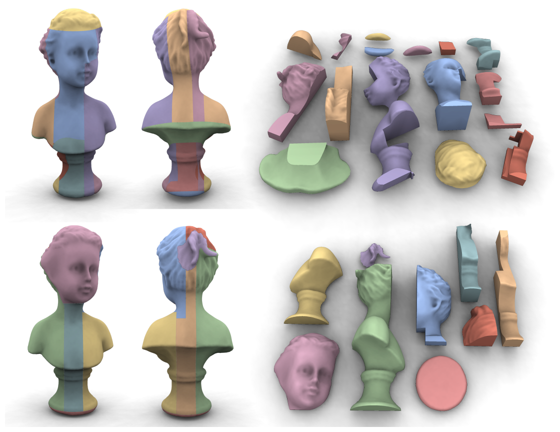


Fig. 16. Our canonical orientation algorithm is not guaranteed to produce decompositions with minimal number of height blocks. A failure example is illustrated here: with automatic orientation the BU statue is decomposed in 16 blocks (top); with manual orientation the number of blocks goes down to 9 (bottom). Finding the orientation that minimizes the number of blocks in the decomposition is a challenging problem that we plan to tackle in future work.

6 CONCLUSIONS AND DISCUSSION

We presented an automatic and robust pipeline to decompose a triangle mesh into a collection of non-overlapping, valid height-field blocks, which can be directly manufactured using a 3-axis CNC milling machine. Our method is compatible with existing milling systems , and can be used to produce high quality real-life replicas of complex virtual geometries in a range of sizes. Our pipeline is automatic and robust: the only hard requirement on the input is that it should be a closed surface. The number of blocks

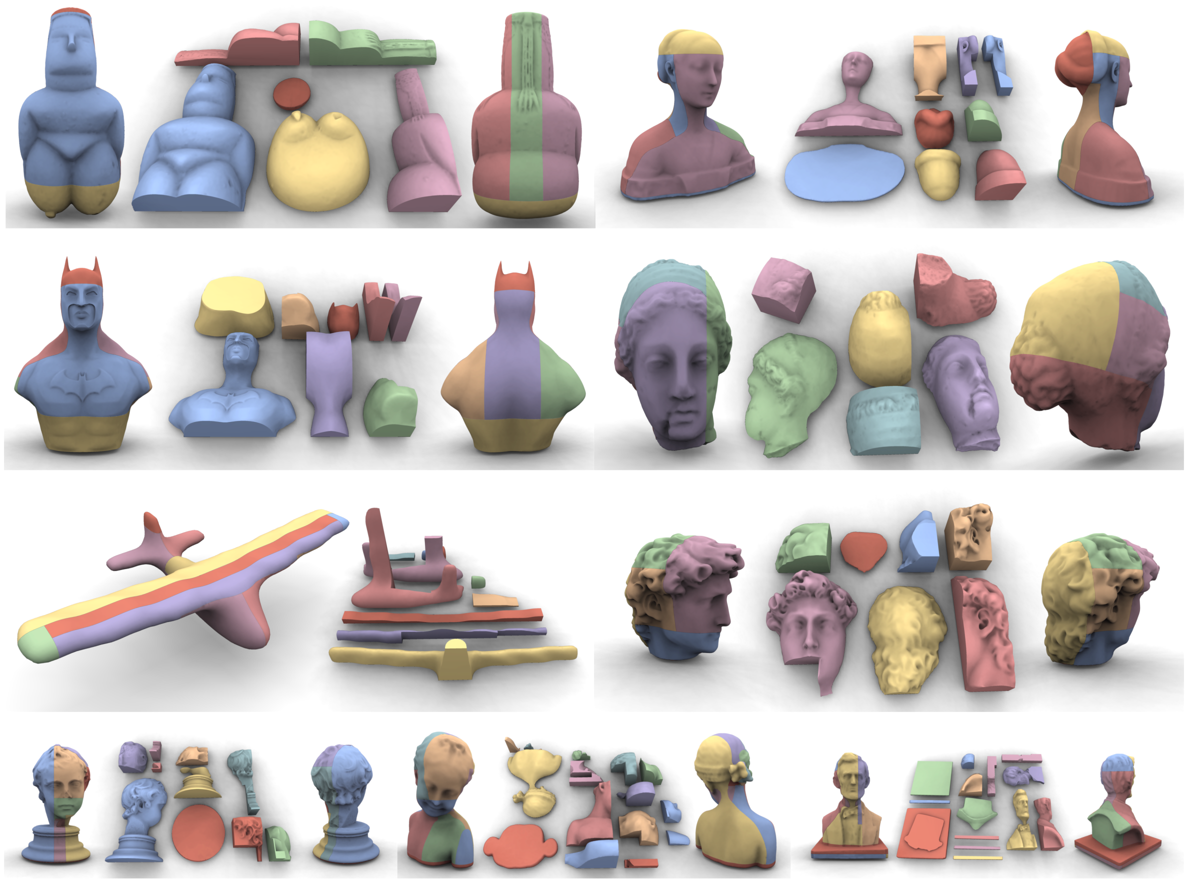


Fig. 17. Additional results generated using our method.

we produce is dependent on the input model complexity, and can significantly increase for models with narrow prominent features or high genus, resulting in objects that may be hard to assemble. Manual decomposition and independent processing of the different parts can help reduce the block count. Our output depends on the orientation of the model, which might lead to decompositions with more pieces than necessary. While our orientation choice works well for many models, manual orientation can sometimes reduce block count (Figure 16). While we explicitly seek to avoid blocks with small features and successfully avoid cases that lead to fabrication failures, we are not guaranteed to produce a decomposition with the smallest number of such blocks. We do not consider optimal cut placement, such as avoiding seams on salient features, as in [Herholz et al. 2015]. Preventing seam placement in certain regions could possibly be incorporated into our minimal covering step: boxes that only partially cover such regions can be discarded or have lower priority. Finally, while in our practical experience we had never encountered these problems, we provide no guarantees that internal voids introduced by our system do not affect balance or structural strength. An extensive FEM analysis should be performed, and possibly directly integrated in the form-finding.

Assemblability. Being a height-field, each block can be locally extracted with a linear motion parallel to its build direction [Attene 2015]. However, the extent of such motion may be limited by collisions with nearby blocks, thus preventing assemblability. We tested the existence of a valid assembly sequence only for the fabricated models in Figure 1 and 12. We do not guarantee the existence of a valid one for the other decompositions shown in the paper. In particular, for complex models such as the ones shown in Figure 13 a valid assembly sequence may not exist.

Height-Field Blocks and Milling. There are multiple milling machine configurations. Our algorithm produces height-field blocks that are *face millable* (i.e., no undercuts are allowed). Our height-field constraints can be relaxed for *shoulder millable* settings, where the drill bit can remove material sideways, generating some undercuts (for example to chamfer a hole) [Smid 2003]. The advantage of using the face milling configurations is the availability of off-the shelf milling-path computation software that can perform the path computation automatically. We do not address all milling constraints, e.g. we do not account for drill-bit thickness, as these problems are complementary to our decomposition focus.

ACKNOWLEDGMENTS

This work was supported in part by the NSF CAREER award IIS-1652515, the Italian DSURF PRIN 2015 (2015B8TRFM) project, the European Union's Horizon 2020 research and innovation programme under grant agreement No 680448 (CAxMan), a gift from Adobe, and a gift from NTopology.

REFERENCES

Giuseppe Alemanno, Paolo Cignoni, Nico Pietroni, Federico Ponchio, and Roberto Scopigno. 2014. Interlocking pieces for printing tangible cultural heritage replicas. In *Eurographics Workshop on Graphics and Cultural Heritage*, Reinhard Klein and Pedro Santos (Eds.). Eurographics Association, 145–154.

Marco Attene. 2015. Shapes In a Box: Disassembling 3D Objects for Efficient Packing and Fabrication. *Computer Graphics Forum* 34, 8 (2015), 64–76.

Marco Attene, Michela Mortara, Michela Spagnuolo, and Bianca Falcidieno. 2008. Hierarchical convex approximation of 3D shapes for fast region selection. *Computer graphics forum* 27, 5 (2008), 1323–1332.

Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. 2014. Spin-it: optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 96.

Bernard Chazelle. 1984. Convex Partitions of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm. *SIAM J. Comput.* 13, 3 (1984), 488–507.

Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. 2014. Field-aligned Mesh Joinery. *ACM Trans. Graph.* 33, 1, Article 11 (Feb. 2014), 12 pages.

Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. 1998. Metro: measuring error on simplified surfaces. *Computer Graphics Forum* 17, 2 (1998), 167–174.

Robert L Cook. 1984. Shade trees. *ACM Siggraph Computer Graphics* 18, 3 (1984), 223–231.

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. *Introduction to Algorithms* (2nd ed.). McGraw-Hill Higher Education.

Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. 2014. Assembling Self-supporting Structures. *ACM Trans. Graph.* 33, 6, Article 214 (Nov. 2014), 10 pages.

Michael Doggett and Johannes Hirche. 2000. Adaptive view dependent tessellation of displacement maps. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*. ACM, 59–66.

Sándor P Fekete and Joseph SB Mitchell. 2001. Terrain decomposition and layered manufacturing. *International Journal of Computational Geometry & Applications* 11, 06 (2001), 647–668.

Wei Gao, Yunbo Zhang, Diogo C Nazzetta, Karthik Ramani, and Raymond J Cipra. 2015. RevoMaker: Enabling multi-directional and functionally-embedded 3D printing using a rotational cuboidal platform. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 437–446.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org. (2010).

Jingbin Hao, Liang Fang, and Robert E Williams. 2011. An efficient curvature-based partitioning of large-scale STL models. *Rapid Prototyping Journal* 17, 2 (2011), 116–127.

Philipp Herholz, Wojciech Matusik, and Marc Alexa. 2015. Approximating Free-form Geometry with Height Fields for Manufacturing. *Computer Graphics Forum* 34, 2 (2015), 239–251.

Kristian Hildebrand, Bernd Bickel, and Marc Alexa. 2012. Crdbrd: Shape Fabrication by Sliding Planar Slices. *Comput. Graph. Forum* 31, 2pt3 (May 2012), 583–592.

Kristian Hildebrand, Bernd Bickel, and Marc Alexa. 2013. Orthogonal slicing for additive manufacturing. *Computers & Graphics* 37, 6 (2013), 669–675.

Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. 2014. Approximate Pyramidal Shape Decomposition. *ACM Trans. Graph.* 33, 6, Article 213 (2014), 12 pages.

Alec Jacobson, Daniele Panozzo, et al. 2016. libigl: A simple C++ geometry processing library. (2016). http://libigl.github.io/libigl/.

Arthur B Kahn. 1962. Topological sorting of large networks. *Commun. ACM* 5, 11 (1962), 558–562.

V Kraevoy, D Julius, and A Sheffer. 2007. Shuffler: Modeling with interchangeable parts. *The Visual Computer* (2007).

Jyh-Ming Lien and Nancy M Amato. 2007. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*. ACM, 121–131.

Marco Livesu, Stefano Ellero, Jonàs Martínez, Sylvain Lefebvre, and Marco Attene. 2017. From 3D models to 3D prints: an overview of the processing pipeline. *Computer Graphics Forum* 36, 2 (2017), 537–564.

Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. 2009. 3D Polyomino Puzzle. In *ACM SIGGRAPH Asia 2009 Papers (SIGGRAPH Asia '09)*. ACM, New York, NY, USA, Article 157, 8 pages.

Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. 2012. Chopper: Partitioning Models into 3D-printable Parts. *ACM Trans. Graph.* 31, 6, Article 129 (2012), 9 pages.

Asla Medeiros e Sá, Karina Rodriguez Echavarria, Nico Pietroni, and Paolo Cignoni. 2016. State Of The Art on Functional Fabrication. In *Eurographics Workshop on Graphics for Digital Fabrication (2016)*. Eurographics Associaton. http://vcg.isti.cnr.it/Publications/2016/MRPC16

H Medellín, T Lim, J Corney, JM Ritchie, and JBC Davies. 2007. Automatic subdivision and refinement of large components for rapid prototyping production. *Journal of Computing and Information Science in Engineering* 7, 3 (2007), 249–258.

Przemyslaw Musialski, Thomas Auzinger, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2015. Reduced-order shape optimization using offset surfaces. *ACM Transactions on Graphics* 34, 4 (2015), 102.

Przemyslaw Musialski, Christian Hafner, Florian Rist, Michael Birsak, Michael Wimmer, and Leif Kobbelt. 2016. Non-Linear Shape Optimization Using Local Subspace Projections. *ACM Transactions on Graphics* 35, 4 (2016), 87:1–87:13.

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Transactions on Graphics* 32, 4 (2013), 81:1–81:10.

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Computer Graphics Forum* 32, 5 (2013), 125–135.

Yuliy Schwartzburg and Mark Pauly. 2013. Fabrication-aware Design with Intersecting Planar Pieces. *Computer Graphics Forum (Proceedings of Eurographics 2013)* 32, 2 (2013), 317–326.

Ariel Shamir. 2008. A survey on mesh segmentation techniques. *Computer graphics forum* 27, 6 (2008), 1539–1556.

Mélina Skouras, Stelian Coros, Eitan Grinspun, and Bernhard Thomaszewski. 2015. Interactive Surface Design with Interlocking Elements. *ACM Trans. Graph.* 34, 6, Article 224 (Oct. 2015), 7 pages.

Peter Smid. 2003. *CNC programming handbook: a comprehensive guide to practical CNC programming*. Industrial Press Inc.

Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. 2016. CofiFab: Coarse-to-Fine Fabrication of Large 3D Objects. *ACM Transactions on Graphics (SIGGRAPH 2016)* 35, 4 (2016). Article 45.

Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. 2012. Recursive Interlocking Puzzles. *ACM Trans. Graph.* 31, 6, Article 128 (Nov. 2012), 10 pages.

Peng Song, Zhongqi Fu, Ligang Liu, and Chi-Wing Fu. 2015. Printing 3D objects with interlocking parts. *Computer Aided Geometric Design* 35 (2015), 137–148.

Olga Sorkine. 2006. Differential representations for mesh processing. *Computer Graphics Forum* 25, 4 (2006), 789–807.

Gabriel Taubin. 1995. Curve and surface smoothing without shrinkage. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*. IEEE, 852–857.

S. B. Tor and A. E. Middleditch. 1984. Convex Decomposition of Simple Polygons. *ACM Trans. Graph.* 3, 4 (Oct. 1984), 244–265.

Juraj Vanek, JA Galicia, Bedrich Benes, R Mech, N Carr, Ondrej Stava, and GS Miller. 2014. PackMerger: A 3D print volume optimizer. *Computer Graphics Forum* 33, 6 (2014), 322–332.

Lingfeng Wang and Emily Whiting. 2016. Buoyancy Optimization for Computational Fabrication. *Computer Graphics Forum* 35, 2 (2016), 49–58.

WM Wang, C Zanni, and L Kobbelt. 2016. Improved Surface Quality in 3D Printing by Optimizing the Printing Direction. *Computer Graphics Forum* 35, 2 (2016), 59–70.

Rundong Wu, Huaishu Peng, François Guimbretière, and Steve Marschner. 2016. Printing Arbitrary Meshes with a 5DOF Wireframe Printer. *ACM Trans. Graph.* 35, 4, Article 101 (July 2016), 101:1–101:9 pages.

Shiqing Xin, Chi-Fu Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. 2011. Making Burr Puzzles from 3D Models. In *ACM SIGGRAPH 2011 Papers (SIGGRAPH '11)*. ACM, New York, NY, USA, Article 97, 8 pages.

Miaojun Yao, Zhili Chen, Linjie Luo, Rui Wang, and Huamin Wang. 2015. Level-set-based Partitioning and Packing Optimization of a Printable Model. *ACM Trans. Graph.* 34, 6, Article 214 (2015), 11 pages.

Cha Zhang and Tsuhan Chen. 2001. Efficient feature extraction for 2D/3D objects in mesh representation. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, Vol. 3. IEEE, 935–938.

Xiaoting Zhang, Xinyi Le, Athina Panotopoulou, Emily Whiting, and Charlie C. L. Wang. 2015. Perceptual Models of Preference in 3D Printing Direction. *ACM Trans. Graph.* 34, 6, Article 215 (Oct. 2015), 12 pages.

Yunbo Zhang, Wei Gao, Luis Paredes, and Karthik Ramani. 2016. CardBoardiZer: Creatively Customize, Articulate and Fold 3D Mesh Models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 897–907.

Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh Arrangements for Solid Geometry. *ACM Transactions on Graphics (TOG)* 35, 4 (2016).

Qingnan Zhou, Julian Panetta, and Denis Zorin. 2013. Worst-case Structural Analysis. *ACM Trans. Graph.* 32, 4, Article 137 (July 2013), 12 pages.

A BOX-INTEGRATION OF A TRICUBIC SCALAR FIELD.

Let B be a an axis aligned box defined by its two extreme points $p = (p_x, p_y, p_z)$ and $q = (q_x, q_y, q_z)$ and L a regular lattice. We define S as the set of all the cubes $(s_{u,v,w})$ in the lattice L partially or completely contained in B:

$$S : \{s_{u,v,w} \in L | s_{u,v,w} \cap B \neq \emptyset\}$$

and we define D as the set of triplets of indexes in S:

$$D : \{(u,v,w) | s_{u,v,w} \in S\}$$

Each cube $s_{u,v,w}$ in L is defined by its two extreme points (x_m, y_n, z_p) and $(x_{m+1}, y_{n+1}, z_{p+1})$ and to each cube we associate a set of coefficients $a^{(u,v,w)}_{i,j,k}$ for performing the tricubic interpolation inside $s_{u,v,w}$. The interpolated value f in a generic (x,y,z) point inside $s_{u,v,w}$ is:

$$f(x,y,z) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\left(a^{(u,v,w)}_{i,j,k} x^i y^j z^k\right)$$

The energy to minimize is given by the integral over B, hence the sum of all the integrals over the cubes inside B:

$$\min \sum_{u,v,w \in D} \int_{x_{\min}, y_{\min}, z_{\min}}^{x_{\max}, y_{\max}, z_{\max}} f(x,y,z)$$

where:

$$x_{\min} = \begin{cases} x_m & \text{if } x_m > p_x \\ p_x & \text{otherwise} \end{cases}$$

$$x_{\max} = \begin{cases} x_{m+1} & \text{if } x_{m+1} < q_x \\ q_x & \text{otherwise} \end{cases}$$

and similarly for y and z.

The parameters are the coordinates of the points defining B. To be sure that the starting box will always cover the first primitive, we sum to the energy a barrier function tending to $+\infty$ when one of the coordinates of B is too near to one of the coordinates of the *points to cover* (i.e, the endpoints of a segment), and is 0 when the coordinates are enough far from these points as in [Schüller et al. 2013].

Suppose to have a set C of points $c = (c_x, c_y, c_z)$ to be covered by our box B. For each $c \in C$ we will have a function for p and for q:

$$\Phi_{c,t}(p_x) = \begin{cases} +\infty & \text{if } p_x \geq c_x \\ \frac{1}{g(p_x)} & \text{if } c_x - t < p_x < c_x \\ 0 & \text{if } p_x \leq c_x - t \end{cases}$$

$$\Phi_{c,t}(q_x) = \begin{cases} 0 & \text{if } q_x \geq c_x + t \\ \frac{1}{g(q_x)} & \text{if } c_x < q_x < c_x + t \\ +\infty & \text{if } q_x \leq c_x \end{cases}$$

and similarly for y and z, where t is $\frac{1}{10}$ of the lattice's edge, and g is:

$$g(x) = \frac{1}{t^3}x^3 - \frac{3}{t^2}x^2 + \frac{3}{t}x$$

Defining

$$\Phi_{c,t}(p) = \Phi_{c,t}(p_x) + \Phi_{c,t}(p_y) + \Phi_{c,t}(p_z),$$

$$\Phi_{c,t}(q) = \Phi_{c,t}(q_x) + \Phi_{c,t}(q_y) + \Phi_{c,t}(q_z)$$

we can add the barriers to the energy function to minimize:

$$\min \sum_{u,v,w \in D} \int_{x_{\min}, y_{\min}, z_{\min}}^{x_{\max}, y_{\max}, z_{\max}} f(x,y,z) + \sum_{c \in C} \left(\Phi_{c,t}(p) + \Phi_{c,t}(q)\right).$$

B VALID HEIGHT-BLOCK DECOMPOSITION VIA AA BOX SPLITTING.

Let B_1 and B_2 be a pair of intersecting axis aligned height boxes associate to the milling directions m_1 and m_2, which are in the set $(\pm X, \pm Y, \pm Z)$. Since B_1 and B_2 are axis aligned, their intersection $BI = B_1 \cap B_2$ is an axis aligned box. The planes on which the six facets of BI lie, partition both B_1 and B_2 into eight sub-boxes each, namely $B_{1.1}, \ldots, B_{1.8}$ and $B_{2.1}, \ldots, B_{2.8}$. Note that, since B_1 and B_2 intersect, there always exist two indices $i, j \in [1, 8]$ such that $B_{1.i} \equiv B_{2.j} \equiv BI$.

Let us now consider which milling directions, between m_1 and m_2, could be chosen for these sub-boxes. We can observe that: (i) B_1, B_2 and all the sub-boxes are axis aligned, therefore the angle between the box facets and the milling direction is either 0° or 90°; (ii) each sub-box of B_1 $(B_{1.1}, \ldots, B_{1.8})$ has m_1 as candidate milling direction, and each sub-box of B_2 $(B_{2.1}, \ldots, B_{2.8})$ has m_2 as candidate milling direction. The only exception is BI, which has both m_1 and m_2 as candidate milling directions.

From (i) and (ii) descends that any sub-box has *at least* one valid milling direction with an axis aligned facet as supporting base. In other words a valid height block decomposition obtained by splitting the original boxes using axis aligned planes always exists. □

Note that this is true only for the special case of axis aligned boxes and milling directions; in any other case condition (i) would not be satisfied, as the angle between the box facets and the milling direction may be greater than 90°, thus violating the height-field condition.