DHFSlicer: Double Height-Field Slicing for Milling Fixed-Height Materials

JINFAN YANG, University of British Columbia
CHRYSTIANO ARAUJO, University of British Columbia
NICHOLAS VINING, University of British Columbia
ZACHARY FERGUSON, New York University
ENRIQUE ROSALES, University of British Columbia and Universidad Panamericana
DANIELE PANOZZO, New York University
SYLVAIN LEFEVBRE, INRIA
PAOLO CIGNONI, CNR ISTI
ALLA SHEFFER, University of British Columbia

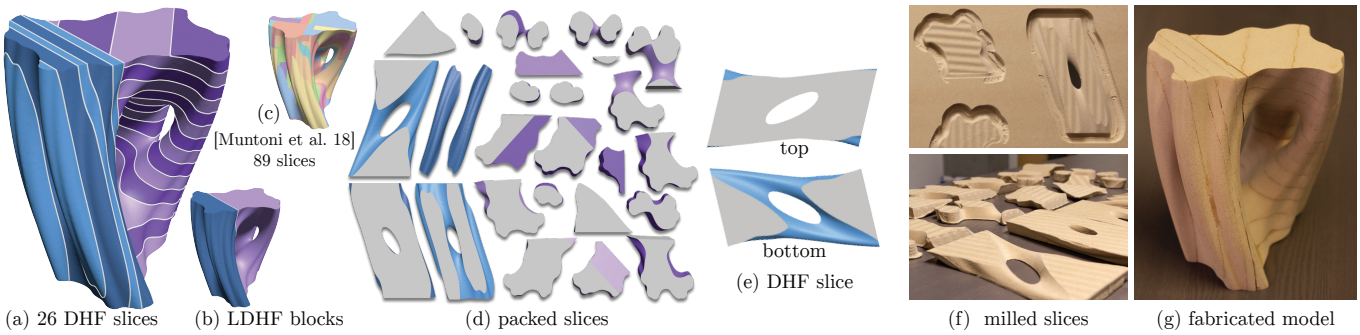


Fig. 1. *DHFSlicer* partitions complex input shapes into bounded-height double-height-field (DHF) slices (a,e) that once packed (d) and augmented with automatically computed fixtures and registration patterns (f) can be milled out of fixed height slabs of material, and assembled to produce accurate replicas of the input (g). It first partitions the inputs into coarse blocks that satisfy a *local* DHF (LDHF) criterion with respect to their respective axes (b), and then cuts those into well-sized bounded-height slices (e). Using our DHF slices halves the milling time and reduces material waste by over 40% compared to using slices produced by a state-of-the-art alternative [Muntoni et al. 2018] (c). We render slices belonging to the same block using alternating same hue colors.

3-axis milling enables cheap and precise fabrication of target objects from precut slabs of materials such as wood or stone. However, the space of directly millable shapes is limited since a 3-axis mill can only carve a height-field (HF) surface during each milling and their size is bounded by the slab dimensions, one of which, the *height*, is typically significantly smaller than the other two for many typical materials. Extending 3-axis milling of precut slabs to general arbitrarily-sized shapes requires decomposing them into bounded-height 3-axis millable parts, or *slices*, which can be individually milled and then assembled to form the target object. We present *DHFSlicer*, a novel decomposition method that satisfies the above constraints and significantly reduces both milling time and material waste compared to alternative approaches. We satisfy the fabrication constraints by partitioning target objects into *double height-field* (DHF) slices, which can be fabricated using two milling passes: the HF surface accessible from one side is milled first, the slice is then flipped using appropriate fixtures, and then the second, remaining, HF surface is milled. *DHFSlicer* uses an efficient coarse-to-fine decomposition process: It first partitions the inputs into maximally coarse blocks that satisfy a *local* DHF criterion with respect to per-block milling axes, and then cuts each block into well-sized DHF slices. It minimizes milling time and material waste by keeping the slice count small, and maximizing slice height. We validate our method by embedding it within an end-to-end DHF milling pipeline and fabricating objects from slabs of foam, wood, and MDF; demonstrate that using the obtained slices reduces milling time and material waste by 42% on average compared to existing automatic alternatives; and highlight the benefits of DHFSlicer via extensive ablation studies.

Authors' addresses: Jinfan Yang, University of British Columbia, yangjf@cs.ubc.ca; Chrystiano Araujo, University of British Columbia, yangjf@cs.ubc.ca; Nicholas Vining, University of British Columbia, nvining@cs.ubc.ca; Zachary Ferguson, New York University, zfergus@nyu.edu; Enrique Rosales, University of British Columbia, albertr@cs.ubc.ca, Universidad Panamericana; Daniele Panozzo, New York University, panozzo@nyu.edu; Sylvain Lefevbre, INRIA, sylvain.lefebvre@inria.fr; Paolo Cignoni, CNR ISTI, paolo.cignoni@isti.cnr.it; Alla Sheffer, University of British Columbia, sheffa@cs.ubc.ca.

0730-0301/2020/12-ART1 $15.00
https://doi.org/10.1145/3414685.3417810

ACM Reference Format:
Jinfan Yang, Chrystiano Araujo, Nicholas Vining, Zachary Ferguson, Enrique Rosales, Daniele Panozzo, Sylvain Lefevbre, Paolo Cignoni, and Alla Sheffer. 2020. DHFSlicer: Double Height-Field Slicing for Milling Fixed-Height Materials. *ACM Trans. Graph.* 39, 6, Article 1 (December 2020), 17 pages. https://doi.org/10.1145/3414685.3417810

1 INTRODUCTION

3-axis CNC milling, in which a computer-controlled rotary cutter is constrained to travel along the principal axes without a change in orientation and gradually removes material from a workpiece, is a widely available and robust fabrication method. This technology allows users to cheaply and precisely fabricate objects from commercially available precut fixed-size slabs of material, such as wood or stone, at a large range of scales; however, 3-axis CNC milling imposes strict constraints on the fabricated geometries. Since each milling pass can only carve a single height-field (HF) surface accessible along the milling direction, *millable* geometries are limited to a union of HF surfaces that can be appropriately oriented using suitable *fixtures*. Fabricating generic geometric objects using this approach requires decomposing them into *millable* and *assemblable* parts, or *slices*, whose sizes are bounded by the dimensions of the material slabs (Figure 1). We propose *DHFSlicer*, a new fully automatic technique for decomposing general complex 3D objects into such bounded-size, millable and assemblable slices (Figure 1a).

While most prior methods satisfy millability constraints by using approximately cylindrical or single height-field slices, we satisfy millability by constraining each slice to be a *double-height-field (DHF)* surface. We characterize a DHF surface as a union of two height-field surfaces, defined with respect to the two opposite directions of the *same* axis. A DHF slice can be accurately milled using a standard 3-axis CNC milling machine by first milling it along one of the axis directions to generate the first HF surface, then flipping the slice over using appropriate fixtures, and milling it from the opposite direction to generate the second HF surface (Figure 2, top). From a fabrication perspective, DHF slices are almost as easy to fabricate as HF slices, but encompass a much larger class of geometries. We use the additional degrees of freedom enabled by DHF surfaces as a basis for a novel decomposition approach that outperforms both automatic and manual alternatives (Section 2, Figure 5). DHFSlicer produces significantly fewer slices, and much taller slices, than previous work, reducing both milling time and material waste by 42% compared to prior automatic approaches, and by 22% compared to semi-manual ones.

For most commercially available millable materials, one of the precut slab dimensions — which we refer to as *height* — is significantly smaller than the others. To facilitate fixture placement and to reduce material waste we place the slabs into the milling chamber vertically, aligning the milling axis with the slab height. Notably, this placement requires bounding the height of each slice along the DHF axis to be below the slab height. We allow the slice-axes with respect to the decomposed geometry to vary arbitrarily, and compute the slice-axes that promote compactness and maximize slice height as part of the decomposition process (Figure 5d).

We produce an assemblable, bounded-height DHF decomposition by leveraging the following observations.

Millability requires both the interior (shared with other slices) and exterior (shared with the input object) surfaces of each slice to satisfy the DHF requirement with respect to a common axis. We note that convex shapes satisfy the DHF constraints for *any* axis (Figure 4a); consequently a slice defined by intersecting an open

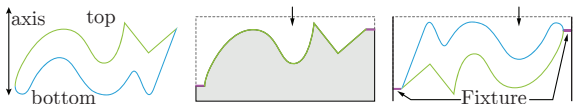


Fig. 2. DHF surface milling (along the vertical axis) (left to right): input DHF surface, surface portion milled from the top starting from a precut slab, portion milled from the bottom after flipping (purple lines show fixtures).

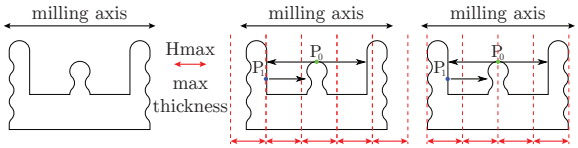


Fig. 3. The shape on the left is not a DHF surface with respect to the horizontal axis, but satisfies the LDHF criterion with respect to this axis – cutting it into slices with the illustrated or narrower heights is guaranteed to produce DHF slices with respect to this axis, as for any point on a slice either the left or right occluder will *always* be on another slice.

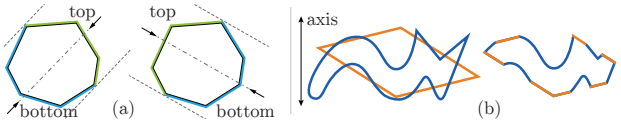


Fig. 4. (a) A convex shape is a DHF surface for any choice of axis (here illustrated for two choices). (b) An intersection of a DHF shape (blue) and a convex shape (orange) remains DHF.

or closed convex shape with a DHF surface remains DHF with respect to the surface's axis (Figure 4b). We thus adopt a binary-space partition (BSP) strategy as a key element of our decomposition algorithm; as each BSP tree cell is convex, a volumetric decomposition produced by slicing the input object using a sequence of cut planes is guaranteed to satisfy the DHF requirement as long as the exterior surface of each slice satisfies this requirement for some axis (Fig. 4b). BSP decompositions are also guaranteed to be assemblable [Luo et al. 2012].

We further observe that the slice height constraint H_{max} allows us to predict which portions of an object's surface can be guaranteed to satisfy the the DHF requirement with respect to a given axis *once sliced*, without pre-slicing them: a surface point may violate the DHF requirement with respect to an axis after slicing *only* if the sum of distances from said point to the just-above and just-below occluders along the axis is below H_{max} (Figure 2, bottom surface). We refer to surfaces that satisfy this property as *Locally* DHF, or LDHF (Figure 3); see Section 3.2 for details.

This observation motivates our two-step decomposition process: we first apply cuts to form large volumetric *blocks* whose surfaces satisfy the LDHF requirement with respect to a corresponding axis (Figure 1b), then cut each *LDHF block* into as-tall-as-possible bounded-height slices that satisfy the DHF criterion with respect to that axis (Figure 1ac). Separating block slicing from block computation enables us to better control the height and number of the individual slices, reducing overall slice count and producing taller slices, and in turn reducing milling time and material waste.

We enable an end-to-end computational milling pipeline by algorithmically packing the slices to fit into the milling machine chamber, and computing fixtures that hold the packed slices in place during milling and flipping (Figure 1c, Section 6).

We demonstrate the applicability of our method by manufacturing six different artifacts using precut slabs of wood, foam, and medium-density fibreboard (MDF). We showcase its adaptivity by creating decompositions with different user-specified properties. Finally, we perform a range of ablation studies highlighting the advantages of our method over prior art and baseline alternatives. These studies demonstrate that our method produces decompositions with on average 43% fewer slices and 70% taller median slices. Our milling simulation demonstrates that these improvements reduce fabrication time and material waste by over 42% on average (Section 9).

2 RELATED WORK

Decomposition for Fabrication. Volumetric partitioning has been extensively used to overcome a range of manufacturing hardware constraints and to widen the range of fabrication techniques applicable to a given input [Bickel et al. 2018; Livesu et al. 2017; Medeiros e Sá et al. 2016]. Most methods seek to minimize part count since decreasing it typically decreases fabrication time and material waste. They also seek to avoid generating parts that are smaller than necessary: thin parts can be fragile [Livesu et al. 2017; Muntoni et al. 2018] and milling parts which are significantly shorter than the workpiece height increases milling time [Rattat 2017].

Several methods seek to minimize part count while satisfying an upper bound on part size stemming from the need to fit these parts into a a machining chamber [Alemanno et al. 2014; Hao et al. 2011; Luo et al. 2012; Medellín et al. 2007; Song et al. 2016, 2015; Yao et al. 2015]. Our setting requires the slices to both satisfy maximal sizing constraints and be millable; none of these methods account for millability constraints. Mahdavi-Amiri et al [2020] decompose volumes into carvable elements: starting with a slab of material, elements are carved away by milling the slab from different carving directions, leaving the target object. They assume that these objects can fit into a single material slab smaller than the milling chamber.

Assemblability. The ability to assemble parts together to form the target model is a critical requirement when decomposing models for subsequent fabrication. Existing methods use a wide range of strategies to satisfy this requirement [Bickel et al. 2018; Livesu et al. 2017; Medeiros e Sá et al. 2016]. Models partitioned using cut-through planes [Attene 2015; Chen et al. 2015; Hildebrand et al. 2013; Hu et al. 2014; Luo et al. 2012] can always be assembled using an assembly order that reverses the cutting order. By employing cut-through planes we guarantee assemblability as a byproduct of decomposing our input shape into DHF slices.

Surface to Volume Segmentation. Recent methods [Araújo et al. 2019; Yao et al. 2017] partition shapes into assemblable parts whose exterior boundaries are defined by an input surface segmentation. They compute parts whose interior boundaries satisfy height field constraints with respect to axes computed based on assemblability considerations. Starting from an HF surface segmentation [Cook 1984; Doggett and Hirche 2000] these methods can produce parts consisting of two HF surfaces, albeit with *different* HF axes. Such parts are not suited for double sided milling and it is not clear how to design fixtures to mill them. Forcing the interior boundaries generated by such methods to satisfy HF requirements with respect to the same axis as the exterior surface would likely violate assemblability.

Convex Decomposition. Convex objects are DHF by definition; thus exact convex decompositions satisfy the DHF constraint. While computing minimal exact convex decompositions is NP-hard [Chazelle 1984], numerous methods [Asafi et al. 2013; Kraevoy et al. 2007; Lien and Amato 2007] compute coarse *approximately convex* decompositions. Convexity is a much stricter constraint than required for milling purposes (see Figure 2; while the object in this figure satisfies the DHF constraints as-is, it is far from convex); enforcing it is likely to require a lot more slices.

Cylindrical Decomposition. Layered manufacturing research uses parallel planes to cut objects into maximally tall slices whose geometry is well-approximated by generalized cylinders [Houtmann et al. 2007; Jun et al. 1998; Tyberg and Bohn 1998]. Hildebrand et al. [2013] decompose objects into equal thickness slices that approximate generalized cylinders using cuts aligned with one of three orthogonal directions. While generalized cylinders satisfy the DHF constraints, cylindricity is a more stringent constraint; enforcing it is unnecessarily restrictive and leads to needless fragmentation (while the shape in Figure 2, top is DHF, it is far from cylindrical). It is unclear how to extend these strategies to DHF slicing.

Uniform Parallel Slicing. Commercial tools [Autodesk 2020; Cirtes 1991; Schmidt and Singh 2010] enable users to slice objects using evenly-spaced parallel cuts (Figure 5a). As the cut density grows, this strategy becomes increasingly likely to produces slices which approximately satisfy HF or DHF constraints with respect to the cutting plane normal. Some of the tools, e.g. [Cirtes 1991], automatically compute a slicing direction expected to lead to a smaller slice count, and allow users to manually decompose shapes into blocks with different manually-specified slicing directions. Automatically achieving a desirable DHF approximation precision using this method often requires forming slices which are much thinner than the input material slabs; in our experiments the median slice height using this method was half of the height-bound used. This results in excessive slice counts, significant material waste, and high milling times (Figure 5a, Section 9). Our method employs differently oriented cutting planes for different parts of the models; it computes these plane orientations automatically, producing both fewer and better-sized slices than the alternatives (Figure 5d).

Single Height-Field Decomposition. Pyramidal, or height-field decomposition of general volumes, produces parts millable using a single milling pass. Exact pyramidal decomposition into a minimal number of parts is NP-hard [Fekete and Mitchell 2001]. Early approaches for approximate HF decomposition [Hu et al. 2014] cannot control the degree to which the resulting parts satisfy the height-field constraints; their milled outputs can uncontrollably deviate from the target geometry, making the method unsuitable for milling. Alemanno et al. [2014] use semi-manual decomposition and expect users to employ sophisticated modeling tools such as Blender to create the base structures that define the decomposition. Hernholz et al. [2015] generate HF shells rather than volume decompositions, and may produce non-assemblable parts. Gao et al. [2015] propose

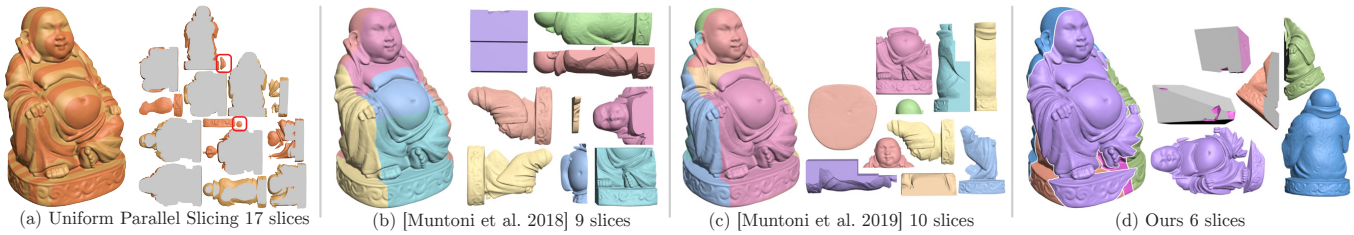


Fig. 5. Alternative decomposition approaches for milling (a-c): (a) uniform parallel slicing, (b) HF decomposition [Muntoni et al. 2018], (c) semi-manual HF decomposition [Muntoni et al. 2019]. Our method (d) produces fewer slices when using the same maximal slice height and milling precision (allowed deviation from perfect DHF/HF slices) thresholds, and avoids tiny slices (highlighted in red in a). See Table 3 for full statistics.

an HF partition method only suitable for shapes that can be described by a union of height fields defined over the faces of a cuboid. Muntoni et al. [2018] automatically compute assemblable HF decompositions of general shapes, overcoming the shortcomings of prior techniques. Unfortunately, their method often produces very fine decompositions with multiple tiny parts (Figure 5b), motivating the semi-manual approach of [Muntoni et al. 2019]. While this latter method allows expert users to obtain coarser outputs (Figure 5c), the authors report that users spend half an hour or more using their interface to decompose even medium complexity shapes, and require dedicated training to do so. By replacing the single HF constraints with the more general DHF ones, our fully automatic method is able to produce coarser decompositions with much taller slices on average than those of both Muntoni et al. [2018] and Muntoni et al. [2019], given similar milling precision and slice height constraints; see Figure 5 and Section 9. These improvements lead to significant material and fabrication time savings (Section 9).

Fixtures. Keeping workpieces stable during milling and manipulating them between milling passes require fixtures [Bakker et al. 2013; Bi and Zhang 2001; Trappey and Liu 1990]. Our fixture design (Section 6, Figure 1, f (top)) follows milling practices designed for DHF shapes with plane-separable upward and downward pointing surfaces, extending them to handle more general DHF shapes.

3 PROBLEM STATEMENT AND OVERVIEW

3.1 Problem Statement

The input to our method is a closed surface represented using a triangle mesh. Our goal is to create a physical copy of the shape enclosed by this surface using 3-axis CNC milling, starting from slabs of material with a fixed height H_{max}. Satisfying this goal requires solving a constrained volumetric partitioning problem where the produced parts, or *slices*, must satisfy the following requirements:
Milling Axis: Each slice has an associated milling axis.
Bounded Height: The height of each slice along the milling axis is at most H_{max}.
Double Height-Field: Each slice is millable using two passes along the milling axis - one from above and one, from below. In other words, each point p on the slice surface is accessible from either the above or below direction along the milling axis (Figure 2, top).
Coverage: The outer surface of the union of the slices must cover the surface of the input model.
Assemblability: There should exist an assembly order that we can follow to assemble separately manufactured slices into the desired union.

Subject to these constraints, we want the output partition to optimize the following objectives:
Count: To speed up processing and reduce material waste, we want the number of slices produced by the decomposition to be small.
Size: Milling thinner slices requires removing more material, and consequently requires more milling time; such slices are also more likely to break during processing. We seek to maximize slice height and reduce the prevalence of very thin slices whose height is below a user-provided threshold H_{min}.

These two objectives may argue for different solutions. We prioritize slice count minimization over the formation of tall slices, as slice count typically impacts milling time and waste more than slice height; however, we let slice count increase if doing so allows us to reduce the number of extra-thin slices. As minimal coverage problems are NP-hard [Cormen et al. 2001], we cannot expect to achieve the smallest possible slice count.

Finally, we note that it makes no sense to require the decomposition precision to be more accurate than the resolution supported by the milling machine, and that users want to control the precision A_{max} to which the DHF property is satisfied. For the remainder of the text, references to DHF slices will refer to slices which satisfy the DHF requirements within A_{max} tolerance.

3.2 Algorithm Overview

We develop a slicing method (Figure 6) accounting for these considerations by leveraging the key observations discussed in Section 1. Recall that an object defined by the intersection of a convex shape (i.e. an intersection of half-spaces) with a DHF surface defined with respect to a given axis, remains DHF with respect to this axis (Figure 4b). We also recall that slices formed using a sequence of cut-through planar cuts can always be assembled using an assembly order that reverses the the cutting order [Luo et al. 2012]. We consequently use the intersections of half-spaces formed by *cut-through planes* to define the interior boundaries of the DHF slices.

Slicing DHF Shapes into Bounded-Height Slices. Before discussing our algorithm in its entirety, we first consider the simpler case of an input shape that satisfies the DHF constraints for some axis $\mathcal{A}$ (Figure 7). In this scenario, our goal can be cast as cutting the shape into a minimal number of tall slices whose height along the axis is at most H_{max}.

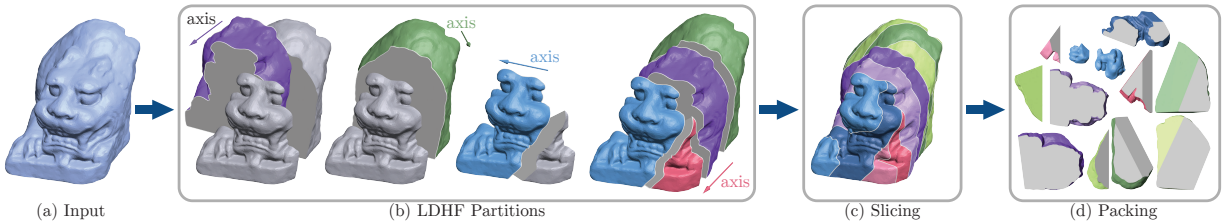


Fig. 6. DHFSlicer overview: Our first stage (Section 5) incrementally partitions the input (a) into LDHF blocks (b) defined with respect to the illustrated axes. The second stage (c, Section 4) slices each block into well-sized, bounded height slices, which are subsequently packed for double-sided milling (e, Section 6).

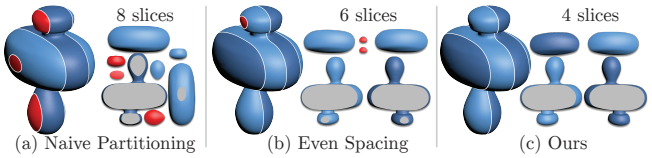
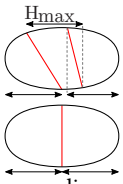
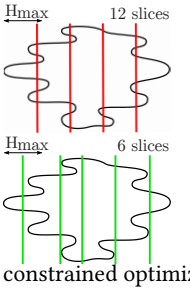


Fig. 7. Naively partitioning a DHF shape into slices by placing cuts at H_{max} intervals starting from the front-most (or back-most) point along the DHF axis (a), or evenly spacing them (b), can produce both very thin (in red) and redundant slices; (c) our discrete-continuous optimization generates fewer (4 slices versus eight (a) or six (b) in this example) and taller slices.

Fig. 8. Using LDHF blocks (b) instead of DHF ones (a) keeping the rest of the process the same introduces significantly fewer slices (18 vs 26 in this example), and reduces the prevalence of thin slices.

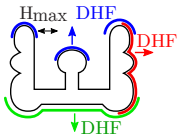
Due to the maximal height constraint, for most inputs the slice count in this setting is minimized when the cutting planes used are orthogonal to the axis (see inset and Appendix A). We therefore cast our goal as slicing the shape using planes orthogonal to the axis $\mathcal{A}$ at H_{max} or smaller intervals while minimizing slice count and avoiding shorter than necessary slices.





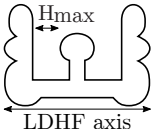
As the inset on the left and Figure 7 demonstrate, this is nontrivial when the input shapes have multiple height extrema along the axis direction; sub-optimal choices of cut locations can lead to formation of both extra short and redundant slices. We produce a compact set of as-tall-as-possible slices and avoid unnecessarily thin slices by casting and solving the cut placement problem as a mixed discrete-continuous constrained optimization problem (Section 4, Figure 7c).

LDHF Blocks. We now consider the general case. A natural approach would be to first decompose general shapes into DHF blocks using cutting planes, then slice those blocks using axis-aligned planes as above. Since interior block boundaries formed by cut-through planes satisfy the DHF constraints for any axis by construction, assessing if a block is DHF only requires us to evaluate whether its exterior boundary is DHF. This suggests first identifying continuous regions on the input surface that satisfy the DHF requirement with respect to some axis and that are separable from the rest of the shape using one or more planes, and then separating blocks bounded by such regions from the input shape using these bounding planes, one block at a time.



A major drawback of this approach is that, on many models, both DHF surface regions and their corresponding blocks are likely fairly small (see inset). This approach results in a high block count, leading in turn to a high final slice count, limiting our ability to obtain well-sized slices during the bounded-height slicing step (Figure 8a).

We overcome this challenge by utilizing blocks whose exterior boundaries satisfy *local*, rather than global, DHF constraints with respect to some axis. We define a point p as *locally DHF (LDHF)* with respect to an axis if it is either (a) directly accessible along this axis from either above or below (DHF), or (b) if the distances from a point to its left and right occluders along the milling axis sum to a value that is greater than H_{max} (Figure 3). We note that when slicing an object into slices whose height along the axis is bounded by H_{max}, any pair of points on the original surface that are more than H_{max} away from each other along the axis are **guaranteed** to be on separate slices (Figure 3). Consequently, **any point that satisfies the LDHF criterion with respect to an axis is guaranteed to be on a different slice than either one or both of its occluders following bounded-height slicing**. A block whose outer boundary satisfies the LDHF constraints with respect to an axis can therefore be sliced into DHF slices using **the same** bounded-height slicing method (Section 4) as a DHF block.



By replacing the DHF criterion with the LDHF one, we dramatically increase the size of the continuous surface regions satisfying our block decomposition criterion with respect to many potential axes (the previous example in the inset is entirely LDHF for the horizontal axis); this significant increase in region size allows for coarser block decomposition with higher slice quality (Figure 8b).

Iterative LDHF Decomposition. We perform LDHF block decomposition using an iterative process. Each decomposition iteration locates multiple potentially separable LDHF blocks. It then separates the block whose separation is predicted to lead to the best decomposition (Figure 6b). This process is repeated until the input is fully separated into LDHF blocks. In selecting the block to separate, we account for both the anticipated quality of the separated

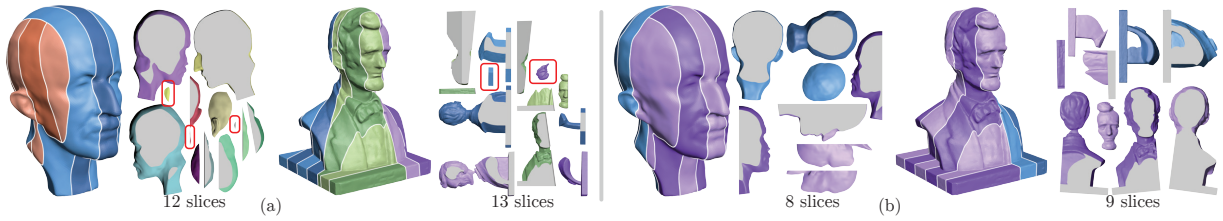


Fig. 9. Greedy coarse decomposition where each iteration separates the block deemed best (a) produces increasingly fragmented blocks (and consequently slices) as it proceeds. Our LDHF surface-segmentation-based method (b) balances the quality of current and future blocks (and slices), avoiding fragmentation.

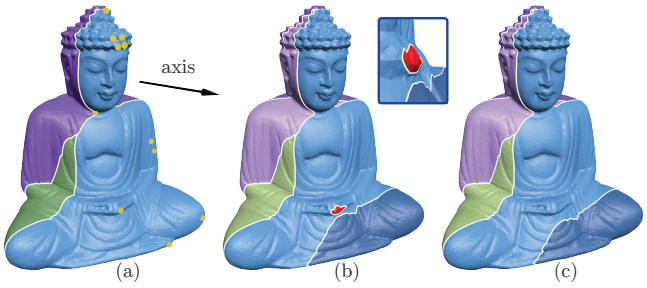


Fig. 10. (a) LDHF block with local height extrema along the milling axis highlighted in yellow; (b) Naive slicing produces an undesirably thin slice (red); (c) Well-sized slicing output produced by our method.

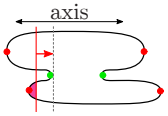
block itself (slice count and height) and the estimated quality of the predicted optimal partition of the remaining model. We predict this optimal partition by segmenting the surface of the processed models into balanced size LDHF charts designed to resemble the exterior surfaces of the best potential output blocks (Figure 11); we then separate the block whose exterior surface matches the best chart in this partition.

Post-process. Once the models are fully sliced, the slices are packed and fixed in place for milling (Section 6), and are optionally augmented with registration marks (Section 7). The following sections discuss the core steps of the method in more detail.

4 BOUNDED-HEIGHT SLICING OF (L)DHF BLOCKS

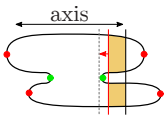
Given a block which satisfies the DHF or LDHF requirements with respect to a chosen axis, we aim to partition it into slices satisfying the bounded height requirement (and recalling that on LDHF blocks such slices satisfy the DHF requirement by construction). Among the solutions satisfying this constraint, we prioritize ones that minimize the prevalence of slices shorter than our minimal height threshold H_{min}, keep overall slice count low, and produce as-tall-as-possible slices (we prioritize count over height). Since all cuts are performed orthogonally to the milling axis, the computation of the slicing locations we seek for can be formulated as a discrete-continuous 1D optimization problem, whose variables are the number of cuts, and the locations of these cuts along the milling axis.

Fixed cut count. Before discussing our complete formulation, we consider the case where the number of cuts k is known. We describe the extension to the general case below. We formulate our goal of avoiding extra short and redundant slices in terms of desirable and undesirable cut locations. To assist the computation we define a height function across the block surface by projecting the locations of all surface vertices to the milling axis, and defining vertex height using its 1D coordinate along this axis.



Our key observation is that the topology of our height function, specifically its critical points, provide us with strong cues as to which cut location choices are likely to lead to the formation of undesirable slices. We first note that away from height-function saddles (green dots in inset) and convex extrema (red dots), the only factor that impacts slice count is the number of cuts; changing cut locations away from these critical points does not change slice count.

We further note that if we minimize cut count and optimize for cut spacing that maximizes the heights of all slices, the only practical scenario where avoidable extra thin slices can emerge is when a cut is placed within a distance of H_{min} or less below a convex local height function maximum, or respectively above a local convex minimum (red slice in the inset). We can consequently reformulate our goal of avoiding thin slices as one of avoiding cuts within H_{min} wide *extremum intervals* below, or respectively above, such extrema, computed as described below. Notably, preventing cuts just below convex maxima or just above convex minima is also likely to reduce the overall slice count. Lastly we observe that in the vicinity of saddle points with upward (downward) pointing normals, avoiding cuts just above (below) the saddle points is likely to further reduce slice count. Specifically, we observe that given two connected mesh components separated by an upward oriented saddle, consecutive cuts placed above the saddle bound two separate slices (yellow in the inset);



in contrast if one cut is placed above and one below the saddle, they bound only a single slice. We promote cuts that form fewer slices by defining *saddle intervals* just above (below) upward pointing (downward pointing) saddle points and discouraging cuts within these intervals.

Forbidden Intervals. Using this heuristic to relate cut location to slice count, we can formulate our goal as computing cutting plane locations that produce as-tall-as-possible slices, subject to our maximal height bound and constraints that prevent plane placement inside extremum or saddle intervals. We detect extrema and saddles using the basic formulation described in [Edelsbrunner and Harer 2008], which works well enough for our setting. Given the segment

$S = [s_l, s_h]$ between the two farthest points on the block along the axis, where $s_l < s_h$ with respect to the up axis direction, we define the set of forbidden intervals $f_m = [f_m^l, f_m^h] \in F$ along this segment as all intervals $[h^+ - H_{min}, h^+]$ of length H_{min} below convex height maxima h^+, and above convex height minima h^-, $[h^-, h^- + H_{min}]$, and all intervals of length H_{min} above saddles s^+ with upward pointing normals, and below saddles with downward pointing ones s^-, $[s^+, s^+ + H_{min}]$ and $[s^- - H_{min}, s^-]$. Note that by construction this set always includes intervals of length H_{min} next to segment end points, $[s_l, s_l + H_{min}]$ and $[s_h - H_{min}, s_h]$. Finally, we merge any partially overlapping forbidden intervals into one.

We note that if any of the intervals produced by this merging step are longer than H_{max} ($f_m^h - f_m^l > H_{max}$) the slicing problem becomes over-constrained, as it is no longer possible to compute bounded-height slices without placing cuts inside these intervals. If this situation occurs we locally relax the minimal slice height requirement by halving the lengths of the intervals that jointly formed the offending one and recomputing the combined intervals. In our experience one halving step was sufficient to resolve all offending intervals for all inputs tested.

Solution. Our goal of computing the best cut locations $c_0, \ldots, c_{k-1}$ along the axis can now be formulated as:

$$\min_{\{c_i\}} F_{cut} = ((c_0 - s_l) - H_{max})^2 + ((s_h - c_{k-1}) - H_{max})^2 + \\ + \sum_{i=0}^{k-1} ((c_i - c_{i-1}) - H_{max})^2 \quad (1)$$

subject to

$$0 \le c_0 - s_l \le H_{max}, \quad (2)$$
$$0 \le c_i - c_{i-1} \le H_{max} \ \forall i \in [1, k-1], \quad (3)$$
$$0 \le s_h - c_{k-1} \le H_{max}, \quad (4)$$
$$c_i \notin f_m \ \forall i \in [0, k-1], \ \forall f_m \in F. \quad (5)$$

Our solution needs to satisfy the non-convex constraints of preventing the cuts being placed inside forbidden intervals. Solving optimization problems with such constraints, in general, is known to be NP-hard [Garey et al. 1976]. In our scenario the number of cuts and the number of forbidden intervals are known to be small, motivating us to adopt the following brute-force strategy. We first convert the forbidden interval constraints into *allowed* interval constraints, by computing all intervals $a_j = [a_j^l, a_j^h] \in A$ where cuts *can* be placed. We set $A = S \setminus F$. We then observe that if we know *a priori* which cuts should be placed in each allowed interval, then the problem we seek to solve reduces to a constrained quadratic minimization problem, which can be solved using standard methods:

$$\min_{c_i} F_{cut} \text{ subj. to}$$
$$0 \le c_0 - s_l \le H_{max}, \quad (6)$$
$$0 \le c_i - c_{i-1} \le H_{max} \ \forall i \in [0, k-1], \quad (7)$$
$$0 \le s_h - c_{k-1} \le H_{max}, \quad (8)$$
$$a_{j(i)}^l \le c_i \le a_{j(i)}^h \ \forall \ i \in [0, k-1]. \quad (9)$$

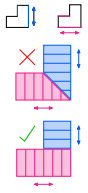
To obtain the best cuts to intervals assignment (for each i, defining the allowed intervals $j(i)$), we consider each possible assignment of cuts to allowed intervals, explicitly solve the minimization problem for each assignment, and then select the solution minimizing F_{cut} globally. We prune the search space by avoiding assignments violating the maximal slice height constraints, or cut order (equations 6 to 8). Since we enumerate all possible combinatorial possibilities, and each fixed-assignment problem can be solved optimally, this approach leads to an optimal solution.

Solving for Cut Count. The number of cuts k necessary to obtain the desired set of slices is not *a priori* known. We solve for a slice count k that satisfies the constraints with the smallest slice count as follows: we first attempt to find a solution with $k = \lceil (s_h - s_l) \rceil - 1$ (the smallest cut count required to satisfy our maximal height constraints) and increase k by one at a time until a solution is found. To ensure termination, if even after doubling the original value of k, no solution is located, we return a naive solution where all slices are set to have the same thickness. In practice, this condition was never triggered in our experiments. We could potentially attempt to further reduce slice count when the number of slices s is larger than $k + 1$, by running the method with $k + 1$ as the cut count, and using the resulting solution if it contains fewer slices. We found that in practice this or similar extensions did not lead to further improvement.

Figures 7 and 10 contrast slicing outputs computed using our approach with naive cut placement.

5 COARSE LDHF PARTITION

Our partitioning method for general shapes is motivated by the observation that the outer surfaces of most shapes are dominated by large contiguous regions which satisfy the LDHF property with respect to some axis. For example, in Figure 11, all non-red triangles in (a) are LDHF for axis A_1, and all non-red triangles in (b) are LDHF for axis A_2. Sub-regions of these LDHF regions that can be separated from the rest of the model using cutting planes (e.g. the brown and blue strips in Figure 11a) can serve as natural exterior boundaries for LDHF blocks. Our goal is to select the sub-regions and planes that would produce a compact decomposition of the input into maximally large LDHF blocks that, in turn, can be sliced into a small number of slices.



We use a decomposition strategy centered around the use of planar cuts that separate LDHF blocks from the rest of the model one at a time. Specifically, we separate blocks whose exterior surface satisfies the LDHF criterion with respect to some axis using cutting planes *orthogonal* to this axis (inset, bottom).

While using generically oriented planes as partition boundaries may allow for formation of larger or more evenly sized LDHF blocks (inset, top), even basic examples such as the one in the inset show that using planes orthogonal to the axis (inset, bottom) typically decreases the final overall slice count. The exterior surfaces of such blocks are defined by *LDHF strips*: contiguous surface regions bounded by axis-orthogonal planes that are entirely LDHF with respect to the candidate axis (Figure 11a). Each strip can be turned into a block by placing axis-orthogonal separating planes at its

farthest points along the axis; if the strip has a single boundary it is separable using one plane (Figure 11a, top), and is separable using two planes otherwise (Figure 11a, bottom).

Our goal of selecting a best block to separate can now be recast as selecting the best LDHF strip to use as the exterior surface of such a block. When selecting the strip to use, we seek to balance the quality of the block it encloses against the quality of the decomposition of the remainder of the model (the connected components left after cutting the block out), and more specifically on our ability to compactly decompose and then slice this remainder without introducing unnecessary thin slices. The quality of each potential block can be estimated by analyzing the surface of its corresponding LDHF strip; however assessing the impact of our choice on the decomposition of the remainder requires a global analysis of the input being processed. We obtain a solution by approximating the computation of the searched-for volumetric partition via surface segmentation (Section 5.3). We compute a segmentation whose charts are designed to correspond to the exterior surfaces of the desired optimal blocks, and which is guided by the computed LDHF surface strips (Figure 11c).

The key to the success of this strategy is our ability to reliably predict the quality of the final decomposition output throughout decomposition iterations. This prediction ability is well illustrated by the similarity between the output of our first iteration segmentation (Figure 11d) and our final decomposition output (Figure 6b), and hinges both on our choices of segmentation criteria (Section 5.3) and the inherent locality of the LDHF accessibility computation. We explicitly note that this locality property does not hold for HF or DHF decompositions.

We now describe the method in more detail.

5.1 Decomposition Flow

Each decomposition iteration checks if the currently processing model is entirely LDHF with respect to one or more axes in a densely sampled set of possible axes, computed as described in Section 8 and, if so, terminates and associates the model with the axis predicted to require the minimal number of slicing cuts. Slice count is predicted by measuring the *axis-aligned diameter* (the difference between the maximum and minimum of the model's height function defined with respect to this axis): we expect to require fewer slices when cutting a model orthogonally to a shorter axis. Otherwise, the decomposition algorithm proceeds to identify LDHF stripes and corresponding blocks that can be separated from the current model (Section 5.2), and separates the block deemed best (Section 5.3). The method then iterates separately on each remaining connected component. We typically perform fewer than ten separation steps, producing under a dozen blocks (full statistics in tables 1, 2). Note that for any given input one can always locate at least one block to separate since the immediate neighborhoods of the height-function extrema along each axis always satisfy the LDHF criterion for this axis.

5.2 LDHF Strip Computation

At each iteration, we test each triangle on the current model to see if it satisfies the LDHF criterion with respect to each of the densely sampled axes (Section 8). We then compute the LDHF strips that define the exterior surfaces of maximal LDHF blocks that can be separated from the current model. For each potential axis direction, we first project to the axis all triangles that violate the LDHF property and then identify intervals along the axis which are free from LDHF-violating triangles. We define the maximal *LDHF strips* to include all triangles whose projections lie within these free intervals (Figure 11a).

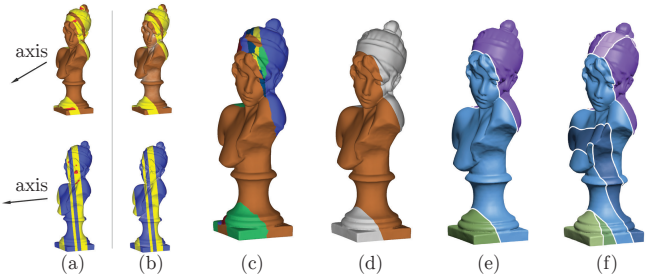


Fig. 11. Single separation iteration: (a) LDHF strip computation: non-LDHF regions (red), and LDHF strips (brown and blue) for a pair of axes A_1 and A_2. Blocks formed by separating these strips using axis-orthogonal planes are, by construction, LDHF. (b) Same LDHF strips, with triangle luminance reflecting relative accessibility (Equation 11; lighter triangles are less accessible). (c) Segmentation using per-axis labels (charts affiliated with A_1 in brown, and ones affiliated with A_2 in blue). (d) Selected strip (affiliated with A_1). (e) LDHF decomposition, note the correspondence between the blocks selected and the larger charts in (c). (f) Final slices.

Avoiding Thin Slices. We avoid the formation of excessively thin remainder connected components following block separation. We locate convex height function extrema along each axis and if the distance from the top end of a free interval to the closest convex maximum e_m above the interval, or from the bottom end of a free interval to the closest convex minimum below the interval, are less than H_{min}, we move this interval endpoint inward, placing them H_{min} away from the corresponding extremum.

5.3 Surface Segmentation.

We seek a compact segmentation consisting of large contiguous charts which satisfy the LDHF property with respect to some axis, and which overlap with LDHF strips defined with respect to this axis. We express these criteria via a classical graph-cut formulation. We treat each potential axis direction as a label, define for each triangle $t \in T$ (where T is the set of *original* exterior surface triangles on the currently processed mesh) a unary cost $c(t)$ of associating it with that label, and assign a binary compatibility cost $c(t, t')$ to each pair of triangles t and t' that share an edge $(t, t') \in E$. Our unary cost reflects the suitability of associating this triangle with a strip defined with respect to this axis, while the binary cost promotes compactness. Using these costs our goal can be cast as minimizing

$$F = \sum_{t \in T} c(t) + \sum_{(t,t') \in E} c(t, t')$$

We compute the labeling that minimizes F using the optimization code of [Boykov et al. 2001; Boykov and Kolmogorov 2004] which provides an approximate solution to this NP-hard problem. While

in theory we could get a segmentation even more reflective of the optimal solution by assigning per-strip instead of per-axis labels, doing so would dramatically increase computation costs.

Unary Cost. We design the per-triangle labeling costs $c(t)$ to produce a labeling which reflects a surface segmentation corresponding to a likely and desirable volumetric partition of the input model. This labeling cost accounts for both the impact on the overall segmentation of associating each individual triangle t with a particular axis A, denoted $c(t, A)$, as well as the overall quality $C(A)$ of the strips associated with this axis:

$$c(t) = c(t, A) + C(A). \tag{10}$$

We observe that we are likely to need to use one of the axes that each triangle currently satisfies the LDHF property for to mill this triangle. Consequently, the smaller the set of axes that a portion of the surface is accessible from, the more likely we are to need to use strips associated with one of these axes in our final partition. Delaying the use of one of these axes can lead to undesirable output fragmentation (Figure 9). We avoid such fragmentation by defining the cost of a triangle within each strip associated with the axis A based on the number of alternative milling axes that can be used to mill this individual triangle:

$$c(t, A) = w(t) \cdot Ar(t)/(\sum_t Ar(t) \cdot w(t)) \tag{11}$$

$Ar(t)$ is the area of the triangle t and $w(t)$ is the number of directions the triangle is *not* LDHF for normalized by the average of these numbers across all triangles.

We prefer forming larger blocks, as using them is likely to produce fewer final slices, and thus prefer axes with larger associated strips; we also prefer axes with shorter corresponding diameters, as blocks formed using them are likely to require less slicing. To reduce fragmentation we bias the area component of the axis cost to account for the number of axes that can be used to mill each triangle, and formulate the overall axis cost as:

$$C(A) = \alpha \cdot (1 - (\sum_{t \in Sl(A)} Ar(t) \cdot w(t))/(\sum_t Ar(t) \cdot w(t))) + (1 - \alpha) \cdot L \tag{12}$$

Here L is the ratio of the object's diameter along the axis to its bounding box diameter and $Sl(A)$ are the triangles in the strips associated with the axis A. The first term prioritizes axis choices that lead to larger charts and incorporate fewer accessible triangles. The second term serves as a proxy for the number of slices we expect each block to be cut into: we expect to need fewer slices when the diameter along the milling axis is smaller. We set the weight $\alpha = 0.9$ to ensure that the cost is dominated by the area term, and that diameter differences only come into play when the area terms are nearly identical. We use an infinite unary cost (in practice $10,000$) for assigning a triangle to an axis if it is outside the LDHF strips associated with this axis.

Binary Cost. The compatibility cost $c(t, t')$ of two triangles is set to 0 if the two triangles are associated with the same label, and $l(t, t')$ (the length of the edge between t and t', normalized by the average edge length) otherwise. This choice is designed to minimize chart boundary length, thus indirectly minimizing the number of charts and labels used in the output segmentation. Since charts are used as proxies for blocks, minimizing chart count indirectly minimizes the number of output blocks.

Figures 11b and 11c show the per-triangle costs resulting from this computation and a typical surface segmentation obtained using this graph-cut framework.

Charts to Blocks. At each partition iteration we select the strip which maximizes the overlap area between the strip and the segmentation charts corresponding to the strip's axis, and separate its corresponding block from the model.

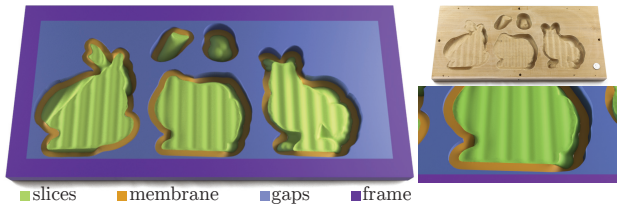


Fig. 12. Typical slice pack virtual and milled with fixtures, with a close-up of the registration patterns added to internal slice surfaces.

6 PACKING AND FIXTURES

We complete the digital fabrication pipeline by packing the slices and fixing them in place to enable stable double-sided milling and flipping the packs in-between the two milling passes. We orient all slices so that their axes are aligned with the vertical axis of the milling machine, and pack them into one or more rectangular packs using the method of [Lévy et al. 2002]. The width and length of each pack are determined by the smaller between the respective milling bed and milled slab dimensions. To enable unobstructed milling access to the slices and to accommodate fixtures, we space the slices during packing to leave sufficient gaps between them, and add additional space between slices and the sides of the milled slab. The size of these gaps, as well as all subsequent sizing decisions, are guided by machine and material specifications.

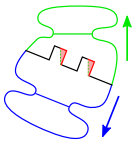
Our fixture computation generalizes the standard approach used for doublesided milling of DHF shapes. Like these methods, we first generate *silhouette loops*, or surface curves that separate areas on the slice accessible only from below from those accessible only from above, on each slice, and use these as a basis for our fixtures. We form a membrane surface by first offsetting the silhouette loops horizontally using a fixed distance, forming an offset surface, and then minimally offsetting this surface up and down to form the membrane. To enable easy slab flipping, we leave a fixed width frame around the perimeter of each milled slab; the height of the frame is the original slab height. We preserve the input slab geometry as-is inside all gaps between the membrane surfaces, and between the membrane and the frame. Fig. 12 illustrates the resulting fixtures. These fixtures provide a good balance between stability, milling time, and ease of removal. For most materials, the resulting membrane surfaces are thin enough to be cut off using scissors or a saw; surface bits that remain attached to the slices can then be sanded off. Leaving the material as-is once a small distance away from the slices reduces milling time and strengthens the fixtures.

Silhouette Loops. Our challenge when computing the fixture surface is to maintain access to the slices during milling; areas accessible from each side must remain accessible from that side. Traditional DHF milling settings assume that these areas can be separated using horizontal loops, and expect users to specify those loops manually. We compute the loops automatically and support inputs that do not satisfy the horizontality assumption. The loops we compute are designed to have minimal height oscillations, as such oscillations can make the slices and the membrane harder to mill. We generate maximally flat silhouette loops by formulating loop computation as a min-cut problem. Specifically, we associate each of the slice's triangles with either the *above* or *below* labels, and then use the boundary between the two labeled sets as the silhouette loop. We aim for triangles accessible from only one side to be associated with that side's label, and for the boundary to be as short, and thus as smooth, as possible. We formulate these goals as minimizing

$$\min_{l(t)} F_b = \sum_{t \in T} c(l(t)) + \sum_{(t,t') \in E} c(l(t), l(t')),$$

where $l \in a, b$ is the label assigned to the triangle t. $c(l(t))$ is set to infinity (10,000 in practice) if the triangle is not accessible with respect to the label's direction d_l, and to $1 - n \cdot d_L$ otherwise (here n is the triangle normal). We set $c(l(t), l(t')) = 0$ if $l(t) = l(t')$ and set it to 1 otherwise. We compute the labeling that minimizes F_b using the optimization code of [Boykov et al. 2001; Boykov and Kolmogorov 2004]. We apply Laplacian smoothing to the extracted loops to further reduce local oscillations.

7 REGISTRATION PATTERNS



Milled parts need to be accurately aligned and connected when assembling the final object. Many decomposition methods use protruding pin-like connectors that achieve both goals at once [Delebecque et al. 2008; Koyama et al. 2015; Luo et al. 2012]. Unfortunately the pin holes required to accommodate standard connectors between pairs of adjacent slices may be unmillable for many combinations of slice milling axes (see inset: inaccessible area highlighted in red). Even when millability can be satisfied, milling slices with tall protruding connectors out of fixed height slabs would require substantially reducing the height of the actual slices, leading to large increases in milling time and material waste. Instead, we help users accurately align milled slices prior to gluing them together by providing them with the option to compute subtle registration patterns (Figure 12a). The pattern we use consists of protrusions and concavities, and can accommodate the vast majority of pairwise slice axis combinations along interior boundaries shared by pairs of slices. We accommodate the extra height needed for forming registration patterns by subtracting the pattern height H_r from the maximal slice height H_{max} during decomposition.

Let us consider two slices with milling axes A_1 and A_2, sharing a common interface plane that we want to deform to generate a pattern that helps the registration of the slices. Without loss of generality, we assume that this slicing plane is orthogonal to the Z axis. We use a sine-wave deformation function:

$$F(x, y) = f \cdot H_r \cdot \sin(x/H_r) \tag{13}$$

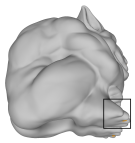
where H_r controls the amplitude of the pattern (how prominent it is), and f determines its frequency. In the models we fabricated we used $H_r = H_{max}/25$, resulting in patterns which were prominent enough for good registration without significantly increasing material waste. We set $f = tan(15°)$, which creates a pattern which is sufficient for registration, and can be applied to all milling axis pairs A_1 and A_2 for which the maximum angle ϕ that A_1 and A_2 form with Z is less than $\alpha = 75°$ (see Appendix A for the corresponding derivation).

To avoid deforming the outer surface of the model, we apply the pattern only in the interior of the surfaces shared by adjacent slices, and use a smooth linear transition between undeformed exterior boundary vertices and the deformed interior, using a similarly computed gradual slope that satisfies our slope constraints. In the rare case where $\phi > \alpha$ we do not apply the patterns. This would, for instance, be the case if one or both milling directions (A_1,A_2) is (near)-orthogonal to the normal of the common interface plane. Adding registration patterns only increases milling times by 5% to 10% in simulation (measured using Autodesk Fusion360 [Autodesk 2020], using our default milling settings.)

8 IMPLEMENTATION

Potential Axis Set. We discretize the set of possible milling directions using spherical Fibonacci sampling [Keinert et al. 2015]. We use 50 directions, and augment them with the three principal axes of each input model; this augmentation is motivated by the observation that many models are meaningfully oriented with principal axes, and hence those axes provide a potentially optimal milling direction choice.

Measuring The LDHF Property. We determine accessibility with respect to each axis using raytracing from uniformly sampled points across the surface, tracing five samples per unit of triangle area, and at least one sample per triangle. For each sample, we measure the distance to its nearest occluders along the up and down direction of the axis in question. A triangle is denoted as LDHF with respect to a given axis if, for all samples within this triangle, either no above or below occluders exist, or the sum of distances to those occluders is above H_{max} (when measuring the DHF property we only test for occluder existence). We accommodate the precision threshold A_{max} conservatively by offsetting each sample along the outward normal direction by A_{max} before performing the occlusion test (we constrain the offset samples to remain outside the object). The input meshes we process may contain tiny but deep concavities which are accessible from very few directions (such as the areas between the Gargoyle's toes in the inset). While such surface patches are often too small for a milling machine to mill accurately, forming slices that allow mill access to such patches may lead to a significant increase in slice count and decrease slice size. We avoid unnecessary fragmentation when assessing if a model or a slice are DHF/LDHF and when computing LDHF strips by ignoring surfaces patches that do not satisfy the DHF/LDHF criterion with respect to an axis if their area is below a fixed tiny percentage p_s of the original model's surface (we use $p_s = 0.00025$, unless indicated otherwise) The measurement choices and parameters above were chosen empirically, accounting for both computation time and



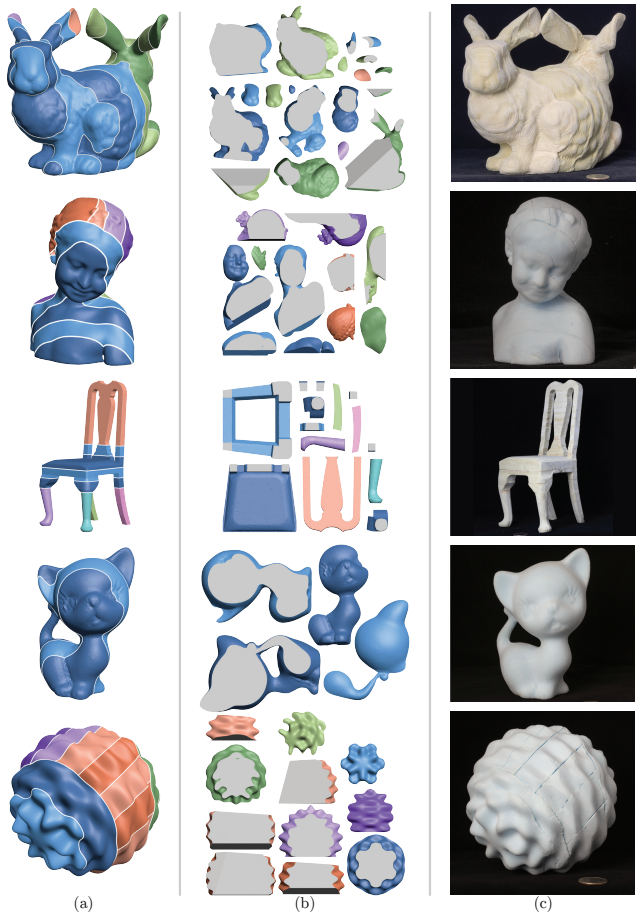


Fig. 13. Additional fabricated results. For each input we show the DHF slice decomposition, packed slices, and fabricated model. Statistics for the models are listed in Table 2.

output quality. Section 9 discusses the impact of changing these parameter values.

Mesh Processing. We use the libigl [Jacobson et al. 2018] implementation of exact Boolean operations [Zhou et al. 2016] to perform the cuts. To facilitate registration pattern formation we generate conforming surface meshes for all slices by remeshing all slices using Tetwild [Hu et al. 2018], labeling tetrahedra based on the slice their barycenter is in, and projecting all shared interface vertices to their common cutting planes.

9 RESULTS

Throughout the paper we demonstrate a range of fabrication-ready, bounded-height DHF decompositions on 21 models of varying geometric complexity decomposed with different parameter settings (Tables 1, 2, 3). These include high-genus models (*fertility*, *kitten*, *chair*), models with high-frequency fine details (e.g. the hair on *David*, the scales on the *dragon*), relatively smooth objects (*fertility*, *knot*), organic shapes (*kitten*, *bimba*), decorative objects (*bumpy sphere*, *Julia statuette*), furniture (*chair*), engineered shapes (*rocker arm*), and decompositions computed with both lax (Figures 15, 16)

	H_{max}	A_{max}	#blocks	#slices	min. height	med. height	precision
maxplank (Fig.9)	10%	0.50%	2	8	0.076	0.08	0.10%
lincoln (Fig.9)	10%	0.50%	2	9	0.057	0.09	0.33%
sapphoshead (Fig.11)	10%	0.50%	4	11	0.016	0.06	0.40%
dragon (Fig.14)	10%	0.50%	31	39	0.009	0.06	0.38%
feline (Fig.14)	10%	0.50%	11	27	0.002	0.06	0.40%
fertility (Fig.14)	10%	0.50%	7	19	0.037	0.08	0.35%
rockerarm (Fig.14)	10%	0.50%	3	5	0.056	0.07	0.15%
julia (Fig.15a)	3%	0.50%	1	28	0.024	0.03	0.10%
julia (Fig.15b)	10%	0.50%	4	11	0.081	0.09	0.10%
julia (Fig.15c)	20%	0.50%	4	9	0.065	0.13	0.10%
julia (Fig.15d)	10%	0.50%	4	11	0.081	0.09	0.10%
julia (Fig.15e)	10%	0.05%	11	21	0.014	0.08	0.03%
julia (Fig.15f)	10%	5%	1	8	0.088	0.09	1.08%
vaselion (Fig.19)	10%	0.50%	19	31	0.032	0.07	0.48%

Table 1. Statistics for results shown across the paper. Left to right: model, height bound H_{max}, precision tolerance A_{max}, number of output blocks, number of output slices, minimal and median output slice height, output precision. Since we operate on virtual models whose scale is essentially arbitrary, for consistency we express all distance metrics as percentages of object's bounding box diagonal. All outputs satisfy the precision tolerance.

and tight (Figure 1) slice height bounds. An additional collection of nine comparative examples is included in the supplementary material. We visualize the computed fixtures and the optional registration patterns in Figures 1 and 12, and include examples of slice packs with fixtures in the supplementary.

Parameters and User Control. In a typical milling setting users expect to control three core parameters: the maximal slice height H_{max}, the DHF precision A_{max}, and the minimal slice height H_{min}. The first is determined by the thickness of the workpieces used, the second is directly related to the specifications of the milling machine as well as the user's desired reproduction accuracy, and the third is typically impacted by material fragility. Since we operate on virtual models whose scale is essentially arbitrary, for virtual testing we specify and report these numbers as percentages of the length of the diagonal of the input model's bounding box. For consistency when reporting data for fabricated models, we convert user specified absolute height and precision values into percentages by normalizing with respect to the user specified target object diagonal length.

The typical ratios between low-end milling machine bed diagonal length to off-the-shelf reasonably priced material height is in the 10:1 to 30:1 range, motivating the choice of the typical height bounds H_{max} we test with. Assuming that users want to pack between 1 and 6 slices to process in one milling pass, the slice thickness should be in the range of 3% to 10% of the object's bounding box diagonal. On the examples shown, we set H_{max} to 10% of the bounding box diagonal and A_{max} to 0.5% unless specified otherwise. We use $H_{min} = 0.15 \times H_{max}$ as the default minimal slice height. We demonstrate the impact of changing this and other parameters on the slice count, and minimal slice dimensions for the Julia model in Figure 15. Our method seamlessly adjusts to different parameter combinations, with slice count predictably increasing as maximal slice height or DHF precision decreases. Increasing H_{min} (Figure 15d) predictably increases the height of the shortest slice, resulting in more even slice sizing, at the expense of an increase in slice count.

Evaluation. In evaluating decompositions for milling, the key metrics are the number of slices, the dimensions of the minimal slices produced, and the final DHF precision. We measure precision using the sampling-based occlusion test in Section 8. All our outputs

	H_{max}	A_{max}	#slices			min. height			med. height			precision			waste			milling time		
			M18	PS	Ours (Blocks)	M18	PS	Ours	M18	PS	Ours	M18	PS	Ours	M18	PS	Ours	M18	PS	Ours
Bimba (fabricated)	0.10	0.5%	42	34	11(4)	0.0003	0.0001	0.0612	0.06	0.03	0.08	1.1%	0.4%	0.4%	7.52	32.73	4.84	05:40:48	09:59:42	03:56:11
Bumpy (fabricated)	0.10	0.5%	86	14	11(4)	0.0022	0.0030	0.0627	0.03	0.05	0.09	0.8%	0.3%	0.5%	5.20	4.11	2.22	33:53:13	18:40:41	15:31:15
Bunnies (fabricated)	0.08	0.8%	148	25	19(4)	0.0002	0.0006	0.0154	0.02	0.05	0.06	0.5%	0.6%	0.7%	6.03	3.89	3.18	39:01:35	10:42:16	09:12:40
Chair (fabricated)	0.08	0.5%	42	10	14(6)	0.0063	0.0310	0.0419	0.03	0.05	0.06	0.4%	0.4%	0.1%	22.82	10.45	7.47	15:04:11	02:49:59	03:08:13
Julia (fabricated)	0.05	0.3%	89	24	26(2)	0.0060	0.0477	0.0320	0.04	0.05	0.05	0.3%	0.2%	0.1%	5.29	2.26	2.85	20:22:23	07:14:06	08:50:58
Kitten (fabricated)	0.13	0.5%	27	15	4(1)	0.0183	0.0010	0.1050	0.06	0.04	0.10	0.8%	0.2%	0.2%	9.72	8.70	3.26	19:46:46	12:06:03	05:42:33
Buddha2 (Fig.10)	0.15	0.5%	15	25	5(3)	0.0073	0.0012	0.0941	0.11	0.03	0.11	0.7%	0.4%	0.3%	5.31	14.42	5.66	13:17:14	18:41:18	07:51:59
David (Fig.17)	0.10	0.5%	39	56	28(18)	0.0026	0.0001	0.0393	0.06	0.01	0.08	2.0%	0.3%	0.4%	3.67	15.01	4.74	21:02:45	57:18:53	21:38:43
Gargoyle (Fig.17)	0.10	0.5%	64	51	22(7)	0.0003	0.0005	0.0150	0.03	0.01	0.08	0.9%	0.4%	0.4%	6.06	14.43	2.99	34:56:47	52:53:13	15:52:10
Guaje (Fig.7)	0.13	0.5%	15	6	4(1)	0.0012	0.0082	0.0883	0.07	0.10	0.10	0.0%	0.0%	0.0%	3.28	3.01	1.07	08:07:19	05:16:58	05:02:24
Knot (Fig.8)	0.08	0.5%	98	20	18(4)	0.0002	0.0695	0.0407	0.04	0.08	0.07	0.3%	0.0%	0.1%	16.45	5.40	6.44	21:10:36	07:30:17	07:58:57
Lionleft (Fig.11)	0.10	0.5%	13	18	12(4)	0.0101	0.0093	0.0651	0.09	0.03	0.09	0.6%	0.3%	0.4%	1.68	5.28	2.48	09:36:30	21:36:42	11:31:16
Average												0.7%	0.3%	0.3%	7.75	9.97	3.93	20:10:00	18:44:10	09:41:26

Table 2. Comparison vs. [Muntoni et al. 2018] and against parallel slicing. Left to right: maximal height H_{max}, precision tolerance A_{max}, number of output slices (for our method, block count in brackets), minimal and median output slice height, output precision, material waste, and simulated milling time.

	H_{max}	A_{max}	#slices				min. height				med. height				precision				waste				milling time			
			M18	M19	PS	Ours (Blocks)	M18	M19	PS	Ours	M18	M19	PS	Ours	M18	M19	PS	Ours	M18	M19	PS	Ours	M18	M19	PS	Ours
airplane (supp)	0.16	0.1%	7	4	2	3(2)	0.05	0.09	0.07	0.10	0.09	0.15	0.07	0.11	1.0%	0.1%	0.1%	0.1%	28.4	19.3	21.0	17.7	3:20:47	1:59:25	2:08:23	2:07:22
batman (supp)	0.18	0.4%	8	8	2	2(1)	0.11	0.09	0.17	0.17	0.14	0.14	0.17	0.17	0.3%	0.4%	0.3%	0.3%	4.2	3.7	3.4	3.4	6:58:31	6:41:50	4:53:30	4:54:01
bimba (Fig. 16)	0.31	1.2%	17	9	6	2(2)	0.01	0.03	0.10	0.18	0.09	0.15	0.10	0.24	1.7%	1.2%	1.1%	1.1%	11.1	9.5	9.3	4.9	19:42:53	18:16:10	18:41:43	10:13:38
bu (supp)	0.17	1.3%	12	9	7	6(2)	0.02	0.02	0.08	0.06	0.11	0.11	0.08	0.10	0.8%	1.3%	1.0%	0.8%	9.0	7.0	8.1	5.9	6:34:49	5:18:17	6:11:52	4:19:52
buddha (Fig. 5)	0.22	0.9%	9	10	17	6(5)	0.08	0.01	0.04	0.11	0.14	0.09	0.04	0.16	0.9%	0.9%	0.8%	0.7%	4.9	6.0	13.8	5.1	13:21:59	15:03:28	33:43:01	13:13:12
chinese_lion (Fig. 16)	0.23	1.8%	44	15	14	8(7)	0.00	0.01	0.05	0.07	0.05	0.08	0.05	0.15	1.7%	1.8%	1.7%	1.5%	10.5	7.3	14.8	8.9	23:23:05	17:17:23	28:42:27	19:57:43
dea (supp)	0.24	0.4%	5	5	8	4(2)	0.04	0.05	0.06	0.22	0.15	0.14	0.06	0.22	0.2%	0.4%	0.2%	0.3%	5.7	4.4	9.6	4.2	9:12:58	7:50:52	12:40:12	7:04:38
kitten (supp)	0.23	0.8%	24	17	2	2(1)	0.01	0.02	0.22	0.21	0.04	0.09	0.22	0.21	2.0%	0.8%	0.7%	0.6%	11.3	8.3	3.6	3.3	17:26:16	13:13:12	6:01:42	5:56:32
laurana (supp)	0.30	1.4%	13	7	12	3(2)	0.00	0.02	0.09	0.21	0.04	0.13	0.09	0.21	0.8%	1.3%	1.0%	0.9%	11.2	11.2	12.4	5.0	16:15:17	16:07:32	16:16:32	8:58:51
lincoln (supp)	0.19	0.7%	15	8	10	6(3)	0.00	0.03	0.07	0.12	0.04	0.14	0.07	0.16	1.6%	0.7%	0.3%	0.4%	6.7	6.0	6.1	5.5	12:33:35	10:42:55	10:32:56	9:25:54
maxplank (supp)	0.23	1.1%	6	7	7	3(1)	0.01	0.03	0.07	0.16	0.09	0.16	0.07	0.20	1.8%	1.1%	0.4%	0.5%	4.6	4.2	7.8	3.2	9:07:47	8:47:51	14:02:58	6:27:22
moai (supp)	0.20	0.5%	4	2	2	2(1)	0.01	0.18	0.19	0.19	0.16	0.19	0.19	0.19	0.4%	0.5%	0.1%	0.1%	2.4	2.0	2.0	2.0	3:47:20	3:07:32	3:23:26	3:26:19
Average															1.08%	0.87%	0.63%	0.61%	9.2	7.4	9.3	5.8	11:48:46	10:22:12	13:06:33	8:00:27

Table 3. Comparison vs. the works of Muntoni et al. [Muntoni et al. 2018, 2019] and against parallel slicing, on the data set used in [Muntoni et al. 2019]. Left to right: model, maximal height H_{max}, precision tolerance A_{max}, number of output slices (our block count in brackets), minimal and median output slice height, output precision, material waste, and simulated milling time.

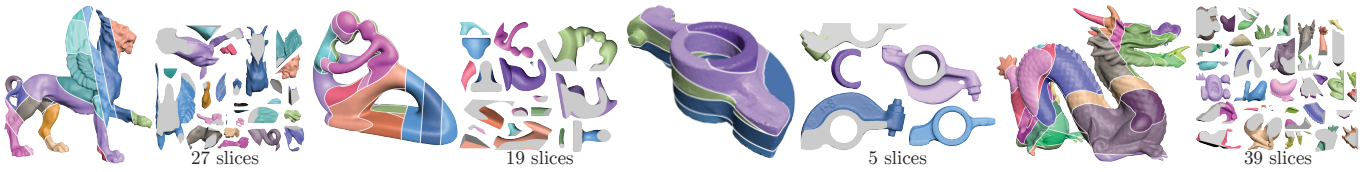


Fig. 14. Additional decomposition results. Decomposition statistics listed in Table 1. Our method works equally well on relatively smooth models such as the *rocker arm*, high genus ones such as *fertility*, as well as on highly complex models such as the *feline* or *dragon*.

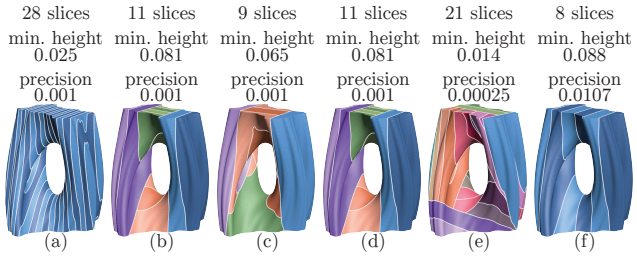


Fig. 15. Decomposing the *Julia* model with different parameters (default values used when not listed): (a) extra thin slices (H_{max} = 3%); (b) default H_{max}; (c) thick slices (H_{max} = 20%); (d) high minimal thickness ($H_{min} = H_{max}/2$), (e) tight DHF precision (A_{max} = 0.05%); (f) lax precision (A_{max} = 5%). Our method satisfies the hard constraints, increasing the slice count when necessary.

satisfy the input precision tolerance A_{max}. The statistics for the results generated using our method are provided in Table 1. We note that even though we prioritize precision over minimal slice height, the vast majority of our outputs satisfy this bound; among all examples included only the *feline* and the *dragon* have two extra thin slices each. Measurement of the median slice height confirms that over half of our slices are taller than 80% of the H_{max} bound.

Ultimately the definitive metrics for assessing decomposition methods that target fabrication are fabrication time and material waste. These metrics, however, depend on material choices and machine settings and parameters. To provide comparable measurements across the entire data set and to enable cross-method comparison, we estimate material waste as described in Appendix B. To provide comparable time measurements across all inputs and methods we measure simulated milling time, as detailed in Appendix B. We cannot directly estimate operator time for assembly using a simulator; however we note that slice count (lower is better) and slice size (larger is better) are the main factors affecting operator time. For the models we milled, we also report actual fabrication times as detailed below.

Fabrication. We fabricated six real-life replicas of the models tested (Figs 1, 13) from standardized precut slabs of material, using several different commercially available CNC milling machines. *Kitten* (4 slabs), *bimba* (5 slabs), and *bumpy* (11 slabs) were manufactured on a Roland SRM-20 from polystyrene foam blocks of 25mm thickness, with a maximum work piece size of 203.2 (X) × 152.4 (Y) *mm*. The *chair* (1 slab) and *bunny* (3 slabs) were manufactured from poplar wood slabs of 605 (X) × 285 (Y) × 25 (Z) *mm* with a ShopBot desktop. The *Julia* (1 slab) model was manufactured on a Precix CNC Router from a light-weight MDF work piece, with dimensions 1112 (X) × 989 (Y) × 18 (Z) *mm*. The decomposition parameters for all these models were selected so as to maximize output object size, subject to acceptable milling time; we set the height bound H_{max} for these models to accommodate the desired size and material slab height (see Table 1), use a tighter precision bound $A_{max} = 0.3\%$ for Julia and a more lax $A_{max} = 0.75\%$ for the bunnies, and set the size of the ignorable regions for DHF precision evaluations to $p_s = 0.0005$ (Section 8). Our results demonstrate the applicability of our method to real-life milling settings.

Fabrication Time. Fabrication time can be broken down into two components: raw milling time and operator time. These times vary with user skill (in our case, CS graduate students), and the tools and milling technology available. Our milling times using the setting described in Appendix C ranged from 5 hours for the *kitten* model to 14 hours for the *bimba*. Assembly and gluing a model took between 15 and 60 minutes. Sanding MDF and wood models to remove fixture traces took up to 3 hours with a manual sanding block. Looking back on the fabrication process, we note that sanding time could be significantly shorted by either using thinner fixtures, or by removing portions of the fixture surface to leave small tabs to hold slices in place. We opted for conservative fixtures due to limited mill time. Fixture removal times would also have been decreased with professional tooling; our fabrication results were generated by graduate students with no previous fabrication or woodworking experience, and little access to professional woodworking tools.

Comparison to Prior Art. We focus our comparisons on three prior methods: slicing using evenly-spaced parallel cut-through planes [Cirtes 1991], the automatic HF decomposition method of [Muntoni et al. 2018], and the semi-manual HF decomposition approach of [Muntoni et al. 2019].

There are no available implementations of the industrial methods for parallel slicing, such as [Cirtes 1991]). We compare to these approaches by evenly slicing models using the shortest axis of their object-aligned bounding box, and using the minimal slice count necessary to satisfy the required DHF precision (Figures 5, 17, Tables 3, 2). For simple inputs, such as the *guaje* (Figure 7) the number of slices this method produces is comparable to ours, and for a couple of models (*chair*, *julia*) it produces fewer slices. For the vast majority of inputs, however, we produce significantly fewer slices. In particular, on more complex models such as *bimba* (Figure 13), *david*, or *gargoyle*, we produce about half the number of slices. We also note that the slices obtained using the industry-employed method are consistently significantly thinner than the maximal thickness threshold we use. While our median slice thickness is on average about 80% of the maximal height bound, the median

	Relative Max			Relative Mean			milling time		
	1/4"	1/8"	1/16"	1/4"	1/8"	1/16"	1/4"	1/8"	1/16"
bimba	0.016	0.016	0.016	0.002	0.001	0.001	3:11:38	3:54:41	4:01:38
buddha	0.033	0.033	0.033	0.002	0.002	0.002	6:55:45	7:49:28	7:58:28
bumpy	0.019	0.019	0.019	0.001	0.001	0.001	13:19:16	15:25:30	15:43:17
bunnies	0.005	0.003	0.003	0.001	0.001	0.001	10:56:32	10:17:04	10:41:26
kitten	0.021	0.02	0.021	0.001	0.001	0.001	4:54:00	5:42:26	5:47:10
lion	0.016	0.016	0.016	0.001	0.001	0.001	11:09:46	11:31:02	11:52:56

Table 4. Comparison between different milling tip sizes. Left to right: model, maximum Hausdorff distance as a percentage of bounding box diagonal, mean Hausdorff distance as a percentage of bounding box diagonal, and simulated milling times.

for this method is about 50%. Given that in real-life settings this threshold is motivated by material thickness, milling such thin slices would result in a tremendous waste of material. Our measurements confirm that on average our method is 43% faster and wasted 49% less material.

We run the method of [Muntoni et al. 2018] using the same slice height constraint H_{max} as our method. We adjust their angle based HF precision threshold to achieve comparable precision to ours when using our metric. Our method produces on average three times fewer slices than [Muntoni et al. 2018]. In some cases the ratio is even more drastic: for instance, [Muntoni et al. 2018] requires 27 slices to decompose the *kitten* (Figure 13) model, which we accomplish using just four. Using our method leads to a 43% reduction in material waste and a 42% reduction in milling time on average compared to this method, representing significant savings over closest automatic prior art.

Since the outputs of the interactive method of [Muntoni et al. 2019] depend on user choices, we use their existing outputs for comparison. We consequently use their results to determine the maximal slice thickness H_{max} and the accuracy bound A_{max} parameters to execute our method with. We first measure the shortest side of each slice in the Muntoni et al. [2019] results, and use the maximum of these as H_{max}; consequently, our slices are at most as thick as the slices we compare to. Similarly, we measured the precision of their results and used the maximal value obtained per model as our precision bound A_{max} for that model. In all but one case, our method produced fewer slices than [Muntoni et al. 2018]; it produced the same minimal number (2 slices) on the *moai*, an object which is already a DHF, on which only the height bound prevents us from creating a single slice. We waste 19% less material and require 18% less time to mill our slices on average compared to this method on this data. These numbers indicate that our framework performs better than human experts operating a semi-manual HF decomposition framework. Full statistics for all these experiments are provided in Table 3.

Additional Ablation Studies. We compare our results to several simple baselines throughout the paper. The importance of using the LDHF criterion rather than the DHF one for block decomposition is demonstrated in Fig. 8; as shown, our approach drastically reduces the output slice count and increases the slice height. Fig. 9 showcases the difference between greedy block computation and our prediction-based approach. Finally, the importance of our size optimization and slice positioning is shown by Figs. 7 and 10.

Tip Size. We measured the impact of tip size on the accuracy and milling time of DHFSlicer outputs by using the Autodesk Fusion360

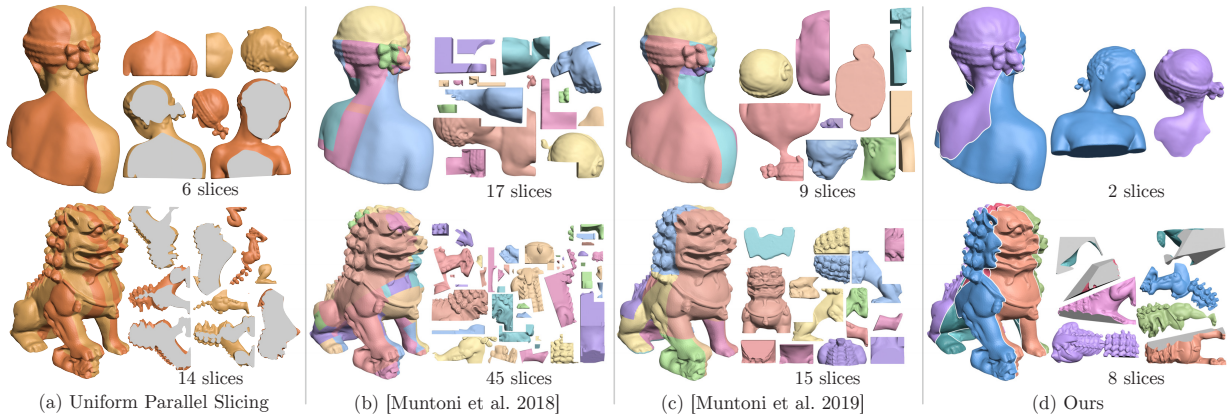


Fig. 16. Our framework (d) produces fewer slices, and taller slices, than parallel slicing (a), and the methods of [Muntoni et al. 2018] (b) and [Muntoni et al. 2019] (c).

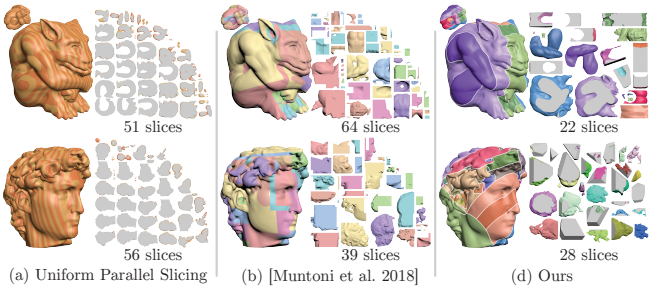


Fig. 17. Our framework (right) produces fewer and more evenly sized slices than parallel slicing (left) and [Muntoni et al. 2018] (center).

rockerarm								
p_s	#blocks	#slices	#dir	#blocks	#slices	#samples	#blocks	#slices
0.0125%	3	5	25	3	7	3	2	4
0.025%	3	5	50	3	5	5	3	5
0.05%	2	4	75	2	5	10	3	6
0.075%	2	4	100	3	7	20	3	6
gargoyle								
p_s	#blocks	#slices	#dir	#blocks	#slices	#samples	#blocks	#slices
0.0125%	12	29	25	8	23	3	8	22
0.025%	7	22	50	7	22	5	7	22
0.05%	8	22	75	8	24	10	7	22
0.075%	8	21	100	8	22	20	8	20

Table 5. Parameter sensitivity: halving, doubling, and even quadrupling our discrete parameters has limited impact on the results. The only exception is decrease in ignored area for the Gargoyle mesh that contains collapsed triangles, our default threshold value p_s is set to avoid this sensitivity (see discussion in text).

simulator with 1/4", 1/8", and 1/16" finishing tips to generate simulated output slices. We then computed the maximum Hausdorff distances between the input surface and the simulated output slices. As expected, as tip size increases, the simulator predicted milling time decreases while the Hausdorff distance increases. Still, all measured Hausdorff distances were below the prescribed precision tolerances A_{max}; see Table 4 for more details.

Stress Tests. We tested our method on several highly challenging inputs including the *feline* (27 slices) and *dragon* (39 slices) models. These objects contain fine features and have high genus. Our method successfully partitioned these inputs into slices that satisfy the DHF precision tolerance, forming largely well-sized slices whose median height is 60% of H_{max}. Both outputs satisfy our precision threshold. On both models, only two slices do not satisfy our soft minimal height threshold; as Section 4 discusses, a solution satisfying this threshold may not always exist.

Runtimes. Total runtime for our method is, on average, around six minutes (ranging from 0.5 minutes for *kitten* to 9 minutes for *julia*. This time does not include the remeshing step performed prior to registration pattern creation - the external library we use [Hu et al. 2018] takes about five minutes to remesh our inputs. Timings were measured on a Intel Core I7-9700k 3.60GHz with 32GB of system memory running Windows 10. We note that the total time to fabricate a model is heavily dominated by milling time which, as reported above, takes hours. Thus our computation cost is negligible compared to the milling and assembly times.

Parameter Sensitivity. We experimented with doubling, quadrupling, and halving the number of possible milling directions, the number of samples chosen for ray tracing, and the size of the ignorable regions for precision evaluations p_s, as described in Section 8, for two representative inputs: the *rockerarm* and the *gargoyle*. The vast majority of changes led to minimal change in slice count. The only exception is the halving the ignore region area p_s on the high-detail *gargoyle* model, where reducing these regions requires the decomposition to respect minuscule details (see inset in Sec. 8).

Limitations. While our framework successfully reduces the prevalence of extra thin slices, one could potentially reduce their prevalence further. We observe that such slices are most likely to be formed on inputs with prominent narrow concavities, where the distance between opposite sides of the concavities is below the maximal slice height (e.g. lion's mouth and chin in Figure 18). The opposite sides of such concavities act as one another's occluders across multiple axis choices, preventing formation of reasonably

sized LDHF blocks. Using our default framework the resulting partitions on such inputs can contain many undesirably small slices (Figure 18a). One possible approach for addressing such scenarios, once detected, is to cut the models using planes that separate the opposite sides of such concavities. We experimented with triggering such *exposure* cuts during LDHF decomposition when the height of *all* LDHF strips drops below $\frac{1}{2}H_{max}$. We position an exposure plane inside the deep concavity by instantiating it at the centers of regions not covered by any LDHF strips (yellow in Figure 18c). This placement is designed to maximally reduce the percentage of the surface area not covered by such strips. We then perform an exhaustive search over a finite set of possible plane orientation, using the same set of directions as the one used for our axis selection (Section 8). We execute each cut and compute new LDHF strips, then we finally select the cut that maximizes the area covered by these strips. While this exhaustive search is computationally expensive, it can improve the performance on such pathological inputs.

Our method is not guaranteed to obtain, and is not geared toward obtaining, the *most* compact decomposition; doing so requires solving a NP-hard problem, whose complexity is a function of the size of the input mesh. It does, as demonstrated above, produce decompositions that are more compact than those produced using any existing alternatives. While on average we produce significantly better results than previous work, we note that parallel slicing produces fewer slices than our method on the chair and Julia models. One consequence of our decision to maximize slice height rather than volume is that models may exhibit slices that look small from a front view. Our method's accuracy is limited by the resolution of the input mesh, since our discrete computations operate on triangles. Refining the input meshes to a sufficient resolution would eliminate this constraint. Furthermore, while we account for the main factors that impact fabrication and assembly, users in a practical scenario may want to consider other factors such as physical material properties, milling path efficiency, or milling tip size. These considerations, while important, are outside the scope of our work. Our method does not explicitly optimize for dihedral angles between planar slice sides, which may introduce problems during milling if pieces terminate at an infinitely thin edge. In practice, the minimum dihedral angle across all slices produced by our method is 13 degrees; the median minimal angle across models is 32 degrees, and we encountered no problems during fabrication. Finally, while most of our cuts are orthogonal to the slice milling direction and are not affected by aliasing, aliasing may be an issue for cuts separating LDHF blocks when a larger milling tip is used. Our computations assume a zero-radius milling tip; in future work it would be interesting to consider mesh decompositions that take advantage of multiple milling tip sizes to enable faster milling speeds on areas of the mesh where increased precision is not necessary.

10 CONCLUSIONS

We propose a new, robust, end-to-end computational milling pipeline that allows high-fidelity fabrication of complex 3D shapes from off-the-shelf precut slabs of material whose height is significantly lower than the desired object dimensions. We enable this pipeline by formulating and efficiently solving a new volumetric decomposition problem: partitioning a volume into a compact set of assemblable, bounded-height, well-sized DHF slices. Our key technical insight is that DHF slice decomposition can be recast as computing a compact coarse LDHF decomposition, then slicing LDHF blocks into a small number of well-sized slices. We efficiently compute locally near-optimal solutions to both problems by casting both as classical optimization problems. As demonstrated, our method outperforms existing alternatives and is validated using both virtual and real examples.

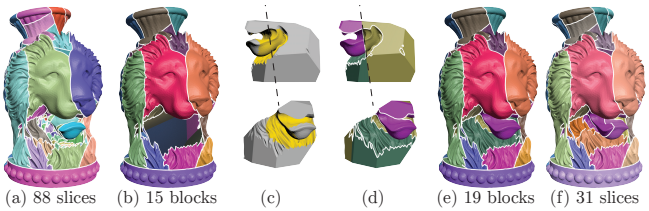


Fig. 18. (a) Default decomposition of the Lion vase contains 88 slices including numerous tiny ones around the lion's mouth (b-e) Partition using exposure cuts: After generating 15 blocks (b) , the remaining connected component of the Lion vase model (c) does not allow for well-sized LDHF strips (the yellow area is not part of any LDHF strip). Using the best exposure cut (dashed line), produces four fully LDHF blocks (d), completing the block decomposition (e). The final output (f) has 31 slices.

Our work introduces several interesting follow-up questions. While our method produces compact decompositions, the slice count we obtain can potentially be further reduced (e.g. the chair in Figure 13). Fixtures play a critical role in enabling efficient milling and subsequent assembly; while necessary during milling, removing fixtures after milling requires extra user time. Designing fixtures that are as effective at milling time as our surface based solution while simplifying their removal is an important practical question.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. We are also deeply grateful to: Luciano S. Burla for help with images creation; Jerry Yin and Yuan Yao for help with running the results. The UBC authors are supported by NSERC.

REFERENCES

G. Alemanno, P. Cignoni, N. Pietroni, F. Ponchio, and R. Scopigno. 2014. Interlocking pieces for printing tangible cultural heritage replicas. In *Eurographics Workshop on Graphics and Cultural Heritage*. 145–154.

C. Araújo, D. Cabiddu, M. Attene, M. Livesu, N. Vining, and Alla Sheffer. 2019. Surface2Volume: Surface Segmentation Conforming Assemblable Volumetric Partition. *ACM Trans. Graph.* 38, 4 (2019).

S. Asafi, A. Goren, and D. Cohen-Or. 2013. Weak Convex Decomposition by Lines-of-sight. In *Proc. Eurographics SGP 2013.*

M. Attene. 2015. Shapes in a box: Disassembling 3D objects for efficient packing and fabrication. *Computer Graphics Forum* 34, 8 (2015).

Autodesk. 2020. Fusion 360. https://www.autodesk.com/products/fusion-360/.

O.J. Bakker, T.N. Papastathis, A.A. Popov, and S.M. Ratchev. 2013. Active fixturing: literature review and future research directions. *International Journal of Production Research* 51, 11 (2013), 3171–3190.

Z. M. Bi and W. J. Zhang. 2001. Flexible fixture design and automation: Review, issues and future directions. *International Journal of Production Research* 39, 13 (2001).

B. Bickel, P. Cignoni, L. Malomo, and N. Pietroni. 2018. State of the Art on Stylized Fabrication. *Computer Graphics Forum* 37, 6 (2018), 325–342.

V. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE. Trans. Pattern Analysis and Machine Intelligence* 23, 11 (2001), 1222–1239.

Y. Boykov and V. Kolmogorov. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Analysis and Machine Intelligence* 26, 9 (2004), 1124–1137.
B. Chazelle. 1984. Convex Partitions of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm. *SIAM J. Comput.* 13, 3 (1984).
X. Chen, H. Zhang, J. Lin, R. Hu, L. Lu, Q-X. Huang, B. Benes, D. Cohen-Or, and B. Chen. 2015. Dapper: Decompose-and-Pack for 3D printing. *ACM Trans. Graphics* 34, 6 (2015).
Cirtes. 1991. Cirtes: Recherche et Developpement. https://cirtes.com/en/stratoconception/.
R. Cook. 1984. Shade Trees. *ACM Siggraph Computer Graphics* 18, 3 (1984).
T. Cormen, C. Leiserson, R. Rivest, and C. Stein. 2001. *Introduction to Algorithms* (2nd ed.). MIT Press.
B. Delebecque, Y. Houtmann, G. Lauvaux, and C. Barlier. 2008. Automated generation of assembly features in layered manufacturing. *Rapid Prototyping Journal* (2008).
M. Doggett and J. Hirche. 2000. Adaptive view dependent tessellation of displacement maps. In *Proc. Graphics hardware*. ACM.
H. Edelsbrunner and J. Harer. 2008. Persistent homology -— a survey. In *Surveys on discrete and computational geometry*, Vol. 453.
S. Fekete and J. Mitchell. 2001. Terrain decomposition and layered manufacturing. *International Journal of Computational Geometry & Applications* 11, 06 (2001).
W. Gao, Y. Zhang, D. Nazzetta, K. Ramani, and R. Cipra. 2015. RevoMaker: Enabling multi-directional and functionally-embedded 3D printing using a rotational cuboidal platform. In *Proc. User Interface Software & Technology (UIST)*. ACM.
M. Garey, D. Johnson, and R. Sethi. 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research* 1, 2 (1976), 117–129.
J. Hao, L. Fang, and R. Williams. 2011. An efficient curvature-based partitioning of large-scale STL models. *Rapid Prototyping Journal* 17, 2 (2011).
P. Herholz, W. Matusik, and M. Alexa. 2015. Approximating Free-form Geometry with Height Fields for Manufacturing. *Computer Graphics Forum* 34, 2 (2015).
K. Hildebrand, B. Bickel, and M. Alexa. 2013. Orthogonal slicing for additive manufacturing. *Computers & Graphics* 37, 6 (2013).
Y. Houtmann, B. Delebecque, G. Lauvaux, C. Barlier, and G. Ris. 2007. Decomposition applied to layered manufacturing using critical points. *Virtual and Physical Prototyping* 2, 2 (2007), 127–133.
R. Hu, H. Li, H. Zhang, and D. Cohen-Or. 2014. Approximate Pyramidal Shape Decomposition. *ACM Trans. Graph.* 33, 6, Article 213 (2014).
Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4 (2018), 60:1–60:14.
A. Jacobson, D. Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.
C.-S. Jun, D.-S. Kim, J.S. Hwang, and T.-C Chang. 1998. Surface slicing algorithm for rapid prototyping and machining. *Geometric Modeling Processing* (1998).
B. Keinert, M. Innmann, M. Sänger, and M. Stamminger. 2015. Spherical Fibonacci Mapping. *ACM Trans. Graph.* 34, 6 (2015).
Y. Koyama, S. Sueda, E. Steinhardt, T. Igarashi, A. Shamir, and W. Matusik. 2015. AutoConnect: Computational Design of 3D-Printable Connectors. 34, 6 (2015).
V Kraevoy, D Julius, and A Sheffer. 2007. Shuffler: Modeling with interchangeable parts. *The Visual Computer* (2007).
B. Lévy, S. Petitjean, N. Ray, and J. Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.* 21, 3 (2002), 362–371.
J-M. Lien and N. Amato. 2007. Approximate convex decomposition of polyhedra. In *Proc. ACM Symposium on Solid and physical modeling*.
M. Livesu, S. Ellero, J. Martínez, S. Lefebvre, and M. Attene. 2017. From 3D models to 3D prints: an overview of the processing pipeline. *Computer Graphics Forum* 36, 2 (2017).
L. Luo, I. Baran, S. Rusinkiewicz, and W. Matusik. 2012. Chopper: Partitioning Models into 3D-printable Parts. *ACM Trans. Graph.* 31, 6 (2012).
A. Mahdavi-Amiri, F. Yu, H. Zhao, A. Schulz, and H. Zhang. 2020. VDAC: Volume Decompose-and-Carve for Subtractive Manufacturing. In *Proc. SIGGRAPH Asia 2020*. To appear.
A. Medeiros e Sá, K. Rodriguez Echavarria, N. Pietroni, and P. Cignoni. 2016. State Of The Art on Functional Fabrication. In *Proc. Workshop on Graphics for Digital Fabrication*.
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).
A. Muntoni, M. Livesu, R. Scateni, A. Sheffer, and D. Panozzo. 2018. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. *ACM Trans. Graph.* 37, 5 (2018), 169:1–169:15.
A. Muntoni, L.D. Spano, and R. Scateni. 2019. Split and Mill: User Assisted Height-field Block Decomposition for Fabrication. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*.
C. Rattat. 2017. *CNC Milling for Makers: Basics - Techniques - Applications*. Rocky Nook.
R. Schmidt and K. Singh. 2010. Meshmixer: An Interface for Rapid Mesh Composition. In *ACM SIGGRAPH 2010 Talks*. Article 6.
P. Song, B. Deng, Z. Wang, Z. Dong, W. Li, C-W. Fu, and L. Liu. 2016. CofiFab: Coarse-to-Fine Fabrication of Large 3D Objects. *ACM Trans. Graph (SIGGRAPH 2016)* 35, 4 (2016).
P. Song, Z. Fu, L. Liu, and C-W. Fu. 2015. Printing 3D objects with interlocking parts. *Computer Aided Geometric Design* 35 (2015), 137–148.
J. C. Trappey and C. R. Liu. 1990. A literature survey of fixturedesign automation. *The International Journal of Advanced Manufacturing Technology* 5, 3 (01 Aug 1990).
J. Tyberg and J.H. Bohn. 1998. Local Adaptive Slicing. *Rapid Prototyping Journal* 4, 3 (1998).
J. Yao, D. Kaufman, Y. Gingold, and M. Agrawala. 2017. Interactive Design and Stability Analysis of Decorative Joinery for Furniture. *ACM Trans. Graphics* 36, 2 (2017), 20.
M. Yao, Z. Chen, L. Luo, R. Wang, and H. Wang. 2015. Level-set-based Partitioning and Packing Optimization of a Printable Model. *ACM Trans. Graph.* 34, 6 (2015), 214:1–214:11.
Q. Zhou, E. Grinspun, D. Zorin, and A. Jacobson. 2016. Mesh Arrangements for Solid Geometry. *ACM Trans. Graph.* 35, 4, Article 39 (July 2016).

Appendix A MATHEMATICAL DERIVATIONS

Cutting Plane Orientation for DHF Shape Volume Slicing. We observed in Section 3.2 that when slicing a DHF shape into slices satisfying the maximal height constraint H_{max}, for most inputs the slice count is minimized when the cutting planes are orthogonal to the DHF axis. To show this, consider two slices separated by a non-axis aligned plane. Since each slice is at most H_{max} high, the distance between the plane's maximum and minimum points along the axis is below H_{max}. For this solution to be better than an orthogonal one, the region between the plane's max/min points must contain at least one local maximum and one local minimum of the height function; since H_{max} is fairly small, this is a rare occurrence. If the interval contains no extrema, an orthogonal plane at the minima or maxima will produce the same slice count as the slanted one; if it contains only maxima (resp. minima), one can place the plane at the location of the highest maximum (lowest minimum), without increasing the height of either slice.

Derivation of Frequency Scale for Registration Patterns. We derive the value $k = \tan(15^\circ)$ for the frequency of the registration patterns as follows. After applying the registration pattern, the deformed surface must remain DHF with respect to the milling direction (without loss of generality, assumed to be the axis Z).

Recall that our deformation function is $F(x, y) = k \cdot H_r \cdot \sin(x/H_r))$. The gradient of F is $\nabla F = k\cos(x/H_r)$ and is therefore bounded by k. In order for the deformed surface to remain DHF, since ∇F is bounded by k, it suffices that F is a height field for any direction that forms an angle with respect to Z less than $90^\circ - \arctan(k)$. To understand this intuitively, consider the extrema cases: if F is flat ($k = 0$), then ∇F is zero everywhere and is trivially a height field for any direction ($\arctan(0) = 0$). Conversely, if F is very steep, ∇F approaches k, and consequently F is only a height field for those directions whose angle with Z is close to 0.

Appendix B MEASUREMENTS

The exact amount of wasted material for each specific milling task depends on factors such as packing method, the length and width of the slabs, and machine and material properties (which dictate the amount of intra-slice spacing required to facilitate machining access). We use a metric that is agnostic, by design, to the first two factors, and which accounts for the intra-slice spacing necessary for milling success.

We compute packing-algorithm independent material waste, while accommodating the need to space slices for milling, as follows. We

project all slices to the plane orthogonal to the milling axis, and compute, for each slice, a tight bounding box. To account for the need to space slices during milling, we extend each box along both dimensions by 15% of the object's bounding box diagonal. We then measure the wasted material by multiplying the area of the bounding rectangle of the pack by H_{max}, and measuring the difference between this volume and the slice volume. We sum these values across all slices to obtain the waste per model.

To provide comparable time measurements across all inputs and methods, we pack the slices produced for each input using the same packing method (assuming infinite milling bed dimensions), spacing them at 15% of the bounding box diagonal away from one another, and setting the fixture offset to 10% of this diagonal. We then measure simulated milling time using the milling simulator in Autodesk Fusion 360 [Autodesk 2020], using the same settings (number of passes, milling tips assignments) for all inputs: all timings were estimated using two passes (pocket clearing with a 1/8" flat endmill, followed by scalloping with a 1/8" round endmill) per side.

Appendix C FABRICATION DETAILS

We used the following processes for milling the fabricated models. *Kitten*: Adaptive clearing on both sides with a 1/8" flat end mill followed by fine step-down of 0.2mm. *Bimba*: Adaptive clearing on both sides with a 1/8" flat end mill followed by fine step-down of 0.3mm. *Airplane*: Adaptive clearing on both sides with a 1/8" flat end mill followed by fine step-down of 0.4mm. *Bumpy:* 2 passes per side (adaptive then scalloping) adaptive clearing using a 1/4" flat end mill and a fine step-down of 0.5mm, and scalloping using a 1/4" ball end mill. Finer details were done with a 1/8" ball end mill. *Chair*: 2 passes per side (pocketing then parallel), both with a 1/4" flat end mill. *Bunny*: 2 passes per side, pocketing using a 1/4" flat end mill, and scalloping using a 1/4" ball end mill.