

Volume-Aware Design of Composite Molds

THOMAS ALDERIGHI, Università di Pisa and ISTI-CNR, Italy

LUIGI MALOMO, ISTI-CNR, Italy

DANIELA GIORGI, ISTI-CNR, Italy

BERND BICKEL, IST, Austria

PAOLO CIGNONI, ISTI-CNR, Italy

NICO PIETRONI, University of Technology Sidney, Australia and ISTI-CNR, Italy

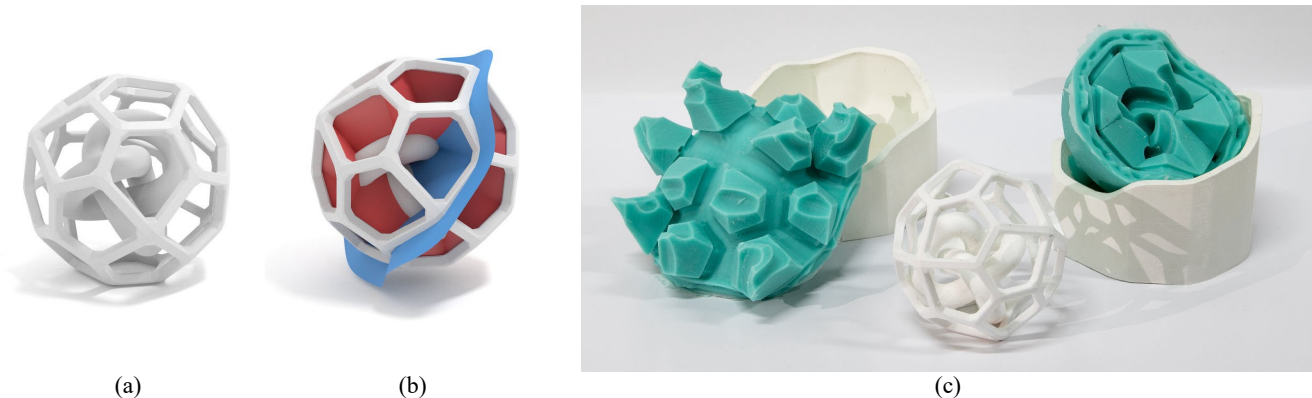


Fig. 1. Our technique for casting extremely complex shapes up to (a) objects made up of multiple entangled pieces. (b) We design a cut layout in the mold volume, which is defined by a parting surface membrane (blue) and a set of additional membranes to enforce moldability (red). (c) We assemble a composite mold made up of a hard plastic shell and a soft silicone part, and then fabricate the given object using liquid casting techniques.

We propose a novel technique for the automatic design of molds to cast highly complex shapes. The technique generates composite, two-piece molds. Each mold piece is made up of a hard plastic shell and a flexible silicone part. Thanks to the thin, soft, and smartly shaped silicone part, which is kept in place by a hard plastic shell, we can cast objects of unprecedented complexity. An innovative algorithm based on a volumetric analysis defines the layout of the internal cuts in the silicone mold part. Our approach can robustly handle thin protruding features and intertwined topologies that have caused previous methods to fail. We compare our results with state of the art techniques, and we demonstrate the casting of shapes with extremely complex geometry.

CCS Concepts: · **Computing methodologies** → *Shape modeling*.

Additional Key Words and Phrases: fabrication, mold design, casting

ACM Reference Format:

Thomas Alderighi, Luigi Malomo, Daniela Giorgi, Bernd Bickel, Paolo Cignoni, and Nico Pietroni. 2019. Volume-Aware Design of Composite Molds. *ACM Trans. Graph.* 38, 4, Article 110 (July 2019), 12 pages. <https://doi.org/10.1145/3306346.3322981>

Authors' addresses: Thomas Alderighi, Università di Pisa, ISTI-CNR, Italy; Luigi Malomo, ISTI-CNR, Italy; Daniela Giorgi, ISTI-CNR, Italy; Bernd Bickel, IST, Austria; Paolo Cignoni, ISTI-CNR, Italy; Nico Pietroni, University of Technology Sidney, Australia, ISTI-CNR, Italy.

1 INTRODUCTION

Casting is a well-established manufacturing technique with many appealing properties: a wide range of materials, such as concrete, plaster, plastic resins, or edibles (e.g., chocolate) can be used, and re-usable molds reduce production costs, while allowing for the efficient reproduction of multiple copies. However, the geometric complexity of shapes that are fabricable by casting is much more limited in comparison to 3D printing, and a significant amount of effort has to be invested to design moldable shapes and the mold itself.

Recently, a new generation of computational design algorithms have been introduced to support the creation of re-usable molds [Alderighi et al. 2018; Herholz et al. 2015; Malomo et al. 2016; Nakashima et al. 2018]. These approaches opened up new avenues for reproducing shapes, by simplifying the production process and making the design workflow accessible to non-experts. This has been an essential step towards democratizing molding; however, reproducing complex free-form geometry by casting remains challenging and is constrained by the available computational design tools.

In this paper, we propose a novel method for the automatic design of composite molds for objects of complex geometry and topology (Figure 1.a). Our molds are made up of an elastic silicone part surrounded by a hard plastic shell (Figure 1.c). In art reproduction and prop-building, the use of composite molds is often referred to as

mother-and-glove molding or *brush-up* molding is a common practice that allows a skilled artisan to reproduce complex objects [Hart 2013]. The composite nature of molds allows for an easy extraction, even in the presence of undercuts, while avoiding undesired deformations during casting. Nevertheless, this kind of molding approach has never been faced from a computational perspective.

The key problem in the computational design of a composite mold is deciding how the mold should open up, that is, how to place cuts in the mold volume to allow for cast extraction. To address this problem, we propose a novel approach based on an analysis of the *escape paths* of mold volume elements. Intuitively, the escape path of a small mold volume element is the shortest path that this portion of the mold would follow when leaving the cast shape.

The behavior of these escape paths can provide information about whether neighboring elements should belong to different mold pieces and whether they should be separated by additional cuts in the silicone part to ease the extraction. We use geodesic shortest paths to the mold volume boundary as geometric proxies for the escape paths. Our cut configurations are modeled as a set of smooth 3D surfaces, which we call membranes, that can intersect with each other to form a (possibly) non-manifold geometry. In Figure 1.b, the blue membranes separate the elements belonging to different mold pieces, while the red membranes correspond to additional cuts.

In contrast to previous works, our formulation does not require solving a complex integer programming problem [Alderighi et al. 2018] or heuristically testing a large number of cut layouts [Malomo et al. 2016]. It allows for elegantly, efficiently, and directly discerning the mold geometry within the volume. We also describe a novel technique to find the optimal liquid pouring directions that minimize the presence of artifacts in the cast object.

We show how our workflow enables the automatic construction of re-usable composite molds and the accurate reproduction of free-form geometries with a complexity significantly beyond what previous techniques can achieve.

2 RELATED WORK

Computational tools for shape processing have significantly contributed to novel, rapidly evolving digital manufacturing workflows [Bermano et al. 2017; Bickel et al. 2018; Liu et al. 2014; Sá et al. 2016; Umetani et al. 2015]. In this section, we place our contribution in the context of automatic mold design.

Molding is a widely employed process for efficiently producing replicas of a shape at a relatively low cost. Many everyday goods and commodities, such as packagings, plastic furniture, specific foods, and jewelry, are made by shaping materials using molds. There is a wide range of molding techniques, for example, spin casting, injection molding, thermoforming [Schüller et al. 2016], or blow molding, which can be used with a variety of materials.

When used in the context of mass or small-series production, a common challenge is designing a mold that allows for the extraction of the fabricated object without destroying the mold, while making it efficient to open, close, and fill. In practice, designing a high-quality mold is an art that requires years of training and expertise [De Garmo et al. 2011; Wannarumon 2011]. Several computational tools have been developed to support the design process.

For simple objects, it is easy to identify the optimal parting surfaces and create molds made of two pieces [Chakraborty and Reddy 2009; Zhang et al. 2010], but complex objects may require being split into several simpler parts that are separately cast and reassembled afterward [Lin and Quang 2014], or they may require multi-piece molds that can be disassembled to release the cast object.

Herholz et al. [2015] introduced the problem to the computer graphics community by proposing an optimization method to automatically define parting surfaces for rigid molds. An input surface is segmented into multiple patches that approximate the surface as a set of height fields. When the input surface cannot conform to a height field, the surface is deformed trying to minimize visual distortions. Restricting the mold complexity to two rigid pieces, Nakashima et al. [2018] proposed an interactive technique to decompose the shell of an object into moldable parts. However, with increasing geometric complexity, the number of rigid molds or cast pieces grows as well, making these methods impractical for complex objects.

To overcome the limitations of rigid molding, a new generation of techniques have allowed for the creation of flexible molds to ease the extraction process. These techniques have significantly increased the range of reproducible shapes [Alderighi et al. 2018; Malomo et al. 2016]. Malomo et al. [2016] proposed *FlexMolds*, a single-piece, thin, and flexible mold whose cut design is driven by a physically-based simulation of the extraction process. FlexMolds are made of a thin layer of TPU, a flexible plastic material, and fabricated using 3D laser sintering. Although FlexMolds can theoretically handle very complex shapes, they still have some drawbacks. First, because the cuts are manually sealed with silicone, there is a high risk of cast material leakage. Moreover, when fabricating large objects, the thin layer of material is prone to becoming deformed under the casting pressure, resulting in possibly large deformations in the replicas. In addition, so far the fabrication of FlexMolds has only been demonstrated with laser sintering, as the removal of an internal support structure may be problematic.

To overcome these limitations, Alderighi et al. [2018] proposed *Metamolds*, a method to create silicone molds. Silicone is a common material in casting and offers excellent flexibility and high accuracy in the reproduction of surface details. Alderighi et al. [2018] generated multi-piece silicone molds by casting liquid silicone into custom 3D printed containers called metamolds. The mold parting surfaces were defined through a segmentation technique based on surface visibility coupled with local surface descriptors. This approach might fail to capture and segment elongated thin features. Moreover, Metamolds only placed additional cuts in the volume corresponding to tunnel holes in the object surface. Not only would this approach fail for knotted objects, it would also miss geometric cuts that are not related to holes. Therefore, whereas Metamolds delivered good results for moderately complex objects, their application would still be limited to a certain set of shapes. On the contrary, we base our approach on a *volumetric* analysis, which places cuts to relieve the internal tension from the mold, even in the presence of thin features and intertwined topologies.

The composite nature of our mold is a key advantage over both Flexmolds and Metamolds. Because the silicone part is thinner than

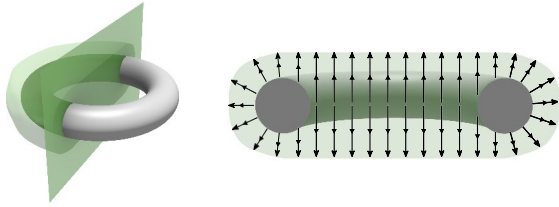


Fig. 2. A schema illustrating the concept behind the escape paths. Left: a torus, surrounded by the volume of its mold (light green), and a cutting plane. Right: the escape paths on the section of the mold volume cut by the plane: for each point in the volume, the escape path is the walk with minimal length toward the exterior of the volume.

for Metamolds, it offers less resistance during extraction. Furthermore, because the hard shell keeps the silicone part in the correct position during the casting operation, we nullify the deformations because of casting material pressure, which is something that affected Flexmolds.

Finally, we note how our definition of membranes that insert cuts in the silicone is related to the notion of the medial skeleton of the mold volume [Tagliasacchi et al. 2016]. The medial axis transform associates to a shape the set of locations having more than one corresponding closest point on the shape boundary. Our membranes are located where the geodesic paths to the boundary of the mold volume diverge.

3 OVERVIEW

We introduce a novel technique to automatically design molds for casting highly complex shapes. The technique generates composite, two-piece molds.

Probably the most critical part of the entire casting process is removing the mold from the cast object. To design how the mold should open up, we tessellate the volume surrounding the object, and analyze the behavior of the *escape paths* of volumetric elements. Escape paths are idealized paths that each element could follow while removing the mold from the model (Figure 2).

We exploit the analysis of the escape paths in the mold volume for the following two tasks:

- The definition of a *parting surface* that cuts the volume in two halves and thereby defines the two mold pieces (Figure 3.a, blue). If we partition the exterior boundary into two parts corresponding to two extraction directions, then the parting surface should separate the neighboring elements that have escape paths that end at different parts of the exterior boundary.
- The definition of *additional membranes* around object features that allow for an easier and safer removal of the silicone mold (Figure 3.b, red). The rationale is that a cutting membrane should separate the neighboring volume elements whose escape paths pass on different sides of a feature in the cast object.

Similar to [Alderighi et al. 2018], we fabricate the molds by pouring silicone into a 3D printed shape. We achieve the cuts in the mold by placing thin membranes in the 3D printed shape.

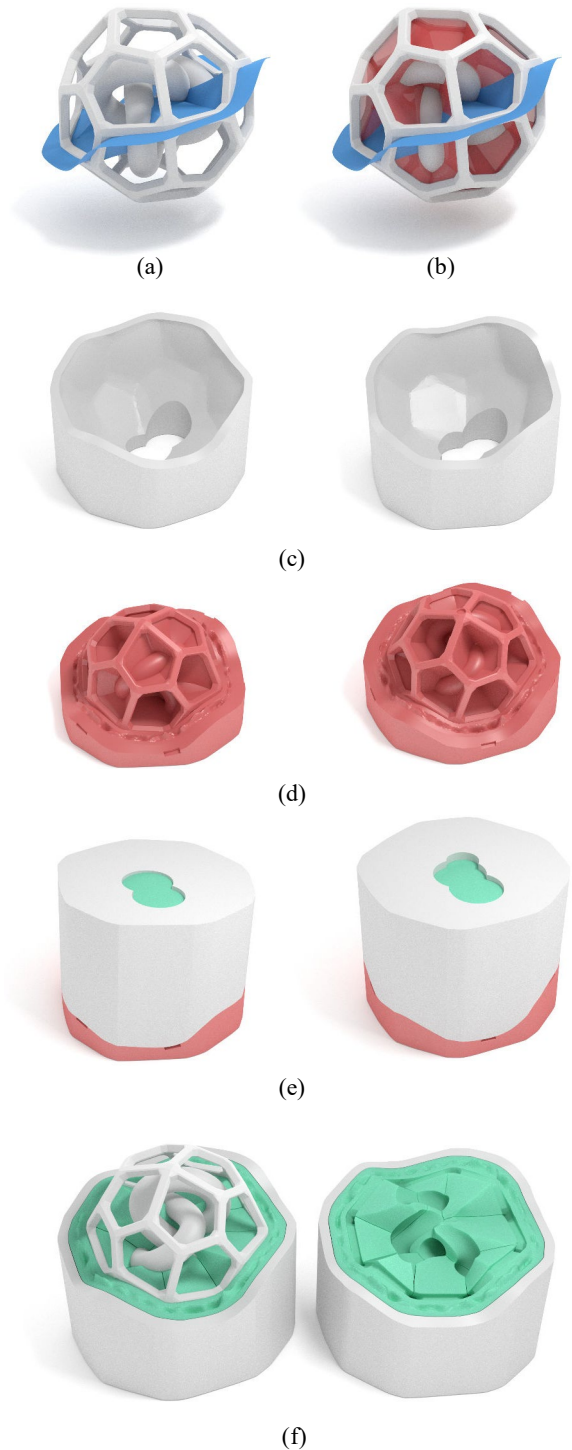


Fig. 3. Our fabrication pipeline. The parting surface (a, blue) and the additional membranes (b, red) define the cut layout; the two pieces of the hard plastic shell (c); the two metamolds whose geometry incorporates the cut layout (d); the closed container obtained by assembling the hard plastic shells and their corresponding metamolds, where we pour silicone to cast the flexible mold (e); the full composite mold and the cast object inside (f).

As geometric proxies for the escape paths, we take the shortest paths from the interior points to the external boundary of the volume, according to an object-aware metric defined in the volume (Section 4.5). The soundness of a purely geometric approach to evaluate moldability, as opposed to costly and complex physical simulations, was demonstrated in [Alderighi et al. 2018]. However, while they generated multi-piece molds by analyzing local visibility and surface topology, we define a *volume-aware* algorithm to generate cut layouts that enable the casting of extremely complex objects, with the molds here being made up of only two pieces. The moldable objects are not limited to highly challenging geometries with elongated, thin features and surfaces of non-zero genus: they also include knotted shapes and objects made up of multiple entangled pieces that could not be cast using previous techniques.

Figure 3 illustrates the steps required for the physical fabrication of the composite two-piece mold. We first 3D print the two parts of the hard plastic shell (Figure 3.c) and the metamolds whose shape incorporates the geometric details of the parting surface and of the additional membranes (Figure 3.d). Subsequently, we build a closed container by assembling each hard plastic shell part with its respective metamold (Figure 3.e); then, we pour silicone in the container to cast the two flexible silicone parts. Finally, we assemble the hard plastic shell and the flexible silicone parts to create the composite mold, and we use it to cast the input object by pouring casting material (e.g., resin) inside the resulting cavity (Figure 3.f).

In the following: Section 4 describes the design of the cut layout, which includes the parting surface and the additional membranes; Section 5 details how to generate, fabricate and assemble the molds; Section 6 shows a number of fabricated objects and includes a comparison with previous work.

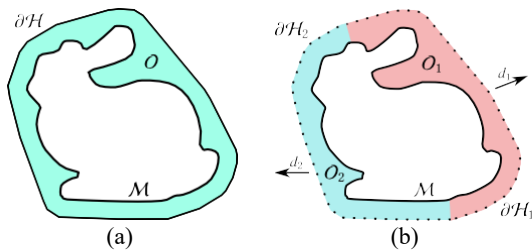


Fig. 4. A 2D schema of the entities involved in the silicone mold design. (a) The silicone mold corresponds to the volume O enclosed by an exterior surface ∂H and the object surface M . (b) Given two parting directions d_1 and d_2 , the parting surface separates the exterior surface into ∂H_1 and ∂H_2 ; hence, the silicone mold is split into two pieces O_1 and O_2 .

4 CUTTING MEMBRANES DESIGN

Our input is a closed triangulated manifold surface $M \subset \mathbb{R}^3$. Consider an offset surface of M , and let H be a tetrahedral tessellation of its convex hull, with M a subcomplex of H . Then, M separates H into two parts: an inside and an outside. The silicone mold part corresponds to the outside O , namely, to the portion of the volume bounded by the surface M and the boundary ∂H (Figure 4.a).

Our membrane generation procedure is divided into two steps: first, we identify the parting surface that splits the mold into two

parts (Section 4.1), and then, we locate the additional membranes required for the mold removal (Section 4.2).

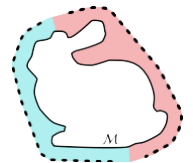
4.1 Parting surfaces

Because we aim to have a two-piece mold, the first step is to identify the two parting directions d_1 and d_2 and the corresponding parting surface. The parting surface will decompose O into two parts, O_1 and O_2 , which correspond to the two silicone mold pieces. We define the parting surface according to the idea that it should separate neighboring points that have escape paths that end at different parts ∂H_1 and ∂H_2 of the external boundary (Figure 4.b).

We begin by finding the parting directions. We uniformly sample k candidate parting directions on the Gaussian sphere. For each candidate direction, we use GPU-accelerated rendering to compute the visible and non-visible areas of M . Then, we simply select as the parting directions the two directions d_1, d_2 which minimize the non-visible surface area. Note that we do not require that the directions induce mold pieces that are fully extractable because we will enforce moldability with additional membranes in a second step (Section 4.2).

Given the two parting directions, we partition the boundary ∂H into two parts, ∂H_1 and ∂H_2 : we select the two faces F_1 and F_2 of ∂H whose normals best align with d_1 and d_2 and then use a greedy region-growing approach from F_1 and F_2 to assign faces to ∂H_1 and ∂H_2 , according to the alignment of their normal to d_1 and d_2 .

The partition of the boundary surface induces a partition of the interior volume as follows: For each interior edge $e = (v_i, v_j)$ in the tessellated volume O , we compute the shortest paths from v_i and v_j to the boundary surface ∂H . Then, the edge is traversed by a parting surface if the destination vertices w_i and w_j belong to different parts $\partial H_1, \partial H_2$ of the boundary surface (Figure 5.a). The resulting parting surface will separate O into O_1 and O_2 , which correspond to the two silicone mold pieces. We compute the shortest paths towards all vertices of ∂H , except for those whose distance from the boundary between ∂H_1 and ∂H_2 is less than a fixed threshold (see dashed lines in the inset). This is done to avoid discretization noise for the shortest paths computation. In our experiments, we set the threshold to 15% of the convex hull bounding box diagonal.



4.2 Additional membranes

Once the silicone mold volume has been partitioned into two pieces, for each piece we define additional membranes corresponding to features that could prevent the mold extraction. A membrane is introduced when the escape paths of two adjacent vertices traverse the volume on different sides of a portion of the object M . The rationale is that a mold piece can be extracted if all of its points can reach the boundary without having to separate around a geometric feature. Otherwise, an additional cut is needed.

We can formalize the idea as follows: For each interior edge $e = (v_i, v_j)$, let w_i and w_j denote the destination vertices of the shortest paths from v_i and v_j to the exterior boundary. We introduce a cutting membrane across the edge e to separate v_i and v_j if a discrete approximation of the minimal surface bounded by the edge

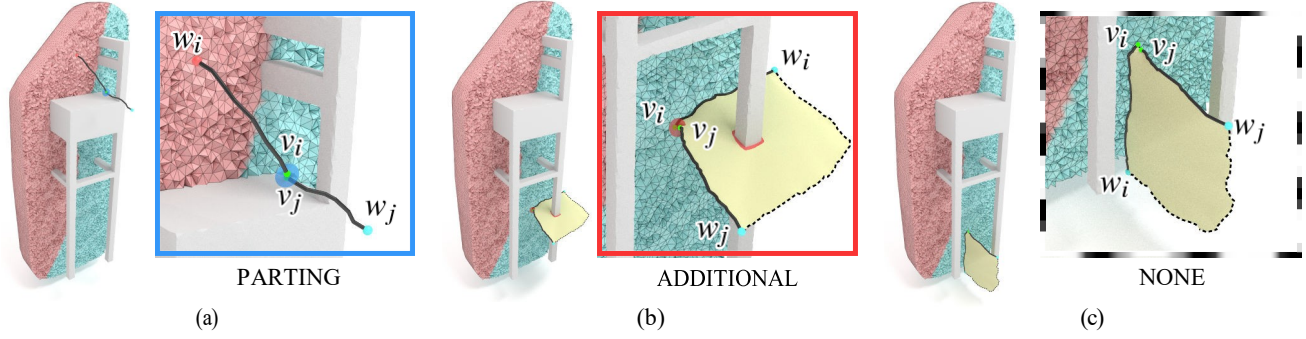


Fig. 5. Membrane computation in the mold volume. (a) The shortest paths from two adjacent interior vertices v_i, v_j end at different sides of the partitioned exterior boundary: the parting surface membrane will cut the edge (v_i, v_j) to separate the vertices. (b) The minimal surface (yellow) spanned by the escape paths of v_i, v_j and the path between the destination vertices w_i, w_j (computed on the boundary), intersects the chair leg: an additional membrane will cut the edge (v_i, v_j) . (c) The minimal surface computed for an edge (v_i, v_j) does not intersect the object: no membrane is inserted.

(v_i, v_j) , the escape paths (v_i, w_i) , (v_j, w_j) , and the shortest path (w_i, w_j) (computed on ∂H), intersects the object M (Figure 5.b,c). The minimal surface is used as a heuristic for defining a criterion to check whether the span of the escape paths from adjacent points encloses part of the object we want to cast. If an intersection is found, then those points cannot be freely extracted along their escape paths unless a membrane separates them.

posing C and the boundaries between C and the object M or the

4.3 Membrane meshing

At the end of the process described in the previous sections, we have a flag for each internal edge that states whether the edge is cut by a membrane (either the parting surface or additional membranes) or not. Extracting a surface from this information is not straightforward because the flags do not directly define a field on O [Bloomenthal 1988; de Araújo et al. 2015]. Indeed, the cutting surfaces can be non-manifold and might not admit an implicit representation on a scalar field. Therefore, we extend the classical marching tetrahedra algorithm [Treece et al. 1999] to include the additional cases needed to generate consistent non-manifold surfaces, by following the ideas in [Bloomenthal and Ferguson 1995; Bonnell et al. 2003; Nielson and Franke 1997]. Specifically, because each of the six edges in a tetrahedron can be either cut or not, we encode a table of $2^6 = 64$ possible configurations. An alternative solution would be to use dual methods, as in SurfaceNets [Gibson 1998].

4.4 Membrane smoothing

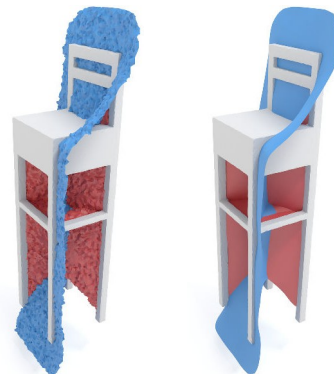
The triangulated surface C encoding the cut layout is composed using a set of patches that are interconnected by chains of non-manifold edges that are bounded by construction by the object surface mesh M and the external boundary ∂H . Unfortunately, C can be noisy. Indeed, because our cuts are identified as a set of traversed edges on the tetrahedral mesh, the resulting mesh C is inevitably affected by the quality and density of the tetrahedral tessellation (Figure 6, left). Therefore, we perform a Laplacian smoothing that preserves both the boundaries between the different patches com-

boundary ∂H (Figure 6, right). The smoothing is performed by alternating two main steps. First we smooth the polyline that includes the boundary vertices only. After this smooth step, we re-project those vertices onto the original surface of M or the external boundary ∂H , depending on which of the two surfaces they belonged to originally. Then, we smooth all the interior vertices, keeping the ones on the boundary fixed (the ones smoothed in the previous step). Each smoothing step is performed using a damping factor (0.5 in our experiments) to ensure a proper convergence to the final solution.

4.5 Shortest path computation

The placement of cutting membranes in the mold volume depends on an analysis of the shortest paths from interior vertices to the boundary. We found that using standard geodesics could result in badly shaped membrane configurations (Figure 7). For example, membranes that travel the volume almost tangentially to the object surface would produce long, thin slivers of silicone that would be difficult to handle (Figure 7.a, top). To avoid this, we use weighted geodesics that push away membranes from the object surface. We compute the shortest paths from the interior vertices to the boundary according to a metric, which makes traveling near the object

Fig. 6. Noisy membranes (left) and their smoothed version (right).



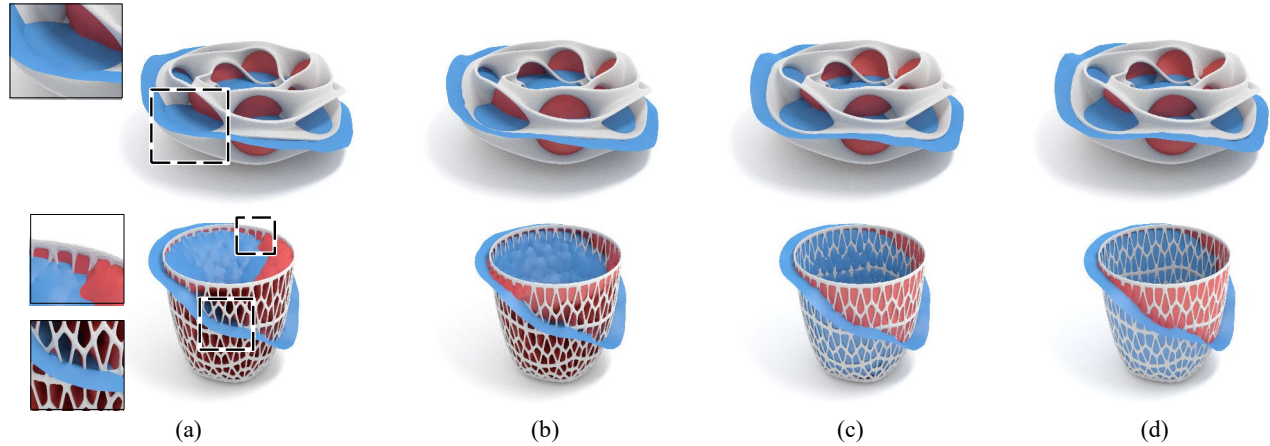


Fig. 7. The effects of the weighting factor for geodesics and the reshaping of the exterior boundary of two examples, here used both singularly and in combination with each other. (a) The membranes computed using plain geodesics and no exterior boundary reshape do not well adapt to the object geometry: (top) part of the membranes are tangent to the object surface, and (bottom) they do not align to the bowl's tiny holes. (b) The effect of adding the exterior boundary reshape to the membrane computation. (c) The effect of adding the Euclidean distance weighting factor into the geodesic path computation: whereas (top) the membranes align to the object features, avoiding the creation of thin slivers near the surface, some artifacts can still be seen in the interior of the bowl (bottom). (d) The combined effect of both the reshape and the weighting factor.

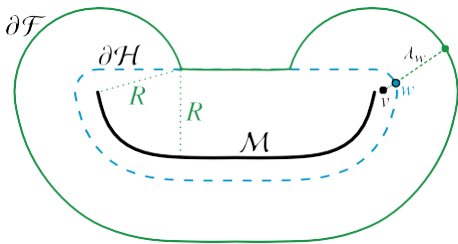


Fig. 8. We define the distance from an interior vertex v to a point w on ∂H as the sum $d(v, w) + \lambda_w$, with d being the weighted geodesic distance and λ_w the weighted distance from w to an offset surface ∂F of M . The offset radius R is the maximum distance from points on the convex hull to M .

longer. This is done by weighting the Euclidean arc length by a scalar function that depends on the squared distance from M . In detail, the shortest paths are computed by multiplying the Euclidean length $l(e)$ of an edge e by a weighting factor $w_e = \frac{1}{d_e^2 + \epsilon}$. The value d_e is the geodesic distance of the midpoint of e from M , and ϵ is a constant added to avoid division-by-zero and numerical issues (set to 0.25 in our experiments). The shortest paths are computed using Dijkstra's algorithm on the connectivity graph of the tetrahedra vertices.

Our weighted geodesics produce paths that tend to align to the gradient of the distance field from the object surface and, therefore, membranes that tend to be perpendicular to the surface.

In addition, computing the shortest paths to the boundary of the convex hull may result in membranes that do not align well to the object geometry because the shape of the convex hull may discard essential features of the original object shape. For example, for the bowl in Figure 7.a (bottom), the convex hull does not retain

information about the bowl being shaped like a container. Therefore, we compute the shortest paths to a surface that better retains information about the shape of M , namely, an offset surface ∂F of M that encloses the convex hull in its interior. The offset radius is set as the maximum distance from the points on the convex hull to M . To avoid computing the tetrahedrization of the larger volume bounded by ∂F , we simply bias the metric by storing on the convex hull vertices their distance from ∂F and adding this distance to the computed one (Figure 8). Figure 7 shows the effects of the two design choices above, namely, the weighted geodesics and modified exterior boundary.

5 COMPOSITE MOLD FABRICATION

Figure 9 summarizes the whole fabrication and assembly pipeline once the parting and additional membranes have been designed. To physically fabricate the composite two-piece mold, we use liquid silicone casting for the two flexible parts, and 3D printing for the hard plastic shell.

We 3D print a metamold whose shape incorporates the geometric details of the parting surface plus the additional membranes as solid shells, as in [Alderighi et al. 2018]. The hard shell is a prism aligned with the resin pouring direction, from which we then subtract the inner soft mold volume; the prism's flat base is orthogonal to the resin pouring direction and shaped like the silhouette of the convex hull. The metamold and the plastic shell are cut so that they exactly match, using Boolean operations [Zhou et al. 2016] (see Figure 10). Section 5.1 describes our strategy to design the interface surface between the hard and the soft part, while Section 5.2 describes our strategy to choose the pouring directions that prevent the presence of air bubbles during both silicone and resin casting operations to avoid artifacts on the cast model.

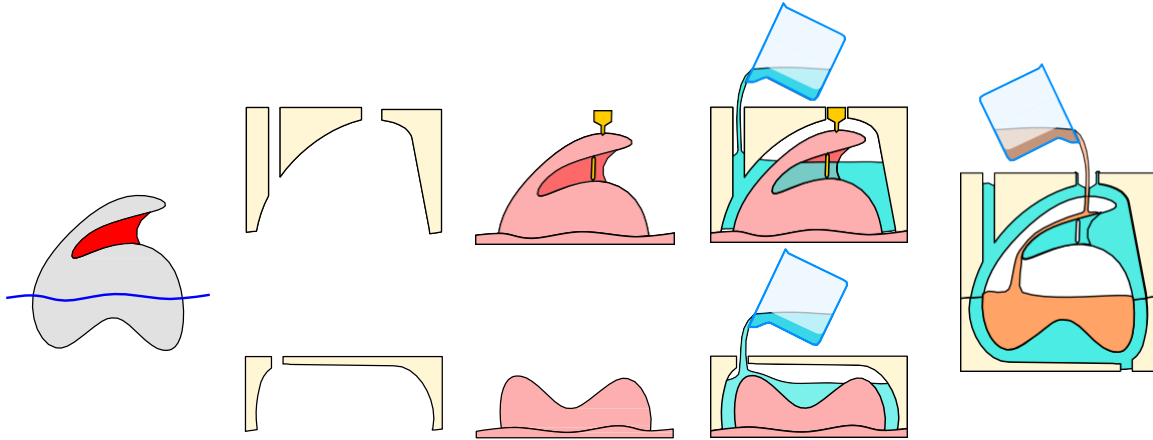


Fig. 9. From left to right, the fabrication and assembly pipeline. Given the parting surface (blue) and the additional membranes (red), we 3D print four pieces: two hard shell pieces (light yellow) and two metamolds (light red). Then, we assemble the pieces (a metamold and a piece of the hard shell) to build two casting containers where we pour silicone (cyan). The two silicone molds are then used together with the hard shell to assemble our composite mold, where we pour the resin (orange).

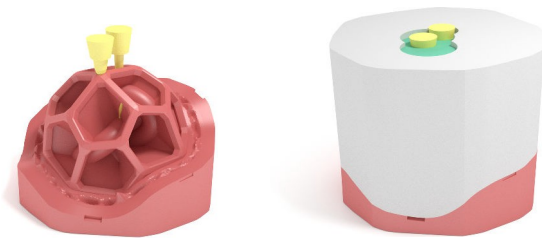


Fig. 10. Air vents and air pipes are considered in the final geometry of the hard shell and the of metamold to pour the casting liquid and let air out.

We build a closed container by assembling one of the pieces of the hard plastic shell and the metamold, and pour silicone inside to cast the silicone part. Finally, we assemble the hard shell and the flexible silicone parts so that we can fabricate multiple copies of the input object by pouring resin (or another casting material) inside the cavity of the assembled composite mold.

Perlin noise is added to the parting surface between the two silicone pieces to improve registration. Perlin noise favors the alignment between fabricated mold pieces. Indeed, aligning irregular surfaces is less error-prone in contrast to flat or very smooth surfaces. In this way, we aim to minimize the misalignment artifacts in the fabricated object.

Once we have the optimal pouring directions, we add air vents in the hard plastic shell and air pipes in the metamolds to pour silicone and resin, respectively, and let air escape (Figure 10).

5.1 Mold shape design

The interface between the silicone part and the hard plastic shell can be designed in different ways. Hence, we evaluated two distinct choices: the inflated convex hull of the object surface and an offset surface modified to correspond to a height field in the resin casting

direction (Figure 11). The latter choice makes the silicone part thinner and more similar to traditional glove molds. The advantage of a silicone part that is thinner and more fit to the input surface is that it is more flexible and easier to remove. We set a fixed offset of $15mm$ for all our models, whose size is in the range between $95mm$ to $300mm$. For both of the silicone mold shape choices described above, this distance sets the minimal thickness of the soft part of the composite mold. The hard shell, being rigid, keeps this distance small even when increasing the object size.

5.2 Persistence pairing for selecting pouring directions

Air bubbles may get trapped at the local maxima for the pouring direction. Assuming we can slightly tilt the mold while casting, some of the bubbles can flow away and some cannot, depending on the shape around the local maximum. Therefore, we need a criterion to select the silicone and resin casting directions that can minimize the presence of *relevant* local maxima.

A pouring direction can be seen as a height function $f : M \rightarrow R$. We use the *pairing* mechanism from persistence homology [Edelsbrunner et al. 2000] to assess the relevance of local maxima for a



Fig. 11. Different choices for the contact surface between the silicone part and the hard plastic shell.

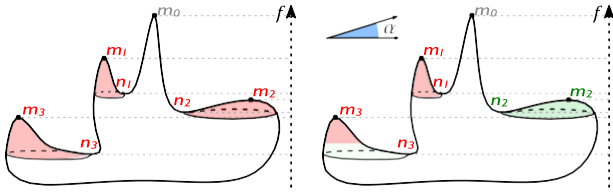


Fig. 12. Left: maximum-saddle pairs identify the regions where air bubbles can get trapped while pouring liquid (red areas); right: when tilting the mold by an angle α , air can escape from the green regions, yet it remains trapped in the red regions. For a candidate pouring direction, the area of the red regions gives an estimate of the visible artifacts that would result from air bubbles.

given pouring direction. *Persistence pairing* is a method to compute sequences of critical points (m, n) for a real-valued function f defined on a topological space. According to Morse theory [Milnor 1963], one can think of m as the point creating a topological feature (e.g., a connected component) and n as the point destroying it, while considering the sequence of superlevel sets $M_a = f^{-1}(a, +\infty)$ as the value a decreases. We are interested in the pairs (m, n) where m is a maximum and n is a saddle: m marks the birth of a new connected component and n marks its death as it merges with another component. Therefore, pairs (m, n) locate the areas where air bubbles can get trapped (Figure 12, left).

Persistence pairs are usually sorted according to their persistence $|f(m) - f(n)|$ under the assumption that the pairs with a high persistence indicate relevant features, while low-persistence ones indicate noise. This sorting creates a hierarchy that was used to filter out irrelevant pairs for simplification [Edelsbrunner et al. 2000] or, conversely, to identify relevant features for segmentation and matching [Biasotti et al. 2013]. Here, we modify the sorting criterion, to get for a candidate pouring direction a hierarchy depending on the amount of air which would be trapped in each feature identified by a maximum-saddle pair. Low-ranked pairs can be neglected, whereas high-ranked, relevant pairs encode features that are likely to create visible artifacts in the cast model.

Given a candidate pouring direction, we compute a relevance score that corresponds to the amount of air that would still remain trapped at m if tilting was considered. If no tilting was assumed, a measure of the amount of air trapped in a maximum-saddle pair

would be the area of the region A_m^n that grows from the maximum

m until points are reached with value $f(n)$. If we assume that we can slightly tilt the mold while casting by an angle α , then we can

reduce A_m^n by starting from the saddle and growing a set T of faces

whose normal forms an angle less than α with the direction f . Then, the relevance score of the pair (m, n) is the area of the region $A_m^n - T$ (Figure 12, right).

We filter out all maximum-saddle pairs whose relevance is below a fixed threshold ($0.5mm$ in our experiments). Finally, we sum the

For each piece P_1, P_2 of the soft silicone part, we evaluate the score of a set of candidate directions for silicone pouring and choose from those scoring lowest a couple of nearly aligned directions f_1, f_2 . The resin casting direction is then chosen as the lowest scoring one in a set of directions sampled in a small cone around the bisector of f_1, f_2 (opening angle of 10° in our experiments). Having nearly aligned silicone and resin pouring directions is required for the silicone to have a good surface to level out while casting.

6 RESULTS

We successfully fabricated a number of highly complex objects. Table 1 reports the statistics about the models and timings to complete the entire computational pipeline on a regular desktop machine. We used OpenVDB [Museth et al. 2013] to compute offset surfaces and TetWild [Hu et al. 2018] to produce a tetrahedral mesh of the mold volume. Even though TetWild produces a tetrahedral mesh that does not conform precisely to the boundaries, it is robust and produced a uniform tessellation that approximates the shapes well enough for all models except for Medusa, for which we used TetGen [Si 2015] instead. In TetWild, we set the target mean edge length to be 0.0065 times the bounding box diagonal.

The hard shell, the metamolds, and the pipes were printed using different 3D printers (Ultimaker 2+, Ultimaker 5S, and Stratasys J750). We used standard silicone (available at hobby stores) for casting the soft molds and a simple bi-component resin for casting the final objects.

Our approach can handle objects with intricate geometric features (Figure 13) as well as non-zero genus objects, up to knotted surfaces and objects consisting of multiple entangled pieces (Figure 14). To the best of our knowledge, no automatic molding pipeline enables the successful casting of objects of such complexity. The composite molds enabled us to fabricate multiple copies of objects easily (Figure 15). The extraction process required little effort and time, and no damage was observed either to the mold or to the cast objects.

A key ingredient to the success of our approach is the design of smart membranes that release tension while removing the silicone mold: our membranes are aware of the volume surrounding the object; they stick to the object features until they vanish in the volume, and they can freely intersect with each other (Figure 16).

To quantitatively validate our results, we measured the error introduced by the fabrication process for three models (Table 2). The measurement is based on the Hausdorff distance between a 3D scan

of the cast object and its digital model. We measured the error on

two different replicas for each of the three models. The errors are relatively low, especially considering the unavoidable resin shrink-

age and the fact that the casts were performed by non-professionals

scores of the remaining pairs to get a global score for each candidate pouring direction. Good pouring directions are those with a low score, that is, those that are likely to minimize the risk of causing visible artifacts.

and without any special equipment (e.g., vacuum chambers).

We compared our method with the state-of-the-art Metamolds technique in [Alderighi et al. 2018]. Figure 17 shows how our method always provides a valid solution for all the models in their dataset; here, we can use a two-piece mold to cast the goblet model, which was shown to require three pieces in [Alderighi et al. 2018]. In addition, our method leads to a valid mold design even for failure cases of Metamolds (Figure 18). Being based on surface properties only, Metamolds failed to produce a valid segmentation for some

Table 1. Model statistics and timings for the examples shown in results. The columns report the number of vertices and faces of the digital object, the number of tetrahedra of the surrounding volume discretization, and the execution times of the different algorithm steps: $t_{tetrahedra}$ is the time required by the tetrahedralization software we used (TetWild or Tetgen); $t_{membrane}$ is the time required to compute the escape paths and membranes; t_{smooth} is the time required to apply the Laplacian smoothing; and t_{mold} is the running time of the composite mold geometry generation procedure, requiring the computation of several boolean mesh operations. Timings were set using a desktop computer equipped with an Intel Core i7-6700K processor (4.00Ghz).

Model	Surface (#verts $\dot{\sim}$ #faces)	Volume (#verts $\dot{\sim}$ #tets)	$t_{tetrahedra}$	$t_{membrane}$	t_{smooth}	t_{mold}
Medusa	156,556 $\dot{\sim}$ 313,260	2,829,558 $\dot{\sim}$ 17,056,249	1m (Tetgen)	58m	5s	27m
Laocoonte	30,944 $\dot{\sim}$ 61,724	461,194 $\dot{\sim}$ 2,523,800	39m	6m	2s	9m
Dragon Head	418,566 $\dot{\sim}$ 837,136	743,955 $\dot{\sim}$ 1,958,062	47m	6m	3s	13m
Dragon	14,510 $\dot{\sim}$ 29,020	300,173 $\dot{\sim}$ 1,658,964	35m	4m	3s	10m
Wheel of Life	628,920 $\dot{\sim}$ 1,257,856	479,375 $\dot{\sim}$ 2,629,168	41m	6m	4s	9m
Knot	28,729 $\dot{\sim}$ 57,458	578,479 $\dot{\sim}$ 3,221,235	37m	16m	3s	20m
Caged Knot	30,593 $\dot{\sim}$ 61,274	602,332 $\dot{\sim}$ 3,354,202	47m	25m	9s	50m

Table 2. Hausdorff distance RMS errors between a 3D scan of the cast object and its digital model. In parenthesis we show the percentage of RMS error over the diagonal length. All values are expressed in *mm*. Size stands for the diagonal of the bounding box for each model. The 3D scans were taken using a GOM Atos scanner that yields a precision of 0.2*mm*.

Model	Size	Error #1 (%)	Error #2 (%)
Dragon Head	286	0.493 (0.172)	0.488 (0.171)
Wheel of Life	181	0.264 (0.146)	0.357 (0.197)
Laocoonte	173	0.178 (0.103)	0.181 (0.105)

classes of models. Metamolds’ topological membranes are unable to handle complex knotted loops (Figure 18, top) or, depending only on topological features, in some cases might hinder the extraction (Figure 18, middle). Also, for complex geometries like the dragon model, Metamolds might fail to produce a valid segmentation (Figure 18, bottom). Our method, instead, exploits all of the available information about the volume around the objects, thus providing membranes that accommodate even for such complex shapes.

7 CONCLUSION

We have introduced a novel technique to automatically design composite, two-piece molds that are well-suited for casting highly complex shapes. For the first time, the proposed approach extends computational design to a kind of mold type that so far has been reserved to skilled reproduction artisans to cast works of art.

Our technique is based on an innovative volumetric analysis of the possible escape paths in the volume surrounding the object. This analysis allows us to understand how the mold could be released from the cast object and determines how it should be cut and opened. The proposed volumetric approach overcomes previous limitations in terms of the geometric and topological complexity.

The main limitations of the method are mostly related to the intrinsic properties of casting. First of all, bottle-shaped objects with large cavities and small escape holes cannot be reproduced by simple casting. Another challenging configuration is extremely long thin parts that are almost isolated from the rest of the object (think, for example, of a long spear kept in a hand); although in this case our approach would produce a valid result, the resulting mold could be unnecessary large and complicated. These cases are traditionally solved by cleverly decomposing the object into a few separate parts

that are significantly easier to cast; the problem of how to perform this kind of decomposition in a robust way on large complex models is an interesting open research direction. Furthermore, even if our formulation produced good membranes for the range of complex objects we fabricated, for certain cases, it is possible that our method would produce unfeasible silicone molds: for example, on a sphere with a thin and long cylindrical cavity, our method would produce a correctly extractable mold but with a very thin cylinder of silicone that would be deformed during the casting procedure. Detecting and handling this kind of problem is an interesting research challenge.

Finally, a practical limitation is given by the need for a volumetric tetrahedral decomposition; although there are many robust techniques that claim to work for highly detailed boundaries, we have found that the sheer complexity of some of the models proposed by artists is challenging for all the available tools. Moreover, multiresolution approaches are not trivial to apply because it is difficult to robustly transfer the generated geometry, such as the membranes defining the cuts, between models of different resolutions.

Besides its limitations, we believe our approach significantly extends the range of models that can be fabricated by molding. Also, a potential application could be metal casting. While we were able to get partially satisfying results with a pewter-based alloy (Figure 19), metal casting presents very hard constraints, and further investigation is needed to get a truly robust solution.

ACKNOWLEDGMENTS

The authors thank Marco Callieri for helping with the resin casts and 3D scanning, Jon O’Neill and Tarrant Saphin from UTS Protospace for helping with 3D printing, and Marco Tarini for helping with the smoothing code. The models are courtesy of the AIM@SHAPE Shape Repository, Stanford 3D Scanning Repository, UC Berkeley Rapid Prototyping Project; the Dragon head is courtesy of the user mcallaghan95 on Thingiverse. The research was partially funded by the EU H2020 Programme EMOTIVE: EMOTIVE Virtual cultural Experiences through personalized storytelling (grant no. 727188), the European Research Council (ERC) MATERIALIZABLE: Intelligent fabrication oriented Computational Design and Modeling (grant no. 715767), the EU H2020 Programme EVOCATION: Advanced Visual and Geometric Computing for 3D Capture, Display, and Fabrication (grant no. 813170), and by the Italian PRIN project DSURF (grant no. 2015B8TRFM).



Medusa



Laocoonte



Dragon head



Dragon



Fig. 13. Successful casting of complex shapes.

REFERENCES

- Thomas Alderighi, Luigi Malomo, Daniela Giorgi, Nico Pietroni, Bernd Bickel, and Paolo Cignoni. 2018. Metamolds: Computational Design of Silicone Molds. *ACM Trans. Graph.* 37, 4, Article 136 (July 2018), 13 pages. <https://doi.org/10.1145/3197517.3201381>
- Amit H. Bermanto, Thomas Funkhouser, and Szymon Rusinkiewicz. 2017. State of the Art in Methods and Representations for Fabrication-Aware Design. *Comput. Graph. Forum* 36, 2 (May 2017), 509-535. <https://doi.org/10.1111/cgf.13146>
- S. Biasotti, A. Cerri, D. Giorgi, and M. Spagnuolo. 2013. PHOG: Photometric and Geometric Functions for Textured Shape Retrieval. (2013), 13-22. <https://doi.org/10.1111/cgf.12168>
- Bernd Bickel, Paolo Cignoni, Luigi Malomo, and Nico Pietroni. 2018. State of the Art on Stylized Fabrication. *Computer Graphics Forum* 37, 6 (2018), 325-342. <https://doi.org/10.1111/cgf.13327>
- J. Bloomenthal. 1988. Polygonization of Implicit Surfaces. *Comput. Aided Geom. Des.* 5, 4 (Nov. 1988), 341-355. [https://doi.org/10.1016/0167-8396\(88\)90013-1](https://doi.org/10.1016/0167-8396(88)90013-1)
- Jules Bloomenthal and Keith Ferguson. 1995. Polygonization of Non-manifold Implicit Surfaces. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 309-316. <https://doi.org/10.1145/218380.218462>
- Kathleen S. Bonnell, Mark A. Duchaineau, Daniel R. Schikore, Bernd Hamann, and Kenneth I. Joy. 2003. Material Interface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 9, 4 (Oct. 2003), 500-511. <https://doi.org/10.1109/TVCG.2003.1260744>
- Pritam Chakraborty and N. Venkata Reddy. 2009. Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds. *Journal of Materials Processing Technology* 209, 5 (2009), 2464-2476. <https://doi.org/10.1016/j.jmatprotec.2008.05.051>



Fig. 14. Successful casting of a surface of non-zero genus, a knotted shape, and an object composed of two entangled pieces.



Fig. 15. Multiple casts.

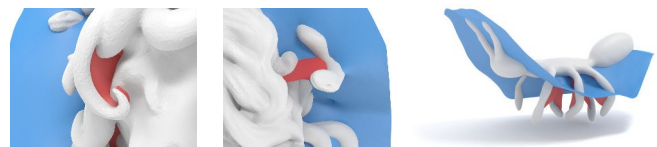


Fig. 16. Left and center: close-ups of the additional membranes on the protruding features of the Medusa head model. Right: parting surface and additional membranes on an ant model.

B. R. de Araújo, Daniel S. Lopes, Pauline Jepp, Joaquim A. Jorge, and Brian Wyvill. 2015. A Survey on Implicit Surface Polygonization. *ACM Comput. Surv.* 47, 4, Article 60 (May 2015), 39 pages. <https://doi.org/10.1145/2732197>

Ernest P De Garmo, J Temple Black, and Ronald A Kohser. 2011. *DeGarmo's materials and processes in manufacturing*. John Wiley & Sons.

H. Edelsbrunner, D. Letscher, and A. Zomorodian. 2000. Topological Persistence and Simplification. (2000), 454s. <http://dl.acm.org/citation.cfm?id=795666.796607>

Sarah F. Frisken Gibson. 1998. Constrained Elastic Surface Nets: Generating Smooth Surfaces from Binary Segmented Data. In *Proceedings of the First International*

Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI '98). Springer-Verlag, London, UK, UK, 888s898. <http://dl.acm.org/citation.cfm?id=646921.709482>

Eric Hart. 2013. *The Prop Building Guidebook: For Theatre, Film, and Tv*. Taylor & Francis.

Philipp Herholz, Wojciech Matusik, and Marc Alexa. 2015. Approximating Free-form Geometry with Height Fields for Manufacturing. *Comput. Graph. Forum* 34, 2 (May 2015), 239s251. <https://doi.org/10.1111/cgf.12556>

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. <https://doi.org/10.1145/3197517.3201353>

Alan C. Lin and Nguyen Huu Quang. 2014. Automatic generation of mold-piece regions and parting curves for complex CAD models in multi-piece mold design. *Computer-Aided Design* 57 (2014), 15 s 28. <https://doi.org/10.1016/j.cad.2014.06.014>



Fig. 17. Running our method on the dataset of Metamolds [Alderighi et al. 2018] always produces a valid set of membranes.

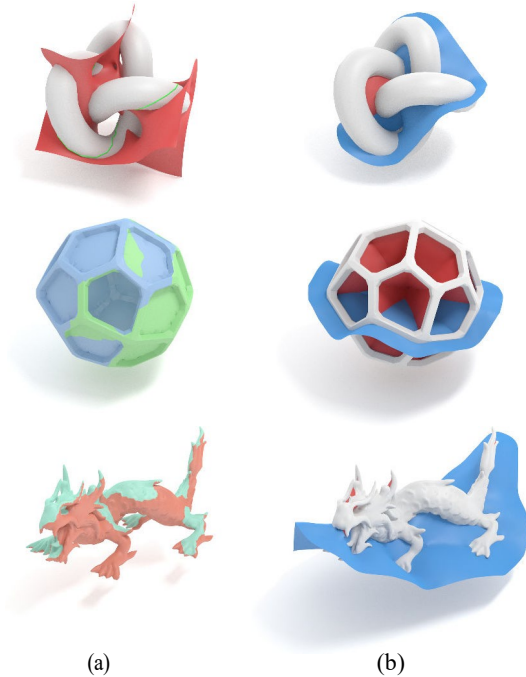


Fig. 18. Some examples where Metamolds [Alderighi et al. 2018] failed to produce a valid surface partitioning (a), whereas our method produced a usable set of membranes (b). Metamolds' membrane generation is unable to handle complex knotted loops (top) and might create membranes that render the extraction impossible (middle). For complex geometries, Metamolds' visibility-based approach may produce isolated segments and induce silicone molds with non-zero genus (bottom).

Ligang Liu, Ariel Shamir, Charlie Wang, and Emily Whiting. 2014. 3D Printing Oriented Design: Geometry and Optimization. In *SIGGRAPH Asia 2014 Courses (SA '14)*. ACM, New York, NY, USA, Article 1. <https://doi.org/10.1145/2659467.2675050>
 Luigi Malomo, Nico Pietroni, Bernd Bickel, and Paolo Cignoni. 2016. FlexMolds: Automatic Design of Flexible Shells for Molding. *ACM Trans. Graph.* 35, 6, Article 223

(Nov. 2016), 12 pages. <https://doi.org/10.1145/2980179.2982397>
 J. Milnor. 1963. *Morse Theory*. Princeton University Press, New Jersey.
 Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. 2013. OpenVDB: An Open-source Data Structure and Toolkit for High-resolution Volumes. In *ACM SIGGRAPH 2013 Courses (SIGGRAPH '13)*. ACM, New York, NY, USA, Article 19, 1 pages. <https://doi.org/10.1145/2504435.2504454>
 Kazutaka Nakashima, Thomas Auzinger, Emmanuel Iarussi, Ran Zhang, Takeo Igarashi, and Bernd Bickel. 2018. CoreCavity: Interactive Shell Decomposition for Fabrication with Two-piece Rigid Molds. *ACM Trans. Graph.* 37, 4, Article 135 (July 2018), 13 pages. <https://doi.org/10.1145/3197517.3201341>
 Gregory M. Nielson and Richard Franke. 1997. Computing the Separating Surface for Segmented Data. In *Proceedings of the 8th Conference on Visualization '97 (VIS '97)*. IEEE Computer Society Press, Los Alamitos, CA, USA, 229&233. <http://dl.acm.org/citation.cfm?id=266989.267066>
 A. Medeiros e Sá, K. Rodriguez-Echavarría, N. Pietroni, and P. Cignoni. 2016. State of the Art on Functional Fabrication. In *Proceedings of the Eurographics Workshop on Graphics for Digital Fabrication (GraDiFab '16)*. Eurographics Association, Goslar Germany, Germany, 189. <https://doi.org/10.2312/gdf.20161073>
 Christian Schüller, Daniele Panozzo, Anselm Grundhöfer, Henning Zimmer, Evgeni Sorkine, and Olga Sorkine-Hornung. 2016. Computational Thermoforming. *ACM Trans. Graph.* 35, 4, Article 43 (July 2016), 9 pages. <https://doi.org/10.1145/2897824.2925914>
 Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (Feb. 2015), 36 pages. <https://doi.org/10.1145/2629697>
 Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 2016. 3D Skeletons: A State-of-the-art Report. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: State of the Art Reports (EG '16)*. Eurographics Association, Goslar Germany, Germany, 573&597. <https://doi.org/10.1111/cgf.12865>
 G.M. Treece, R.W. Prager, and A.H. Gee. 1999. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics* 23, 4 (1999), 583 & 598. [https://doi.org/10.1016/S0097-8493\(99\)00076-X](https://doi.org/10.1016/S0097-8493(99)00076-X)
 Nobuyuki Umetani, Bernd Bickel, and Wojciech Matusik. 2015. Computational Tools for 3D Printing. In *ACM SIGGRAPH 2015 Courses (SIGGRAPH '15)*. ACM, New York, NY, USA, Article 9. <https://doi.org/10.1145/2776880.2792718>
 Somlak Wannarumon. 2011. Reviews of Computer-Aided Technologies for Jewelry Design and Casting. *Naresuan University Engineering Journal* 6, 1 (2011), 45&56.
 Chunjie Zhang, Xionghui Zhou, and Congxin Li. 2010. Feature extraction from freeform molded parts for moldability analysis. *The International Journal of Advanced Manufacturing Technology* 48, 1 (01 Apr 2010), 273&282. <https://doi.org/10.1007/s00170-009-2273-7>
 Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh Arrangements for Solid Geometry. *ACM Trans. Graph.* 35, 4, Article 39 (July 2016), 15 pages. <https://doi.org/10.1145/2897824.2925901>



Fig. 19. A first attempt to perform metal casting.