FlexMolds: Automatic Design of Flexible Shells for Molding

Luigi Malomo[1,2,3] Nico Pietroni[1] Bernd Bickel[3] Paolo Cignoni[1]

[1]Visual Computing Lab, ISTI - CNR [2]University of Pisa [3]IST Austria

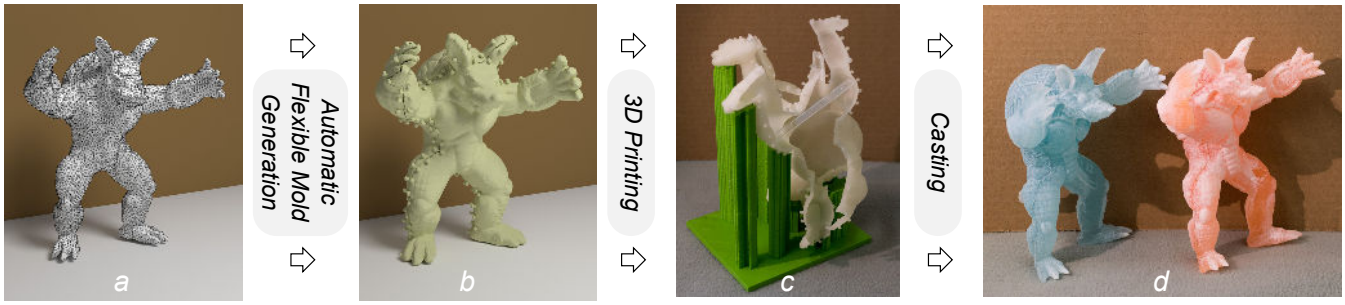

Figure 1: *Starting from a 3D model (a), we automatically generate a set of cuts over its surface that allow the generation of a flexible mold shell (b) that can be 3D printed (c) and used for casting multiple physical copies (d) of the original model.*

Abstract

We present FlexMolds, a novel computational approach to automatically design flexible, reusable molds that, once 3D printed, allow us to physically fabricate, by means of liquid casting, multiple copies of complex shapes with rich surface details and complex topology. The approach to design such flexible molds is based on a greedy bottom-up search of possible cuts over an object, evaluating for each possible cut the feasibility of the resulting mold. We use a dynamic simulation approach to evaluate candidate molds, providing a heuristic to generate forces that are able to open, detach, and remove a complex mold from the object it surrounds. We have tested the approach with a number of objects with nontrivial shapes and topologies.

Keywords: digital fabrication, casting, modeling

Concepts: •**Computing methodologies** → **Shape modeling;**

1 Introduction

Mold casting is a widely used manufacturing process for rapidly producing shapes. It allows the efficient shaping of a wide range of materials, such as resins, and scales well with the number of required copies. However, reusable molds come with the requirement of being detachable from the actual molded part without physically destructing the mold or the molded part. This poses hard constraints on reproducible shapes. For rigid molds, the cavity of a mold piece must resemble a height field. This results in a trade-off between reproducible shape complexity and the number of required mold pieces. Each mold piece should be designed by keeping in mind the complete movement that is needed to separate it from the cast object. Eventually, shapes with high genus or very complex geometries may require to be split into multiple pieces, which can be individually cast and assembled afterwards. In general, automatic mold design is a very complex task since it must satisfy multiple non-local constraints.

A different strategy is silicone mold casting [Bruckner et al. 2010]. An object is surrounded by silicone, and after curing, the silicone is cut out and detached from the original object, obtaining a flexible mold. This allows the creation of stunning replicas of objects with highly complex geometry. However, so far this has been a labor-intensive manual process, requiring a positive copy of the shape, performing a silicone cast, and cutting the silicone mold. Furthermore, in practice, deciding and performing these manual cuts is challenging, especially in the presence of complex shapes and topologies.

Inspired by this approach, we propose a novel reproduction approach for highly detailed free-form shapes, based on *flexible mold shells* (FlexMolds). FlexMolds are made of a thin but still sufficiently shape-preserving shell of deformable material, which can be printed by using a commercial 3D printing device. Thanks to their flexibility, they provide more freedom during the removal process, allowing the reproduction of complex shapes, often even with just a single mold piece. Our fabrication method inherits the main advantages of reusable mold casting manufacturing techniques: the mold shell can be reused for multiple copies, the fabrication process is fast and cheap compared with 3D printing, there is a wide range of manufacturing materials and colors available (see Figure 1).

FlexMolds are designed with a sufficient number of cuts that allows the extraction of the internal object once the cast liquid is solidified, without damaging the mold. Cut placement is a complex non-local problem. A proper cut layout should be designed not only to guarantee local detachment, but also to allow the mold to completely depart from the cast object. This may include complex, global, large-scale deformations that allow the flexible mold to adapt to geometric constraint during the extraction (see Figure 2). To guarantee the existence of a feasible extraction sequence we simulated the entire extraction process. Specifically, we are interested in ensuring the existence of a feasible extraction sequence that keeps the deformation within a certain range. Eventually, the simulated ex-

SA '16 Technical Papers,, December 05-08, 2016, , Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982397

ACM Reference Format
Malomo, L., Pietroni, N., Bickel, B., Cignoni, P. 2016. FlexMolds: Automatic Design of Flexible Shells for Molding. ACM Trans. Graph. 35, 6, Article 223 (November 2016), 12 pages.
DOI = 10.1145/2980179.2982397 http://doi.acm.org/10.1145/2980179.2982397.

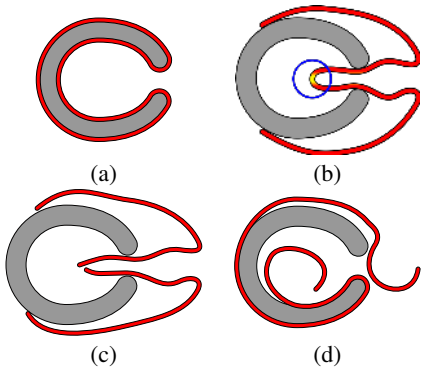

Figure 2: *A simple 2D example: (a) the model to be cast; (b) thanks to their flexibility, FlexMolds allow a simple removal process; however, the stress can be concentrated on a point (blue circle); (c) additional cuts can be added to reduce the stress induced by the removal process; (d) the same effect may be obtained by accurately placing the cut in a different position.*

traction sequence may be different from the one performed by the user.

Ideally, we would like to use as few and as short cuts as possible. However, optimizing cut design is not a well-posed problem, i.e., the extraction sequence does not change continuously with respect to the cut. Figures 2c and 2d show that a small change in the cut topology may cause an abrupt change in the extraction sequence.

2 Related Work

In recent years, the computer graphics community has contributed with many shape-processing approaches and design tools for computing physically realizable objects. While a comprehensive overview can be found in [Liu et al. 2014; Umetani et al. 2015], the following sections will focus on setting our work in context with closely related work.

Mold design Molding is a widely used manufacturing process for rapidly producing shapes. While ancient molds, for example, for creating clay figurines or spear heads, are proof that the technique has existed for thousands of years, mold design is still a highly challenging and important topic in engineering. Expendable mold casting allows the reproduction of shapes of similar complexity as 3D printing with a wide range of materials beyond readily available 3D printing materials [Singh 2010] and is used, for example, for wax casting of digitally designed jewelry [Wannarumon 2011]. In our work we focus on non-expendable mold casting, i.e., molds that can be reused for multiple production cycles. An introduction to manufacturing processes can be found in [De Garmo et al. 2011].

Reusable molds are usually assumed to be made out of a rigid material, such as metal, which restricts the complexity of shapes that can be manufactured as the shape needs to be removable from the mold without damaging them. Therefore, the parting direction and parting surfaces of molds play an important role. Chakraborty and Reddy [2009] proposed a joint optimization to minimize the area of undercuts, the flatness of the parting surface, and minimized draw depth for determining the best parting direction, parting line, and surface for a two-piece molded component. Zhang et al. [2010] performed a geometric analysis of the shape to identify potential undercuts with their possible withdrawal directions; this analysis provides decision support information for designers choosing parting direction, parting lines, and surfaces. For more complex models, Lin and Quang [2014] proposed a heuristic to segment the surface of an object and compute feasible parting directions for automatically generating multi-piece molds, and [Hu et al. 2014] explored the decomposition of shapes into pyramidal pieces. To address the problem that complex geometries might require an extensive number of mold parts, Herholz [2015] proposed a method for approximating free-form geometry with height fields in which they deform the shape slightly in order to meet the height field constraints needed for rigid casting. Thermoforming techniques may accurately reproduce shape and color [Schüller et al. 2016]. While these methods are suitable for large-scale production, their use is limited for the fabrication of simple geometric shapes. In our work, by exploiting the properties of flexible molds, we follow a very different strategy that does not require any changes to the original geometry and is able to manage objects with complex geometry and topology. Traditionally, flexible molds are made by surrounding an object with silicone and then, after curing, iteratively cutting the silicone until the object can be removed. While this provides the potential to reproduce highly complex shapes, so far the design of such molds relies on a manual process and well-trained users [Bruckner et al. 2010]. Furthermore, it is a two-step process requiring a positive of the shape before the mold can be manufactured. In contrast, our method is fully automatic and allows the mold to be directly 3D printed.

Simulation and motion planning A variety of methods have been presented for simulating deformable objects and shells based, for example, on nonlinear finite elements [Sifakis and Barbic 2012], discrete elastic shells [Grinspun et al. 2003], mass-spring systems, co-rotated linear finite elements [Müller et al. 2008], or position-based dynamics [Müller et al. 2007; Bender et al. 2013]. The chosen method is usually the result of a careful balance between required accuracy, speed, and robustness.

Solving the inverse problem, i.e., what forces or displacements are required to achieve a desired goal position, is highly challenging. Methods were proposed for motion planning for robotic manipulation and the assembly planning of deformable objects, such as ropes and wires [Saha and Isto 2006], sheets, and elastic volumes [Jiménez 2012]. These strategies are often defined for relatively simple polyhedral environments and do not extend directly to our case, in which we are facing the removal process of a tightly fitting elastic shell from an object with complex geometric features and topology.

While in general important, we consider the simulation of the flow of material during the casting process [Grande et al. 2007] out of the scope of this paper and restrict ourselves to low-viscous resins.

Shape decomposition and approximation for fabrication In the broader context, numerous algorithms for shape decomposition and fabrication have been proposed in computer graphics. For example, recent work suggested strategies for partitioning models into smaller parts, relevant for packing large objects into 3D print volumes [Luo et al. 2012; Vanek et al. 2014; Chen et al. 2015; Yao et al. 2015].

To approximate the surface with a small number of planar polygonal primitives, Chen et al. [2013] iteratively assigned mesh faces to planar segments. This results in a closed intersection-free mesh, which can be augmented with internal connectors and fabricated. Relaxing the constraint to developable patches, D-Charts [Julius et al. 2005] provide a quasi-developable mesh segmentation, while Tang et al. [2016] and Solomon et al. [2012] provided interactive modeling systems for developable surfaces that can be used for model fabrication from sheets of material. Skouras et al. [2014] included the effects of air pressure to design inflatable structures.

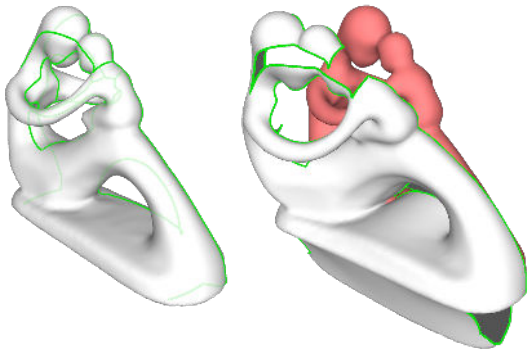

Figure 3: *A cut layout X (in green) characterizes a thin flexible mold M (white), determining how it can be opened and detached from the object surface S (in red).*

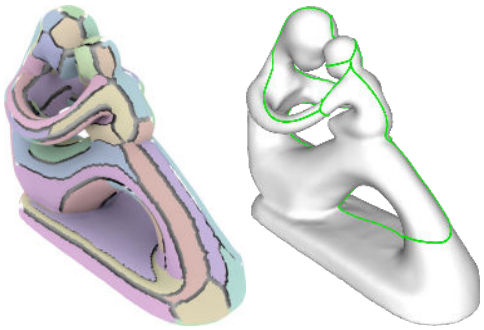

Figure 4: *Left: a cut layout obtained using the approach in [Cohen-Steiner et al. 2004]. Right: a cut layout that opens the mold to a disk is not sufficient to ensure its full removal because of the high deformation induced by the extraction process.*

Starting from a surface model, Hildebrand et al. [2012] and Schwartzburg and Pauly [2013] computed interlocking planar elements that resemble an illustrative approximation of the desired shape. Cignoni et al. [2014] extended this concept and placed the elements driven by a feature-aligned input cross-field [Ray et al. 2009; Bommes et al. 2009], a mathematical entity that catches the overall structure and curvature of the object. Inspired by this concept, we compute potential candidates for cuts along smooth polylines that emerge from a field-aligned patch layout decomposition of the initial mesh [Tarini et al. 2011; Campen et al. 2012; Myles et al. 2014; Razafindrazaka et al. 2015].

3 Designing FlexMolds

Given an object represented by a manifold, watertight, triangulated surface S, we target the problem of the automatic design of a flexible thin mold M that can be used for liquid casting. With respect to the more commonly faced problem of rigid casting, allowing the deformation of the mold makes it possible to remove more complex *non-height-field* shapes without requiring intricate fine-grained mold decomposition or the modification of the original object shape. Our flexible molds are usually very thin (in the order of 2–3 mm) and can be composed of a few pieces or even a single shell that can be unwrapped/detached from the cast object, exploiting the elasticity of the mold material.

Under these assumptions, the main characterizing feature of a mold for a surface S is the cut layout X over S that determines how the mold M_X (which is the mold M, equipped with the cut X) decomposes and opens into one or more patches p_i. Figure 3 illustrates these concepts, showing a cut layout X (in green) of a flexible mold M (white) and shows how X rules how M can be opened and detached from the surface S of the cast object (in red). The other geometric characteristics of the mold (e.g., its thickness and the actual 3D profile of the cut) can be inferred starting from X and S and are discussed in Section 4.

To understand if a cut layout X can generate a feasible mold M_X, we test if M_X can be successfully removed from the cast surface S without suffering high strains. This requires solving the following two problems:

Extraction movement Given a flexible mold M wrapped around an arbitrarily complex object S, we want to find a temporal force field $F_M(t)$ that, once applied to M, detaches it from S in a finite time and in a sound way. We postulate that the complete solution to this problem, given the similarity with the computational hardness of motion planning [Hwang and Ahuja 1992], probably lies in the realm of intractable problems, and we propose a practical heuristic for that (Section 3.1.4).

Strain evaluation Assuming we know how to drive the removal of a mold, we want to evaluate how this process affects the mold itself in terms of the strain suffered during the extraction: we have to ensure that for multiple uses of the same mold, it will not break when the object is extracted (Section 3.1.3).

These two interconnected problems are specific to the case of flexible molds and, to our knowledge, have never been faced before. In this paper, we define that a cut layout X is *feasible* if, solving the two problems above, we find that the mold M_X can be removed from S and during this process the strain is below a certain threshold.

One very important characteristic of flexible molds is that their thickness can be considered negligible in many parts of the feasibility evaluation process. For example, if a cut layout is composed of multiple pieces (as the one shown in Figure 4, left), we can assume that we can evaluate the feasibility of each piece independently, expecting that, given the small thickness of the mold and the flexibility of the material, the patches do not impede/obstruct each other.

3.1 Searching for Good Cut Layouts

As introduced in the previous section, the shape of the cut layout X is the main feature defining a mold.

To obtain a cut layout, we initially tried to decompose the mesh in multiple pieces by using a purely geometric approach that clusters portions of the surface with similar normals [Cohen-Steiner et al. 2004] (see Figure 4, left). Unfortunately, we observed that this approach, when applied to arbitrarily complex geometry, may decompose the mold into an overly high number of disconnected pieces, making the manual assembly of the mold practically infeasible. Conversely, introducing just the minimal amount of cuts to open the mesh to a topological disk [Dey 1994] may result in a configuration that does not allow for extraction due to excessive deformation (see Figure 4, right). This motivates the need for a method to adapt an initial cut layout, either by removing or introducing more cuts.

In this paper, we want to derive a cut that, while being sufficiently short and simple to generate a practically feasible mold, it also allows generating a mold that can be detached from the cast object without suffering excessive deformations.

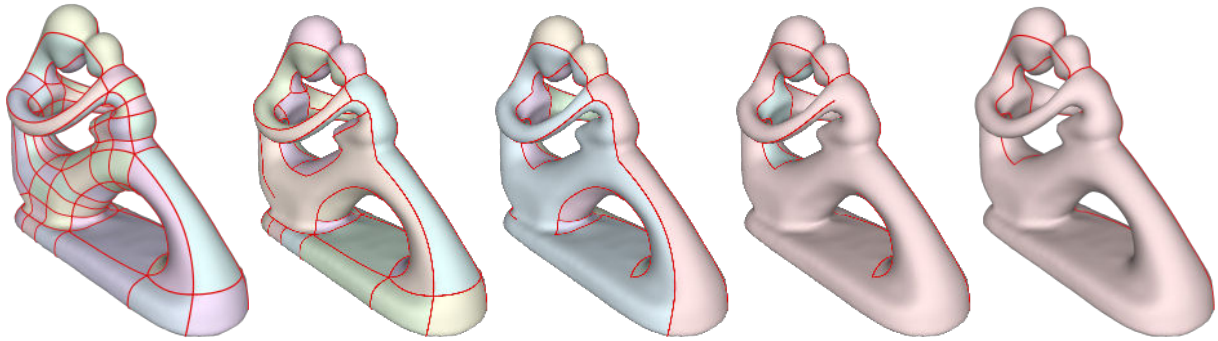

Figure 5: *The bottom-up greedy optimization process. Starting from a dense cut layout (left), generated by a patch decomposition, we iteratively perform operations that remove segments of the initial cut layout, choosing at each step the operation that requires the minimal deformation in the extraction process and stopping when this exceeds a given threshold.*

One possible way to explore the space of possible cut layouts could be a top-down approach in which cuts are grown over the surface starting from an almost closed mold. This class of approaches has been successfully used in texture parametrization to minimize the distortion [Gu et al. 2002], but we found it unsuitable to bootstrap our optimization. In fact, with those approaches in the initial steps, when the cut is short, the resulting deformations and forces required to extract the mold can be arbitrarily high. This can be a significant problem for two main reasons: first, large deformations require more accurate and costly simulations, making their use in the inner loop of an optimization process impractical; second, we experimentally found that when a high deformation is allowed, our heuristic extraction strategy may generate implausible removal movements.

For these reasons, we opted for a bottom-up approach that stays in the region of feasible cut layouts. We start, as illustrated in Figure 5, by partitioning the mesh into a set P of small, simple, disk-like patches p_i whose boundaries provide a dense cut layout X_0. For such a *dense* cut layout, it is easy to test feasibility; all patches are checked individually if they are removable with relatively low deformation. Assuming that this initial condition is fulfilled (each patch in the initial patch layout is removable), we consider the initial cut layout induced by P as composed of a set of curved polylines χ_{ij} over S that correspond to the portion of boundary shared by two patches p_i, p_j. The main idea is that we incrementally make X sparse in a greedy way by iteratively removing one polyline χ_{ij} at a time. Intuitively, at the beginning of the process, this means merging the two patches p_i, p_j along a common boundary.

3.1.1 Greedy Optimization

The optimization process is organized as a greedy loop where at each step we remove the best polyline χ from X. This removal operation can involve one or two patches p_k of the mold M_X according to whether before the removal χ connects two different connected components of M_X or not. At each step of the process, we remove the polyline χ such that the resulting mold can still be extracted with minimal deformation.

We keep all the possible removal operations arranged in a heap where the score function is the maximum deformation found during the feasibility evaluation process described in Section 3.1.3. We continue this greedy process until there is no more feasible operation, or in other words, closing the current cut X by removing a polyline $\chi \in X$ would introduce too much strain or lead to a mold for which we are not able to find a successful extraction movement.

Even if we use a very efficient dynamic simulation system for the feasibility evaluation, it is necessary to keep the number of these tests as low as possible. For this reason, we use the two following lazy update heap strategies.

Mailboxing After each extraction operation that involves the patches p_i, p_j, we should re-evaluate the heap score for all the χ lying on the boundary of p_i, p_j. To avoid all these evaluations, we mark these entries as not updated in the heap. When we extract one of these outdated entries from the heap, we re-evaluate it and eventually put it in the heap again. Such an approach is a reasonable heuristic because it is highly probable that such scores will increase. In most cases, two patches are more difficult to remove when merged than individually.

Early abort of evaluation The most time-consuming evaluations are the ones that require many steps and large deformations to extract the mold piece. For this reason, we abort the simulation when the deformation grows above a given threshold. This deformation threshold is dynamically increased during the whole greedy optimization loop and is set to be the 30 percentile of the entries in the heap. In other words, we abort all the evaluations that involve deformations higher than the 30 best values in the heap. We mark these entries in the heap, and when they pop out from the heap we proceed as above, re-evaluating and putting them in the heap again. The rationale is to delay the slow evaluations only when the process is close to the end and they are really needed.

This approach guarantees that a feasible cut layout will always be generated. It is based on two prerequisites: we require an initial feasible patch decomposition and a method to evaluate whether a candidate cut layout X allows the resulting mold to be extracted. It should be noted that, while we cannot guarantee that the approach ends up with a single patch, all the experiments we performed generated a mold composed of a single piece. From a topological point of view, note that there is no need for having a mold that is disk-like; in fact, several of the presented results are molds with multiple holes.

3.1.2 Initial Patch Decomposition

Our method requires an initial patch decomposition in which each element of the generated mold can be easily detached from the object surface. There is a vast literature about obtaining patch decompositions through mesh segmentation and partitioning (see [Shamir 2008]), and many methods can be reasonable candidates for generating it. We evaluated different approaches for this task, and the

Figure 6: *Sculpt model. Left to right: the flexible mold, the result of the cast using neutral resin, the cast with gypsum, and the cast with colored resin.*

results are discussed in Section 5. For our experiments, we chose the quad layout proposed in [Pietroni et al. 2016] that is generated starting from a cross-field over S, mostly for two intuitive reasons. First, quad patches are simple enough to guarantee feasibility but sufficiently large to not slow down the greedy optimization process. Second, the edges of this class of quad layouts are aligned with a smoothed version of the surface curvature. This characteristic generates curvature-aligned cut layouts, a feature that, intuitively, strongly resembles the natural way of decomposing a mold; see, for example, the long lines in Figure 7 that align with the overall shape of the model. These lines constitute a reasonable set of cut candidates to open the mold without too much deformation. A more detailed comparison between the different decomposition strategies is shown in Section 5.

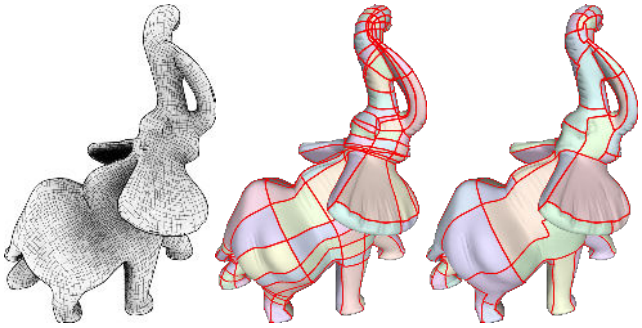

Figure 7: *An example of an input cross-field (left), the resulting quad layout (center), and the merged patch decomposition (right) used as the starting point for the greedy optimization process.*

3.1.3 Evaluation of a Patch Layout

Detaching the mold from an object induces deformations. A single, localized excessive stress in one moment of the extraction might already damage the mold. For this reason, the induced stress must be considered at the local level.

This requires a physically-based simulation of the mold and the derivation of the forces imposed by the user to extract the cast rigid object from the deformable mold surrounding it. While a full FEM simulation would be highly accurate, it comes with significant computational cost. For this reason, we opted for a dynamic simulation and a heuristic to compute a force field that drives the mold off the surface.

We based our dynamic simulation on the projective dynamics method [Bouaziz et al. 2014], which ensures a sufficient degree of accuracy and robustness at relatively low computational costs. We apply this approach on a thin shell model of our mold M_X. The contact between the mold M and the cast object S is simulated using dynamic plane constraints. We uniformly remesh each patch of the initial decomposition to ensure the quality of the triangulation and sufficient degrees of freedom for the physical simulation.

For each step of the extraction simulation, we evaluate the *current* deformation of each triangle. We then record the maximum deformation, defined as the principal stretch of the plane strain, i.e., the maximum singular value obtained from the polar decomposition of the in-plane deformation gradient. Our measure is based on the St. Venant maximum principal strain theory, which assumes that yield occurs when the maximum principal strain reaches the strain corresponding to the yield point during a simple tensile test. However, as we know the deformation field, other measures could be easily employed as well. The goodness of a cut is quantified as being inversely proportional to the maximum deformation among all triangles during all the steps of the extraction process.

Figure 8 shows some steps of one of these extraction simulations, where the color indicates, for the intermediate four steps on the left, the current deformation of each triangle. The last figure on the right shows the maximum deformation encountered by each triangle during the entire simulation; color coding of the deformation is done according to the maximum acceptable deformation.

3.1.4 Extraction Path

We use a dynamic system in which the extraction of a mold patch is simulated by applying a force field $F_M(t)$ to each vertex. The force field $F_M(t)$ deforms and drives the extraction from the cast model. It is defined as the sum of two fields: the *detaching forces* $F^D(S)$, a volumetric force field that pushes the mold piece away from the object surface, and the *moving forces* $F^M(P_k, t)$, an evolving field that depends on the actual shape of each mold piece P_k.

The **Detaching Force** field $F(S)$ is a volumetric static force field that is directed along the negative gradient of the distance field computed for the surface S (see Figure 9). Its main purpose is to push each piece away from the casted object. The magnitude of this force linearly decreases from a fixed maximum on the surface to zero for all the points with a distance greater than one third of the surface bounding box diagonal. For simple cases, detaching forces may be sufficient to separate the mold from the inner object. Unfortunately, for complex objects the mold can still be stuck in a position that wraps the object, even if "locally detached" from the cast. To overcome this problem, we introduced moving forces.

Moving Forces are defined by a dynamic field that depends on current shape and position of the mold: intuitively they grab the

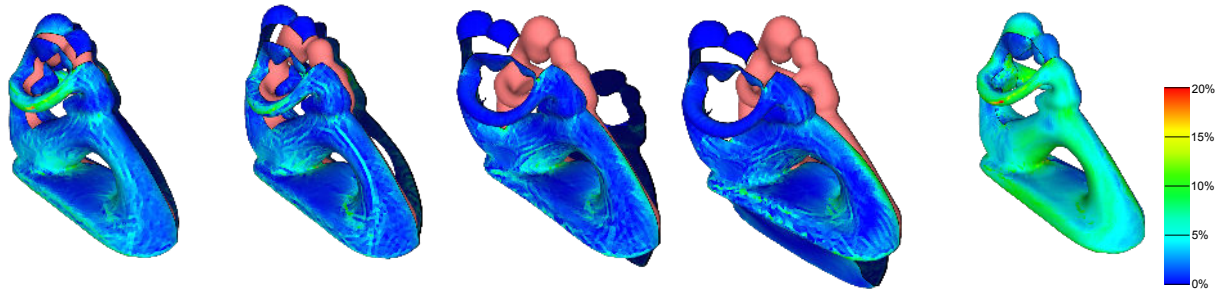

Figure 8: *The feasibility evaluation process, used in the greedy optimization to score candidate cut layouts X, attempts to extract the mold M_X and records the maximum deformation suffered during this process. Deformation is color coded according to the maximum feasible deformation threshold.*

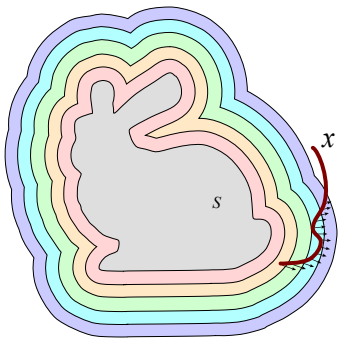

Figure 9: *An illustrative representation of detaching forces. With increasing distance from the surface the forces linearly decrease to zero.*

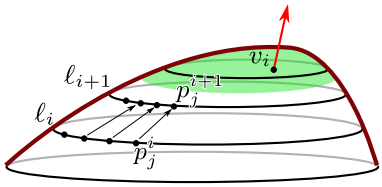

Figure 10: *Moving Forces: for a given mold piece, in the dynamic simulation, we use the transformation τ matching point pairs p_j^i, p_j^{i+1} lying on the geodesic isolines ℓ_i, ℓ_{i+1} to determine, for each point in the inner (green) part of the patch, the direction of the moving force.*

mold from the farthest region from the cuts, and pull it away, along a single direction, from the cast. For each step of the simulation, given the deformed patch P_k of the mold M at time t, we compute the isolines ℓ_i at an increasing constant geodesic distance from the border (with a step of 1/20 of the bounding box). For each pair of consecutive lines ℓ_i, ℓ_{i+1}, we regularly sample points, and for each point p_j^i on ℓ_i, we find the corresponding point p_j^{i+1} closest on ℓ_{i+1} along the geodesic gradient direction. We find the rigid transformation τ_k^t that best matches all these point pairs ($p_j^i \rightarrow p_j^{i+1}\ \forall i, j$), and we use it to compute the moving force direction as follows: we consider the set V_k of the 30 percentile vertices v_i farthest from the border and, for each vertex v_i, we use the displacement vector $\tau_k^t(v_i) - v_i$ as the moving force direction. The intensity of the force grows linearly with the geodesic distance from zero on the boundary of V_k up to a fixed maximum on the farthest point from the boundary. Figure 10 illustrates this

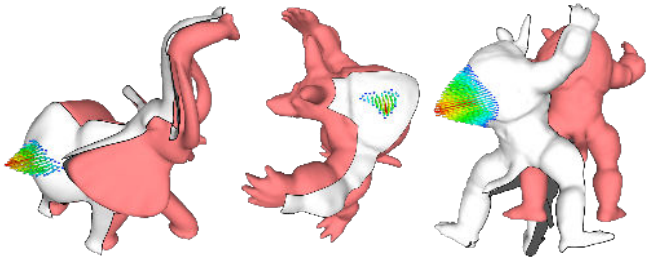

Figure 11: *Moving Forces: a few patches with the resulting moving force directions; each patch P_k^t is shown, in its deformed state, at time t. Color and length of the arrow represent force intensity. The first two examples on the left show patches that only partially cover the model, while the one on the right covers the whole model.*

concept and the resulting force direction used to generate the force $F^M(P_k, t)$ applied to one of the points of the set V_k of the inner vertices of the patch (shown in green).

Figure 11 shows a few patches in their deformed state during the dynamic simulation and the moving force direction applied on their vertices. For the sake of efficiency, assuming the mold piece will suffer only limited deformations, we compute the isolines at rest shape and we track corresponding point pairs during the simulation; then for each step of the simulation, to compute the new forces, we only have to update the transformation τ.

Both detaching and moving forces are essential to derive the extraction sequence in the general case. While detaching force depends only on the shape of the casted object and drive the mold on a global perspective, moving forces depend on the current deformation of the mold and allows the mold to escape from locally intricate situations. We experimentally observed that omitting the detaching forces significantly reduces the convergence speed of the simulation, while removing the moving forces can prevent the mold extraction, even for simple cases.

The evaluation process for a mold piece P_k starts at its rest position. We then start the simulation by applying detaching and moving forces until the mold is completely extracted from the surface S (the cast object). This condition is tested by checking if the intersection between the mold and the cast object's oriented bounding box is zero. If the number of simulation steps exceeds a given threshold, we mark the current mold as non feasible.

Our strategy is designed to guarantee that the computed extraction sequence can completely remove the cast object from its FlexMold.

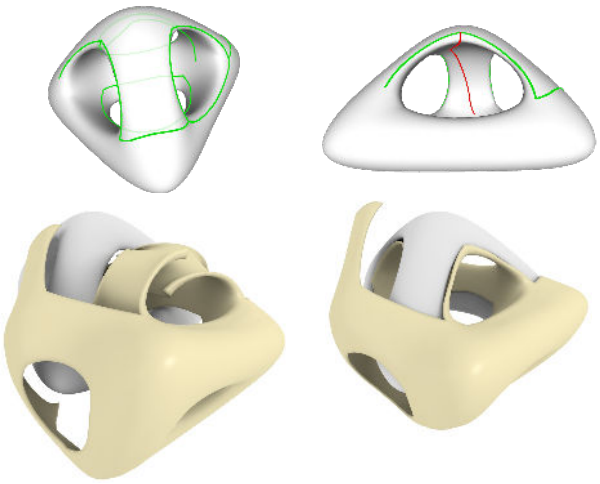

Figure 12: *Left: an internal cut is used to enable a greater deformation of the internal portion of the shell, allowing for its extraction. Right: when the internal cut (shown in red) is not present, the mold is stuck and cannot be extracted.*

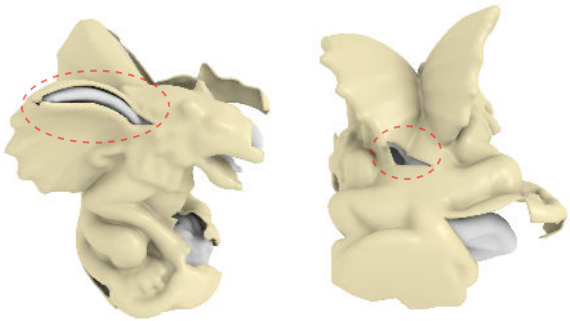

Figure 13: *Some additional cuts help to reduce the local deformations during the extraction of the mold.*

Our method is capable of leaving additional cuts to lower the deformation due to the extraction process or to allow the mold to bend and pass more easily through narrow passages during the extraction. Figure 12 shows an example where an internal cut in the model becomes mandatory for extraction. Figure 13 further illustrates the importance of small cuts to reduce the stress caused by the extraction process. To achieve this result, the simulation of the entire extraction process is required, including modeling the contact between the FlexMold and the cast object.

4 Generating Fabrication-ready FlexMolds

As result of our greedy optimization process, we have a cut layout X defined on the surface S of our starting model. However, to fabricate the final mold M_X, we need to extend it to a 3D solid. Hence, the initial surface is first inflated, then cuts are modeled as solid V-shaped prisms and finally subtracted from the volume by performing boolean operations on the volume.

4.1 Placing Air Vents

To ensure proper air escape when pouring the resin inside the mold shell, we have to accurately place small holes in the mold (see Figure 14). A bigger hole is used to pour the resin inside the mold.

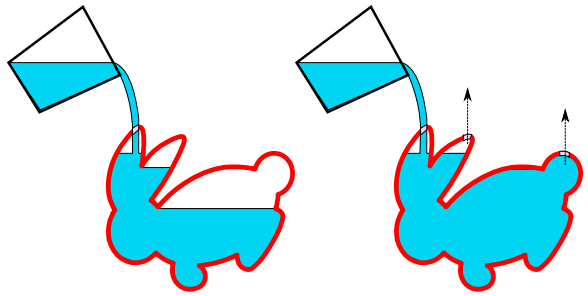

Figure 14: *Diagram illustrating the bubble trap problem.*

Theoretically, for a given mold orientation, every local maxima along the gravity direction should be equipped with a hole for letting the air escape. We select the best orientation by sampling directions on a sphere [Keinert et al. 2015] and pick the one that minimizes the number of required air vents. Due to geometrical noise and high-frequency details, this approach may still find a large number of maxima; however, some of the holes may be practically unnecessary. If we slightly shake the mold before the resin solidifies, the air may escape through nearby maxima. Formally, we can think of shaking being equivalent to varying the mold's vertical direction by an angle α. Following this intuition, we reduce the number of maxima by using a simple strategy.

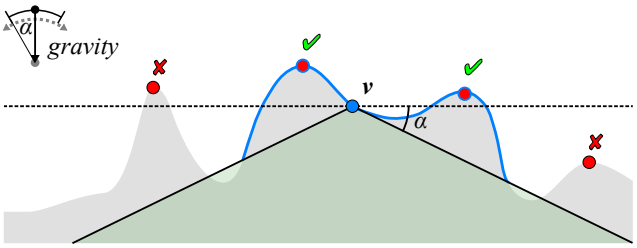

Figure 15: *Diagram illustrating the shaking cone of a vertex.*

For each vertex v, we define a shaking cone with an aperture equal to $\pi - 2\alpha$. We select the portion S_v of the mesh that is above the cone and connected to v and then associate to v all the maxima that are in S_v. Figure 15 shows a vertex v, the shaking cone, the region of S associated to v in blue, and the two maxima that can cover that vertex according to the shaking angle α. At the end of this procedure, each vertex is associated to one or more maxima. We reduce the holes by searching for the smallest set of maxima that cover all vertices. We follow a classical greedy heuristic for this minimum set cover problem. We iteratively add one maxima at a time, favoring the ones that cover the larger uncovered portion of the surface, until the entire mesh is covered. The result of this process on the dragon model is shown in Figure 16.

4.2 Using the Mold

During the casting process, the mold has to be properly sealed. Therefore, we print a sequence of small posts along each side of the cut, tie them together using a thin elastic rubber band (as shown in Figure 17), and seal the seams with silicone. While casting, the mold should be kept in the optimized orientation, otherwise the air vents may not work properly. For this purpose, we print a custom lower envelope support (see Figure 18). As a byproduct, this support helps to more evenly distribute the overall deformation under the pressure of the cast liquid.

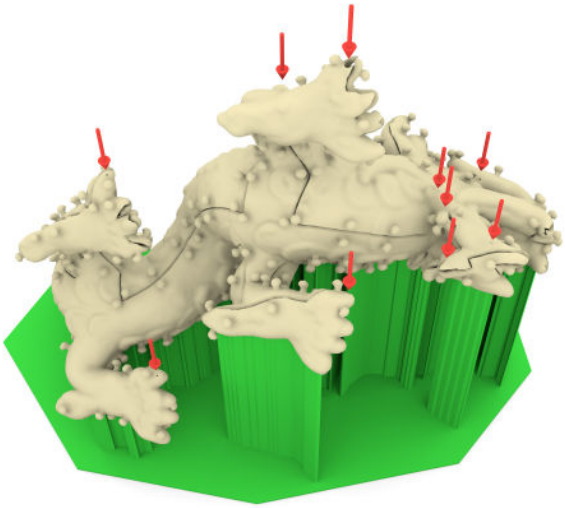

Figure 16: *The holes needed to cast the dragon model after the reduction process has been applied.*

(a) (b)

Figure 17: *(a) The mold is tied together before the resin is poured; (b) the result of the cast of the elephant model.*

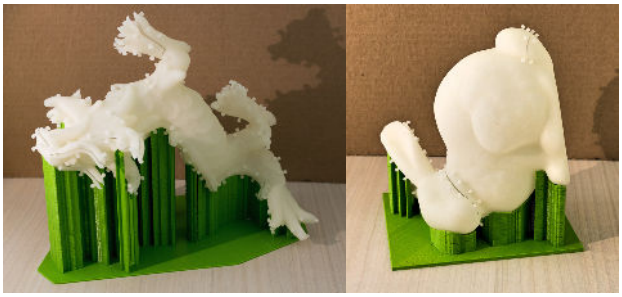

Figure 18: *To keep the models in their correct position, some simple 3D printed supports are used.*

5 Results

We evaluated our approach by generating cut layouts on a wide range of different shapes and initial partitionings. Then we actually 3D printed several of the most interesting FlexMolds and used them in practice to cast multiple copies of objects. In the following, we first validate some of our design choices and then demonstrate the effectiveness of our approach by showcasing several complex models that can be easily reproduced using our method.

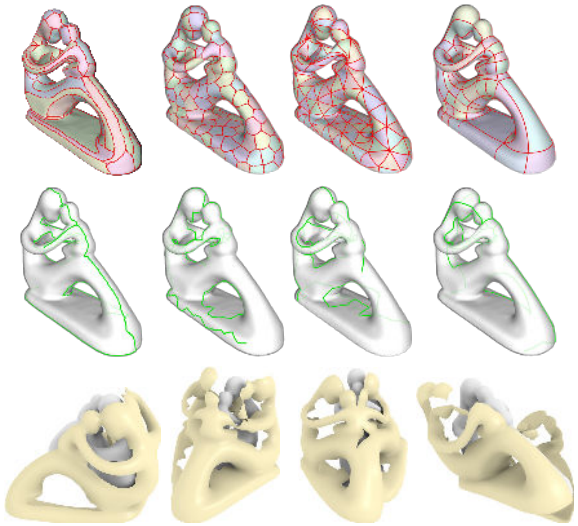

Figure 19: *Initial patch layout, the produced final cut, and a step of the extraction procedure for the approaches proposed in [Cohen-Steiner et al. 2004], [Levy 2014], [Pietroni et al. 2010], and [Pietroni et al. 2016].*

Initial patch layout Our approach for generating a proper cut layout works with any reasonable initial patch layout. Figure 19 shows the result of our algorithm when applied to different initial partitionings. The first row shows the initial partitioning we obtained by using Variational Shape Approximation [Cohen-Steiner et al. 2004], a Voronoi-based mesh partitioning with Lloyd relaxation [Levy 2014], the Almost Isometric patch layout of [Pietroni et al. 2010], and the field-coherent quad patch layout decomposition method proposed in [Pietroni et al. 2016]. The second row shows the final cut layout, while the final row shows a snapshot of the extraction sequence. To evaluate the impact of different initial partitioning on the final cut layout, we measured how the max stretch induced by the extraction grows with the shortening of the cut. We used the cut length as an intuitive measure of the qual-

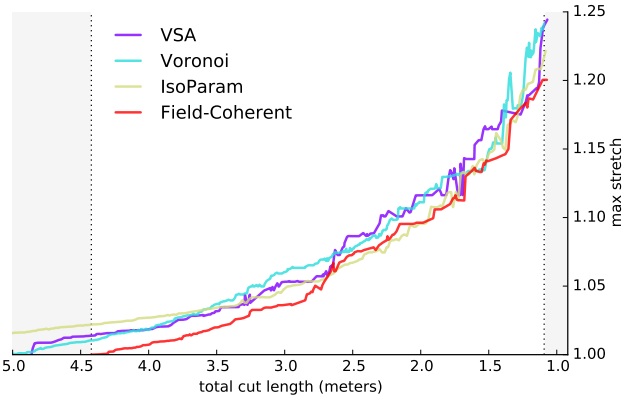

Figure 20: *Maximum stretch reached (y axis) with respect to the total cut length (x axis) for the cut optimization process, using different initial cut layouts.*

ity of a layout. If two layouts allow the extraction of their corresponding molds with the same deformation and arguably a similar effort, the shorter one must be preferred for practical reasons (easier to be closed and sealed). Figure 20 plots the relation between cut length and maximum deformation for the extraction during the optimization process. It has to be noted that the relation is not always monotone. Sometimes a change of topology may force the detaching procedure to choose a significantly different extraction path, resulting in an abrupt reduction of stretch values.

From our experiments, we found that, while any kind of reasonable initial partition works, quad-based, curvature-aligned coarse layouts [Pietroni et al. 2016] offer the best cut-length/deformation ratio most of the time. In the experiment, quad-based patch layout reduces the average deformation of VSA partitioning [Cohen-Steiner et al. 2004] by 5.1%, the Voronoi partitioning by 6%, and the Isoparametrizaion [Pietroni et al. 2010] by 3.4%. Finally curvature-aligned cuts tend to remain straight and more regular, which makes them easier to seal.

In order to speed up the optimization process, we perform a simple initial patch simplification step. We merge the patches where the average normal does not differ more than a given threshold ($\pi/16$), using a VSA-style [Cohen-Steiner et al. 2004] approach (see Figure 7).

Simple geometries We tested our method with several simple geometries, evaluating if the proposed approach produces reasonable cut layouts. As shown in Figure 21, cuts were left at locations where one would expect them intuitively. We can notice the slit at the cube edge, which allows the deformation of the top *hinge part* to be lowered during the extraction.

Fabricated examples We fabricated several models that are widely used in the Computer Graphics community to test our method. We used laser sintering to 3D print the flexible molds with thermoplastic polyurethane (TPU), a commercially available, flexible, and resistant material. The printing process we used requires a tolerance of 0.3 mm as a gap for the cuts; narrower gaps get fused. Our V-shaped cuts merged at the tip, but we were able to easily open them manually.

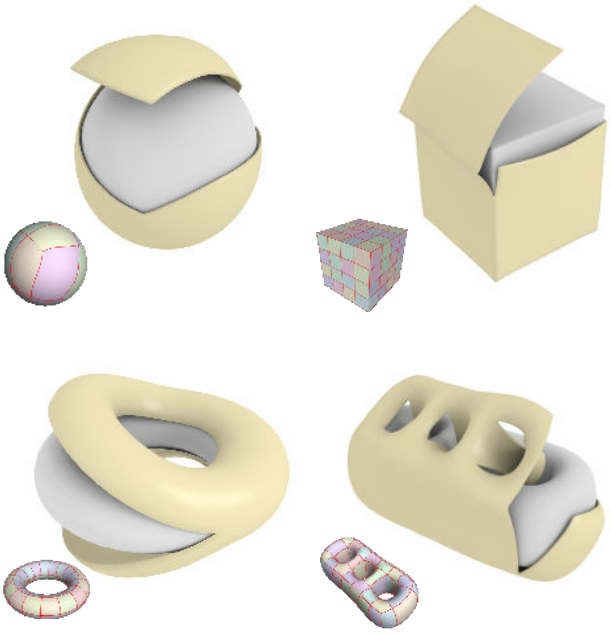

Figure 21: *Some casts obtained for basic shapes.*

The first model we tested is the sculpt, as shown in Figure 6. This model is topologically challenging, requiring several cuts. However, due to the flexibility of the mold shell, we were able to fabricate the replicas with a single-piece mold. In comparison, methods based on rigid molds would require either a highly complex multi-piece mold or alternatively several cast operations to create pieces of the object that would then be assembled in a post-process. Instead, we are able to achieve reproduction in one single cast operation with a one-piece mold. We performed multiple casts with two different types of resins and gypsum. We did not post-process any of our models except by removing excess resin from the seams. All our models were produced by a novice user who had never performed any castings, demonstrating the easy applicability of our approach. Tiny air bubbles trapped in the resin could have been removed by performing vacuum degassing before casting, a step that we ignored to keep the process as simple and lightweight as possible.

After this first experiment, we cast replicas of more complex models (see Figures 22 and 23). All the molds are composed of a single connected component. As shown by the armadillo model, we successfully capture the high frequencies of the surface detail and, as shown in the accompanying video, we can extract the object without damaging the surface details or the mold.

Finally, we pushed the proposed method to the limits by casting complex shapes with thin protruding features and high-frequency details. We successfully reproduced the dragon (see Figure 24) and the elephant (Figure 17) models that have thin tubular features (horns and fangs) and flat features (elephant ears). The elephant model required almost two liters of resin.

The numerical results of the patch decomposition of our algorithm are shown in Table 1. We successfully obtained one single patch for each model we processed. Processing time is provided for the algorithm both with and without the lazy update optimization strategy (see Section 3.1.1). As input decomposition we used [Pietroni et al. 2016] for the sculpt, fertility, bunny, bimba, and elephant models, and [Usai et al. 2015] for the dragon and armadillo models.

Table 1: *Models on which we have tested our approach.*

Mesh	start patches	reduced patches	final patches	air vents	# faces	time (minutes) w/ opt.	time (minutes) w/o opt.
sculpt	78	78	1	1	11k	15	21
bunny	138	138	1	2	31k	12	57
fertility	128	128	1	1	20k	23	91
armadillo	198	151	1	3	21k	12	67
bimba	544	151	1	3	27k	27	39
elephant	445	151	1	4	17k	33	73
dragon	604	151	1	10	12k	3	11

6 Conclusion

We have presented FlexMolds, a method for computing flexible molds that allow the reproduction of complex shapes. The core of our system is an automatic method for generating feasible cut layouts for molds, building on a physics-based approach for simulating and evaluating the parting of the mold from the cast object. Our tests have shown that our method can handle geometrically difficult cases, such as models with high genus and intricate surface details. The flexible molds can be easily fabricated using 3D printing technology and have proven to be robust enough for multiple uses. We

Figure 22: *Molds and casts obtained for the bunny, the fertility, and the armadillo models.*

Figure 23: *The result of the cast of the bimba model.*

believe our system provides an exciting new way of bringing virtual models to the real world as it empowers common users to cast objects with a wide range of resins and provides an attractive option, filling a gap between mass production and direct 3D printing. As demonstrated in the results section, we validated our method by successfully reproducing several of the most popular 3D models in the Computer Graphics community.

Limitations and Future Work Our method has some limitations and exciting opportunities for future work remain. Currently, closing the mold watertight requires sealing the seams with a removable silicone. We believe this problem could be solved by designing a more sophisticated interlocking mechanism, similar, for example, to the click-lock mechanism used for flooring.

We also checked the robustness of our method when applied to models with extremely complex topology. We successfully generated a mold for the heptoroid model (Figure 25, top), which is composed of a single piece and allows for extraction. Unfortunately, the resulting cut is extremely long and complex, which makes it difficult to seal and practically unfeasible for casting. Unavoidably, models with high genus will generate complex cuts. The hollowed sphere model shown in Figure 25 (bottom) generates a FlexMold composed of a large piece for the external part and multiple small pieces for the internal part. As the internal portion of the mold must pass through one of the small holes to be extracted, aggregating pieces makes this process more difficult or even impossible. Hence, our system correctly leaves the internal region decomposed in multiple small patches. Overall, the behavior of our algorithm is justified even for these complex examples; however, in those cases, the resulting molds are practically useless.

A reasonable strategy to overcome these limitations may consist in decomposing the initial pieces in multiple partitions that can be cast singularly and merged together once fabricated. For example, the hollowed sphere can be split in two pieces along the great circle, then each piece can be easily manufactured with our method.

Currently, we do not explicitly take the static stability of the shell into account. While in our experiments for all of our models the shell has proven to be sufficiently stable when filled with resin, large pressure could lead to deformations. One could counterbalance these effects by either taking the pressure into account and computing an optimized rest shape, similar to [Skouras et al. 2012], or creating a mold with spatially varying thickness to provide additional stability if required. Finally, currently all our models are solidly filled. For future work, it will be interesting to provide strategies to create hollow shapes, making the objects more lightweight and saving material.

Acknowledgements

The armadillo, bunny and dragon models are courtesy of the Stanford 3D Scanning Repository. The bimba, fertility and elephant models are courtesy of the AIM@SHAPE Shape Repository. This

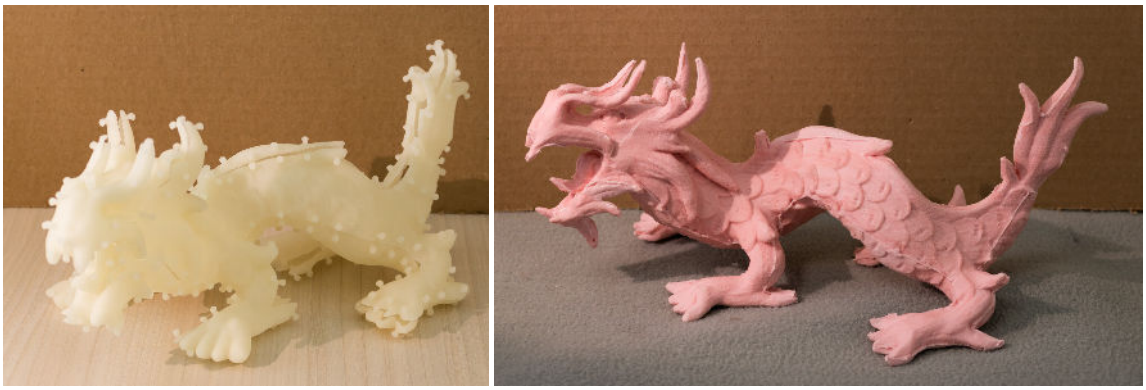

Figure 24: *The result of the cast of the dragon model.*

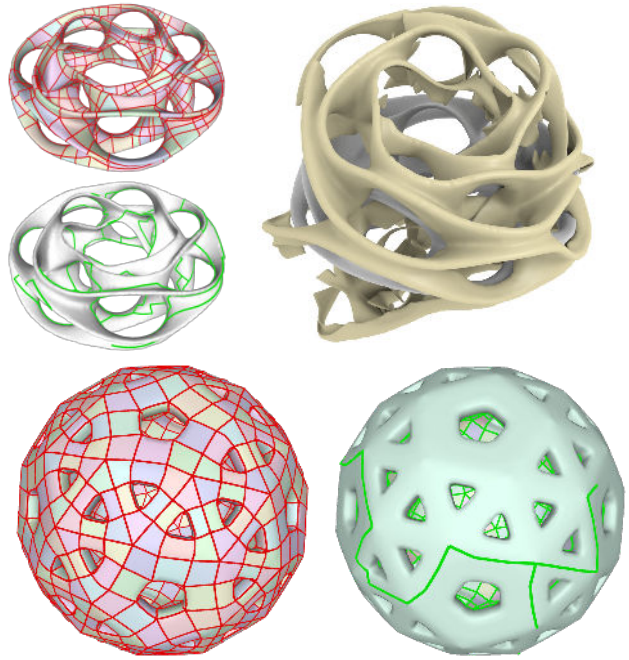

Figure 25: *Deriving the cut for two topologically complex examples, the heptoroid model (top) and the hollowed sphere (bottom).*

project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645599.

References

BENDER, J., MÜLLER, M., OTADUY, M. A., AND TESCHNER, M. 2013. Position-based methods for the simulation of solid objects in computer graphics. *EUROGRAPHICS 2013 State of the Art Reports*.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (July), 77:1–77:10.

BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG) 33*, 4, 154.

BRUCKNER, T., OAT, Z., AND PROCOPIO, R. 2010. *Pop sculpture*. Watson-Guptill Publications.

CAMPEN, M., BOMMES, D., AND KOBBELT, L. 2012. Dual loops meshing: Quality quad layouts on manifolds. *ACM Trans. Graph. 31*, 4 (July), 110:1–110:11.

CHAKRABORTY, P., AND REDDY, N. V. 2009. Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds. *journal of materials processing technology 209*, 5, 2464–2476.

CHEN, D., SITTHI-AMORN, P., LAN, J. T., AND MATUSIK, W. 2013. Computing and fabricating multiplanar models. In *Computer Graphics Forum*, vol. 32, Wiley Online Library, 305–315.

CHEN, X., ZHANG, H., LIN, J., HU, R., LU, L., HUANG, Q., BENES, B., COHEN-OR, D., AND CHEN, B. 2015. Dapper: decompose-and-pack for 3d printing. *ACM Transactions on Graphics (TOG) 34*, 6, 213.

CIGNONI, P., PIETRONI, N., MALOMO, L., AND ROBERTO, S. 2014. Field aligned mesh joinery. *ACM Transacion on Graphics. 33*, 1, art.11–1..12.

COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. *ACM Trans. Graph. 23*, 3 (Aug.), 905–914.

DE GARMO, E. P., BLACK, J. T., AND KOHSER, R. A. 2011. *DeGarmo's materials and processes in manufacturing*. John Wiley & Sons.

DEY, T. K. 1994. A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection. In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, ACM, New York, NY, USA, SCG '94, 277–284.

GRANDE, M. A., PORTA, L., AND TIBERTO, D. 2007. Computer simulation of the investment casting process: Widening of the filling step. In *The Santa Fe Symposium on Jewelry Manufacturing Technology 2007*.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, 62–67.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. *ACM Trans. Graph. 21*, 3 (July), 355–361.

HERHOLZ, P., MATUSIK, W., AND ALEXA, M. 2015. Approximating Free-form Geometry with Height Fields for Manufactur-

ing. *Computer Graphics Forum (Proc. of Eurographics) 34*, 2, 239–251.

HILDEBRAND, K., BICKEL, B., AND ALEXA, M. 2012. crdbrd: Shape fabrication by sliding planar slices. *Computer Graphics Forum (Eurographics 2012) 31*.

HU, R., LI, H., ZHANG, H., AND COHEN-OR, D. 2014. Approximate pyramidal shape decomposition. *ACM Trans. Graph. 33*, 6 (Nov.), 213:1–213:12.

HWANG, Y. K., AND AHUJA, N. 1992. Gross motion planning—a survey. *ACM Comput. Surv. 24*, 3 (Sept.), 219–291.

JIMÉNEZ, P. 2012. Survey on model-based manipulation planning of deformable objects. *Robotics and computer-integrated manufacturing 28*, 2, 154–163.

JULIUS, D., KRAEVOY, V., AND SHEFFER, A. 2005. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum*, vol. 24, Wiley Online Library, 581–590.

KEINERT, B., INNMANN, M., SÄNGER, M., AND STAMMINGER, M. 2015. Spherical fibonacci mapping. *ACM Trans. Graph. 34*, 6 (Oct.), 193:1–193:7.

LIN, A. C., AND QUANG, N. H. 2014. Automatic generation of mold-piece regions and parting curves for complex cad models in multi-piece mold design. *Computer-Aided Design 57*, 15–28.

LIU, L., WANG, C., SHAMIR, A., AND WHITING, E. 2014. 3d printing oriented design: geometry and optimization. In *SIGGRAPH Asia 2014 Courses*, ACM, 1.

LUO, L., BARAN, I., RUSINKIEWICZ, S., AND MATUSIK, W. 2012. Chopper: partitioning models into 3d-printable parts. *ACM Trans. Graph. 31*, 6, 129.

LEVY, B. 2014. Restricted voronoi diagrams for (re)-meshing surfaces and volumes.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2, 109–118.

MÜLLER, M., STAM, J., JAMES, D., AND THÜREY, N. 2008. Real time physics: class notes. In *ACM SIGGRAPH 2008 classes*, ACM, 88.

MYLES, A., PIETRONI, N., AND ZORIN, D. 2014. Robust field-aligned global parametrization. *ACM Trans. on Graphics - Siggraph 2014 33*, 4, Article No. 135.

PIETRONI, N., TARINI, M., AND CIGNONI, P. 2010. Almost isometric mesh parameterization through abstract domains. *IEEE Transaction on Visualization and Computer Graphics 16*, 4 (July/August), 621–635.

PIETRONI, N., PUPPO, E., MARCIAS, G., SCOPIGNO, R., AND CIGNONI, P. 2016. Tracing field-coherent quad layouts. *Computer Graphics Forum (Proceedings of Pacific Graphics 2016)*. In Press.

RAY, N., VALLET, B., ALONSO, L., AND LEVY, B. 2009. Geometry-aware direction field processing. *ACM Trans. Graph. 29*, 1 (Dec.), 1:1–1:11.

RAZAFINDRAZAKA, F. H., REITEBUCH, U., AND POLTHIER, K. 2015. Perfect matching quad layouts for manifold meshes. *Computer Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing) 34*, 5.

SAHA, M., AND ISTO, P. 2006. Motion planning for robotic manipulation of deformable linear objects. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, IEEE, 2478–2484.

SCHÜLLER, C., PANOZZO, D., GRUNDHÖFER, A., ZIMMER, H., SORKINE, E., AND SORKINE-HORNUNG, O. 2016. Computational thermoforming. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 35*, 4.

SCHWARTZBURG, Y., AND PAULY, M. 2013. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum (Proceedings of Eurographics 2013) 32*, 2, 317–326.

SHAMIR, A. 2008. A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6, 1539–1556.

SIFAKIS, E., AND BARBIC, J. 2012. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, ACM, 20.

SINGH, R. 2010. Three dimensional printing for casting applications: A state of art review and future perspectives. In *Advanced Materials Research*, vol. 83, Trans Tech Publ, 342–349.

SKOURAS, M., THOMASZEWSKI, B., BICKEL, B., AND GROSS, M. 2012. Computational design of rubber balloons. *Computer Graphics Forum 31*, 2pt4.

SKOURAS, M., THOMASZEWSKI, B., KAUFMANN, P., GARG, A., BICKEL, B., GRINSPUN, E., AND GROSS, M. 2014. Designing inflatable structures. *ACM Trans on Graphics (Proc. of ACM SIGGRAPH) 33*, 4.

SOLOMON, J., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2012. Flexible developable surfaces. In *Computer Graphics Forum*, vol. 31, Wiley Online Library, 1567–1576.

TANG, C., BO, P., WALLNER, J., AND POTTMANN, H. 2016. Interactive design of developable surfaces. *ACM Transactions on Graphics (TOG) 35*, 2.

TARINI, M., PUPPO, E., PANOZZO, D., PIETRONI, N., AND CIGNONI, P. 2011. Simple quad domains for field aligned mesh parametrization. *ACM Trans. Graph. 30*, 6 (Dec.), 142:1–142:12.

UMETANI, N., BICKEL, B., AND MATUSIK, W. 2015. Computational tools for 3d printing. In *ACM SIGGRAPH 2015 Courses*, ACM, 9.

USAI, F., LIVESU, M., PUPPO, E., TARINI, M., AND SCATENI, R. 2015. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Trans. Graph. 35*, 1 (Dec.), 6:1–6:13.

VANEK, J., GALICIA, J., BENES, B., MĚCH, R., CARR, N., STAVA, O., AND MILLER, G. 2014. Packmerger: A 3d print volume optimizer. In *Computer Graphics Forum*, vol. 33, Wiley Online Library, 322–332.

WANNARUMON, S. 2011. Reviews of computer-aided technologies for jewelry design and casting. *Naresuan University Engineering Journal 6*, 1, 45–56.

YAO, M., CHEN, Z., LUO, L., WANG, R., AND WANG, H. 2015. Level-set-based partitioning and packing optimization of a printable model. *ACM Transactions on Graphics (TOG) 34*, 6, 214.

ZHANG, C., ZHOU, X., AND LI, C. 2010. Feature extraction from freeform molded parts for moldability analysis. *The International Journal of Advanced Manufacturing Technology 48*, 1-4, 273–282.