EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

Approximating Free-form Geometry with Height Fields for Manufacturing

Philipp Herholz[1] Wojciech Matusik[2] Marc Alexa[1]

[1]Technische Universität Berlin [2]MIT CSAIL

Figure 1: *To fabricate a mesh, possibly acquired by laser scanning, in a molding process, we automatically decompose and deform the shape (center left). A reusable multi-piece mold for the object can be produced by CNC milling (center right). Physical copies can be fabricated by mold casting in different materials like resin or plaster (right).*

Abstract

We consider the problem of manufacturing free-form geometry with classical manufacturing techniques, such as mold casting or 3-axis milling. We determine a set of constraints that are necessary for manufacturability and then decompose and, if necessary, deform the shape to satisfy the constraints per segment. We show that many objects can be generated from a small number of (mold-)pieces if slight deformations are acceptable. We provide examples of actual molds and the resulting manufactured objects.

1. Introduction

The recent development in 3d printing has liberated digital manufacturing: today a large variety of shapes can be printed and design is in many cases no longer constrained by the manufacturing process. Nonetheless, 3d printing has still a number of problems, some of them inherent to the technology. For example the size of the production volume is severely limited for most technologies, production time is high and so is the price per unit. Another issue is the limited choice of materials, surface finishes, and structural properties. Classical manufacturing techniques such as casting, stamping, and 3-axis milling are in many aspects complementary. These technologies are applicable for a large variety of materials. Since they are targeted at mass production, cost per item and production times are generally low once a mold or die has been manufactured. On the other hand, these technologies are severely limited in terms of shapes that can be produced. For most classical fabrication techniques the shape has to be a *height field*. In 3-axis milling, for example, the drill bit has to be able to reach every point on the surface. The same is true for stamping, a technique commonly used in the automotive industry to press metal sheets into a required shape. Another example is mold casting, where two or more molds form a cavity that is filled with liquid material. After the material has hardened, the mold pieces have to be removed by a linear translation, hence, again yielding a height field constraint for individual mold pieces. There are two common approaches to respect the constraint on shapes to be a height field. Either shapes have to be designed in such a way that they constitute height fields, or they have to be divided into parts that fulfill the constraint individually. So far, specialized engineers are needed to segment a given design

into parts that can be fabricated by classical manufacturing techniques.

Our objective is to automate this process. In contrast to industrial products, we target the fabrication of free-form geometry, possibly acquired by laser scanning. Industrial parts are typically designed with manufacturability in mind and mold-makers and designers work in a loop, iteratively improving the design. This case is covered by a large body of literature in CAD/CAM [PG04, KBM06, CR09]. Our situation, on the other hand, is more involved, since we oftentimes deal with organic shapes having very intricate features and concavities. However, we are not concerned with functional parts, like gears, that have to fit perfectly into an assembly. This provides us with some additional freedom: we can deform the shape slightly in order to meet the constraints. This deformation often enables a significant reduction in the number of parts that are required while typically only leading to small visual distortion.

We also explain how global constraints on mesh segmentation for fabrication, like total seam length and constraints on the maximal build volume, can naturally be incorporated into our system. Moreover, we provide an interactive user interface, enabling users to manually specify local constraints, such as the amount of allowed deformation.

Contributions of this work include:

- A method to decompose a shape into surface patches that largely satisfy the height field constraint. Constraint violation can be controlled and even completely prevented.
- An adapted version of as-rigid-as-possible mesh deformation to remove local constraint violations.

To demonstrate the practicability of our approach, we fabricated physical mold pieces and used them to cast several input meshes. In addition, we fabricated examples using 3-axis CNC milling.

2. Related work

2.1. Mold design

Mold design is a prominent topic in the engineering literature. Naturally, these works are targeted at shapes already designed for fabrication. Therefore, the requirements and assumptions on the input are quite restrictive. Ravi and Srinivasan [RS90] give a list of decision criteria for mold design, like undercuts, flatness, and stability. Based on these criteria Chakraborty and Reddy [CR09] devise an algorithm for the construction of two-piece permanent molds with side cores. Pryadarshi and Gupta [PG04] try to find a decomposition into multi-piece molds. They consider the problem as a set cover problem. Concurrent with our analysis of set cover techniques, the method generates a large number of mold pieces for free-form surfaces (cf. Figure 4). Closely related to our approach is the work of Li et al. [LML09]. Given a casting direction, they determine an optimal parting line. Undercuts are removed by a mesh deformation technique. They consider only two-piece molds, however, and manufacturability is not guaranteed. Khadekar et al. [KBM06] employ the GPU to display undercuts by rendering the model from potential casting directions. This procedure is extremely fast and allows for a quasi exhaustive search of the space of possible directions. However, we have found this method to be unreliable for general meshes with small triangles since these may be discarded during rendering. Very recently He et al. [HLZCO14] explored the decomposition of shapes into pyramidal pieces. A shape is pyramidal if it has a flat base and the rest of the shape is a height function over that base. They consider molding and reducing the amount of support material in fused deposition printing as applications. Our approach is more general since we do not require the parts to have a flat base.

2.2. Mesh processing for fabrication

In recent years, different works in computer graphics addressed mesh processing for fabrication. Luo et al. [LBRM12] focus on the build volume constraint of 3d printers and try to find a segmentation of a model into a small number of parts such that the assembled object is structurally sound and hides seams. Our approach also allows to consider a build volume constraint and can hence also help to overcome the problem addressed in this work. Julius et al. [JKS05] segment a shape into almost developable patches to enable the fabrication from sheets of materials, for example, for sewing stuffed toys from sheets of fabric. Similarly, Chen et al. [CSaLM13] simplify a model until it consists of only a few polygons that can be fabricated with a laser cutter. Both works aim at 2-axis production devices and hence require more severe changes to the model than necessary with our approach. McCrae et al. [MSM11] and Hildebrand et al. [HBA12] decompose a shape into orthogonal planar pieces that resemble the input model. However, models generated with this approach are only abstractions of the original input while our techniques yields replicas, possibly up to a small error due to deformations. Vanek et al. [VGB*14] segment a model into pieces to save production time and support material during 3d printing by packing pieces tightly into the production volume. Our technique is complementary and saves support material by avoiding the overhangs that make support necessary in the first place. Similar to our work, mesh deformation has also been used in other works to improve fabrication properties of 3d objects. For example, Stava et al. [SVB*12] used deformation to ensure the structural stability of fabricated objects. Prevost et al. [PWLSH13] employ deformation for adjusting the shapes' center of mass such that it can stand upright without falling over.

3. Problem statement

We assume the input geometry is given as a triangle mesh $\mathcal{M} = (V, \mathcal{F})$ with vertex coordinates $V = \{\mathbf{v}_i\}$ and triangles specified by their vertex indices $\mathcal{F} = \{(i, j, k)\}$. The mesh is assumed to represent a discrete, oriented manifold with or without boundary. Our goal is to create a physical copy of the shape enclosed by $\mathcal{M}$ using classical manufacturing techniques such as 3-axis milling, casting, and stamping. All of these techniques constrain the shape to be a height field with respect to a specific *fabrication direction* $\mathbf{d}$: stamping can only produce thin height field sheets; 3-axis milling can only be used to fabricate a height field; casting techniques requires the portion of the shape associated with an individual mold piece to be a height field. In order to use these techniques we segment the model into disjoint connected height field components C_i:

$$C_i \subseteq \mathcal{F}, \quad \bigcup_i C_i = \mathcal{F}, \quad C_i \cap C_j = \emptyset. \tag{1}$$

The sub-meshes C_i can then be manufactured individually and assembled to form the input shape. In the case of multi-piece molding, the segments can be used to produce mold parts which in turn can be used to manufacture the input shape in one cast. Let $\mathbf{n}_t$ be the outward pointing surface normal of triangle $t \in C_i$ and $\mathbf{d}_i$ a potential fabrication direction. A necessary condition for C_i to be a height field with respect to $\mathbf{d}_i$ is

$$\mathbf{d}_i^\mathsf{T} \mathbf{n}_t \geq 0, \quad \forall t \in C_i. \tag{C1}$$

This is, however, not sufficient for C_i to be a height field since global overlaps can still be present (c.f. Figure 2). Lipman [Lip13] has shown that if all triangles satisfy (C1) then it is sufficient that the boundary polygon(s) of C_i, projected along $\mathbf{d}_i$, are *simple*, that is, there are no self intersections. Let P_i be the projection along $\mathbf{d}_i$, i.e., $P_i = I - \mathbf{d}_i \mathbf{d}_i^\mathsf{T}$. Then we can verify the boundary condition by

$$\overline{P_i \mathbf{v_{j_0}}, P_i \mathbf{v_{j_1}}} \cap \overline{P_i \mathbf{v_{k_0}}, P_i \mathbf{v_{k_1}}} = \emptyset. \tag{C2}$$

for all edges $(j_0, j_1), (k_0, k_1) \in \partial C_i$ of the boundary ∂C_i. See Figure 2 for an example of a surface patch that obeys (C1) but violates (C2).

In the following we consider several additional constraints that might be necessary depending on the manufacturing device or application scenario.

3.1. Shell constraint

In multi-piece molding each mold piece has to be removed from the cast after the material has cured. If face normals of a segment use the full half-space of valid normals with respect to the fabrication direction, a linear translation along this direction is the only way to remove the mold piece. The above constraints (C1) and (C2), however, cannot guarantee that it is possible to remove the mold pieces since interlocking can occur. Consider for example two adjacent pieces of a 3d-jigsaw puzzle. Even though the curved surface of each piece might fulfill (C1) and (C2) individually, the pieces can not be disassembled by a linear motion along their fabrication direction. Figure 3 illustrates such a situation. To detect interlocking of adjacent mold pieces we consider tangents of the surface orthogonal to seam edges. As shown in the Figure to the right, the tangent vector $\mathbf{t}_{j_1,j_2,j_0}$ at edge (j_1, j_2) of the triangle (j_0, j_1, j_2) is given by

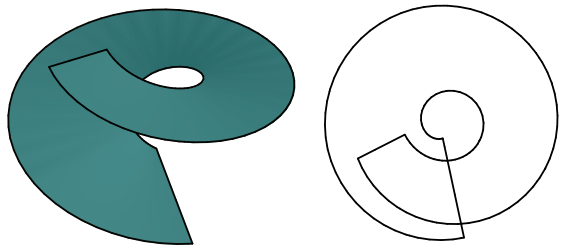


Figure 2: *Global overlaps will occur if the projected boundary polygon has self-intersections.*

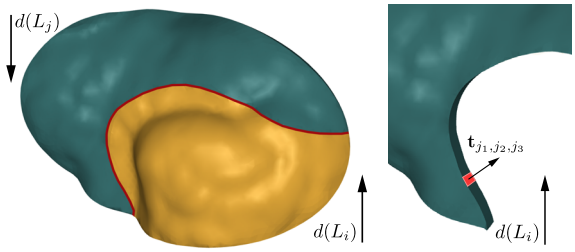
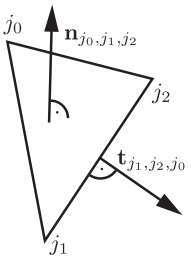


Figure 3: *Illustration of the shell constraint. The two segments can not be removed by a translation along their fabrication directions $d(L_i)$ and $d(L_j)$.*



$$\mathbf{t}_{j_1,j_2,j_0} = \frac{\mathbf{n}_{j_0,j_1,j_2} \times (\mathbf{v}_{j_1} - \mathbf{v}_{j_2})}{||\mathbf{n}_{j_0,j_1,j_2} \times (\mathbf{v}_{j_1} - \mathbf{v}_{j_2})||}. \tag{2}$$

Interlocking can be prevented by enforcing

$$\mathbf{d}_i^\mathsf{T} \mathbf{t}_{j_1,j_2,j_0} \geq 0, \quad (j_0, j_1, j_2) \in C_i \tag{C3}$$

for all seam edges (j_1, j_2). To see this, consider extruding a surface component into a shell piece along the normal at each vertex. Boundary edges are extruded parallel along the face normal, forming planar quadrangular polygons. These polygons will form the *contact surface* with neighboring shell pieces. The normal of such a boundary polygon extruded from edge (j_1, j_2) is $\mathbf{t}_{j_1,j_2,j_0}$ (Figure 3, right). To assemble the shape by moving the shell piece of component C_i towards another one in direction $\mathbf{d}_i$ the contact surface has to be completely visible from direction $\mathbf{d}_i$. This is equivalent to (C3).

3.2. Volume constraint

Most manufacturing devices have a cuboid build volume with side lengths $(dx_{max}, dy_{max}, dz_{max})$. We hence require that for every component C_i there is a bounding box with dimensions $dx \times dy \times dy$ contained in $dx_{max} \times dy_{max} \times dz_{max}$:

$$dx \leq dx_{max}, \quad dy \leq dy_{max}, \quad dz \leq dz_{max}. \tag{C4}$$

3.3. Soft constraints

The fundamental principle of our approach is to combine segmentation and deformation to find a compromise between the number of parts and the necessary deformation of the geometry. One can consider this as additional constraints, i.e. keep the number of components C_i small and minimize $d(V,V')$ under some appropriate metric d. Note, that these constraints are conflicting; allowing for more deformation will reduce the number of parts and vice versa. We employ a user set parameter to balance these objectives.

3.4. Other possible constraints

There are other constraints specific to certain manufacturing devices that do not model here. For examples, we currently do not consider the finite tool size of milling machines. Similarly, we currently do not take global assembly constraints for multi-piece molds into account. This means that while we do enforce that faces f_i are not occluded in their own segment $C_k \ni f_i$, we do not require faces to be globally *ray-accessible*. This means that a ray from f_i into direction $\mathbf{d}_k$ might hit another face of the model. Consequently, mold pieces might not be removable. We feel it is often better to consider such problems after processing the shape and find solutions with the user in the loop in an interactive system. Reasons and consequences of this design decision will be discussed further in Section 9.1.

4. Method

In this section we present our approach for solving the fabrication problem formulated in the last section. First, we uniformly sample a large number of potential fabrication directions $\mathbf{d}_i$ on the Gauss sphere. For each direction $\mathbf{d}_i$ we mark all triangles that satisfy constraint (C1) w.r.t. this direction and add all connected components of these triangles to the set $\{L_j\}$. Since there can be more than one component per direction $\mathbf{d}_i$, we use the notation $\mathbf{d}_i = d(L_j)$ to denote the direction associated with connected component L_j.

The main idea of our segmentation procedure is to cover the shape with a small number of components from $\{L_j\}$, such that all constraints are satisfied. It might appear that a solution to our segmentation problem can be obtained using approximations to the set cover problem. Set cover approximations would select few components whose union cover the surface. However, they lack any information on how these components might overlap and how they might become disconnected as a result (see Figure 4). This is impractical since each connected component of the set cover would have to be manufactured individually. Moreover, it is not straightforward to incorporate (C2) and (C3) into these algorithms. Hence, we rather employ a graph cut based approach which is described in the next section. This allows us to model the problem to obey all necessary constraints while still permitting fast approximations.

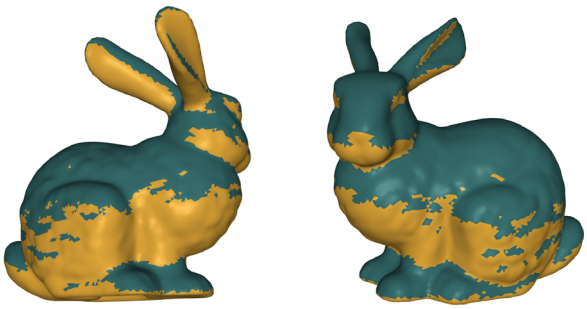


Figure 4: *A solution to the segmentation problem provided by an approximation to set cover generates too many individual components to be useful.*

4.1. Constraint relaxation

For many shapes, especially ones obtained by scanning real world objects, trying to cover the surface with the components $\{L_j\}$ does not lead to a manageable number of parts, say two to seven, even if we ignore all other constraints. Consider for example the hair structures on the model 'lion vase' (Figure 15). No matter how many directions we sample, every face set will inevitably contain many holes. We observe that a small amount of deformation is in many cases enough to significantly reduce the number of required parts. One possible approach would be to deform the shape in a preprocessing step. However, without knowledge of a specific segmentation it is hard to tell how the mesh should be deformed in order to reduce the number of components. Our strategy for minimizing the resulting deformation is to relax the constraints in the segmentation phase (Section 5) and correct for possible constraint violations in a post-processing step (Section 6).

Crucial for the success of our approach is *how* the constraints are relaxed. We process each face set L_j individually and record connected components of the complement $\mathcal{F} \setminus L_j$. For each component in this complement we compute the (unsigned) *projected area* w.r.t. $d(L_j)$, which provides us with an a priori estimate for the necessary deformation. We add a component of $\mathcal{F} \setminus L_j$ to L_j only if its projected area is below a threshold, which we denote γ. This parameter controls the deformation necessary to make the shape manufacturable. Quite intuitively, allowing more deforma-

tion usually leads to fewer parts, so γ can be used to balance the number of parts and how close the manufactured shape matches the original geometry. We demonstrate this effect in Section 8.4.

4.2. Enforcing the volume constraint

Since a subset of a face set L_k will potentially form a surface segment C_i, the spatial extent of L_k represents an upper bound on the extent of C_i. To enforce a cuboid build volume constraint we trim face sets exceeding the build volume. To maximize the area of the trimmed face set we first construct a bounding box that is aligned with the principal directions and the fabrication direction of the face set. The bounding box is voxelized and the enclosed surface area for each voxel is determined. This information is used to find the axis aligned build volume cuboid maximizing enclosed surface area. This problem can be solved efficiently by dynamic programming.

This approach is of course conservative since the segments can be trimmed in many different ways to obey the volume constraint. Another approach would be to enforce the build volume constraint in the optimization phase. Since the constraint is global in nature, this would require a more complex optimization procedure (Section 5.2).

4.3. Adaptive direction sampling

Sampling candidate directions $\mathbf{d}_i$ uniformly works well for many shapes. However, preferable directions might be missed. A preferable direction can be characterized as a direction with a connected component L_j of large surface area. Since sampling and processing a very large number of directions is computationally demanding, we propose an adaptive sampling strategy. Starting with a coarse sampling, we compute connected components of face sets for each direction. Based on the maximal surface area per direction we refine the sampling adaptively with respect to surface area. Figure 5 illustrates maximal surface area per direction color coded on the sphere. Sampling positions are marked as black points, local maxima w.r.t. surface area as red points.

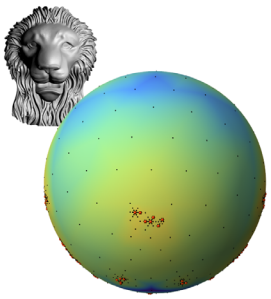


Figure 5: *Area per sampled direction.*

5. Segmentation

We consider mesh segmentation as a discrete labeling problem. Connected components of faces carrying the same label will form *segments* C_i which can be manufactured individually. Our input are the face sets $\{L_j\}$ formed by connected components associated with any of the sampled directions. Every triangle $f \in \mathcal{F}$ is contained in many of these sets. The set $\mathcal{L}(f)$ if formed by indizes of face sets that contain f:

$$\mathcal{L}(f) = \{j \mid f \in L_j\}. \tag{3}$$

Our objective is to choose for every triangle f one of the labels from $\mathcal{L}(f)$. Every labeling $l_i : f_i \to \{1,2,\cdots\}$ such that $l_i \in \mathcal{L}(f_i)$, induces a valid segmentation with respect to (C1). Note that enforcing (C2) requires checking pairs of seam edges which is a global problem, complicating the search for a valid labeling (see Section 5.2). Therefore, we restrict the space of valid labelings even further. For each face set L_j we record all faces that are either flipped or occluded by other faces from L_j in direction $d(L_j)$. This can be achieved efficiently by projecting L_j onto the plane orthogonal to $d(L_j)$ and recording triangle intersections in 2d. If a face $f_i \in L_j$ is occluded or flipped, the labeling $l_i = j$ is only allowed if all adjacent faces to f_i also get the label j. In other words, none of the three triangle edges is allowed to become a seam edge of the segmentation. This is a rather conservative approach since the occluder is part of L_j but might not be part of the segment $C_k \ni f_i$. Therefore the edges of f_i might not be occluded from another seam edge in C_k at all. Checking for (C3) is a simple local task. Suppose (j_0, j_1) is a seam edge with two adjacent faces (j_0, j_1, j_2) and (j_1, j_0, j_3) carying labels k and l respectively. This labeling is only valid if

$$d(L_l)^\mathsf{T}\mathbf{t}_{j_0,j_1,j_2} \geq 0 \quad \text{and} \quad d(L_k)^\mathsf{T}\mathbf{t}_{j_1,j_0,j_3} \geq 0. \tag{4}$$

5.1. Energy minimization

We formulate the problem of finding a valid segmentation as an energy minimization problem defined over the edges and nodes of the face graph $G_\mathcal{F}$. The faces constitute the nodes of this graph and graph edges represent edges shared by two faces. Our energy is designed such that a multi-label graph cut algorithm [BVZ01] can efficiently find an approximate minimizer. For any labeling $(l_i)_{i\in\mathcal{F}}$ the energy is given by

$$\mathcal{E} = \sum_{i\in\mathcal{F}} E_i(l_i) + \sum_{(i,j)\in\mathcal{F}\times\mathcal{F}} E_{i,j}(l_i,l_j) + \lambda|\{l_i, i \in \mathcal{F}\}| \tag{5}$$

where, by abuse of notation, $\mathcal{F}$ is used as an index set over all faces of $\mathcal{M}$. The unary term $E_i(l_i)$ represents the cost associated with setting the label for face f_i to l_i. We use this term to enforce that the label for f_i is valid:

$$E_i(l_i) = \begin{cases} 0 & \text{if } l_i \in \mathcal{L}(f_i) \\ \infty & \text{otherwise.} \end{cases} \tag{6}$$

Minimizing E_i alone will yield a valid labeling with respect to (C1). However, the boundary might not have an injective projection (C2) and seam edges might violate (C3). Moreover, result in the use of many different labels, leading to a large number of segments C_i. The pairwise term $E_{i,j}$ in Eq. (5) helps to alleviate both problems. We use the follow-

ing definition:

$$E_{i,j}(l_i,l_j) = \begin{cases} 0 & \text{if } l_i = l_j, \\ \infty & \text{if } f_i \text{ or } f_j \text{ violates (C1)}, \\ \infty & \text{if } f_i \text{ or } f_j \text{ is occluded in } L_{l_i} \text{ or } L_{l_j}, \\ \infty & \text{if the edge } f_i \cap f_j \text{ violates (C3)}, \\ |f_i \cap f_j| & \text{otherwise.} \end{cases} \quad (7)$$

In the context of Markov random fields, $E_{i,j}$ is frequently called *smoothness term*, since it encourages a consistent choice of labels for neighboring faces. The cost of an inconsistent labeling for two faces connected by an edge is the length of that edge. Edges violating (C2) or (C3) with respect to a specific labeling of their adjacent faces lead to an infinite energy, therby enforcing the constraints. The value of $E_{i,j}$ for non-adjacent faces f_i and f_j is always zero, the energy is therefore local.

The sum of $E_{i,j}$ in Eq. (5), running over all edges in the mesh, measures the total seam length, if it has a finite value. Note that minimizing the total seam length also encourages the use of a small number of labels. By setting λ to a nonzero value it is, however, possible to explicitly punish the use of many different labels.

5.2. The α-expansion algorithm

We give a brief overview of the optimization algorithm used in our implementation. The α-expansion algorithm [BVZ01] finds an approximate minimizer of the energy $\mathcal{E}$ by iteratively computing graph cuts. The graph used by the algorithm is an augmented version of the input graph $G_{\mathcal{F}}$, to unify the treatment of unary and binary energies. Starting from an initial labeling the algorithm proceeds by focusing at one label α at a time. A graph cut answers for every node the question whether to retain its label or to switch to α. This process, called α-expansion, is repeated by cycling through all labels until convergence.

The quality of the result is influenced by two factors: the initial labeling and the order in which labels are processed. While the solution is largely independent of the initial labeling, for which we use a random assignment, the order of the labels has a great influence on the approximation quality of the solution. After assessing several strategies for generating 'good' label orders to speed up convergence, we found that running the algorithm several times for completely random label orders gives consistently the best results.

Minimizing labeling problems with general unary and binary energy terms is NP-hard. To find approximate minimizers of these energies, the α-expansion algorithm relies on graph cuts. This approach comes with a limitation: the binary energy term $E_{i,j}$ term has to be a semi-metric. Therefore, for three labels α, β, γ the following triangle inequality has to hold:

$$E_{i,j}(\alpha,\alpha) + E_{i,j}(\beta,\gamma) \leq E_{i,j}(\alpha,\gamma) + E_{i,j}(\beta,\alpha). \quad (8)$$

Furthermore, the energy has to be positive and symmetric with respect to its arguments.

The condition in Eq. 8 prevents us from incorporating global constraints, like limited build volume, directly into the energy minimization. If we want, for example, to prevent two non-adjacent faces to end up in the same segment because they occlude each other, we could try to add a new edge between these faces in the graph and set $E_{i,j}(l_i,l_j) = \infty$ for $l_i = l_j$ and 0 otherwise. Unfortunately, this would violate the condition stated in Eq. 8. This is the reason why we chose to forbid seam edges that are occluded by a face from the label face set, even though the occluder might not be part of the actual surface patch. For the same reason the maximum build volume constraint is enforced by trimming label face sets L_j.

5.3. User interface

We provide a simple interactive user interface to help the user guide the segmentation process, if desired. The user might want to prevent seams to cut through specific regions of the surface or considers deformation unacceptable for salient features. Our interface enables the user to mark faces with a brush tool as 'no seam region' or 'no deformation region'. These additional constraints are enforced by modifying the unary and binary energy terms. For faces F_i marked as 'no seam region' the term $E_{i,j}$ is set to ∞ for all neighboring faces F_j. Faces that are not supposed to be deformed are only allowed to be given a label if they obey the hard constraint (C1) with respect to the label direction. The user defined constraints have to be consistent; disallowing seams for the whole mesh will not yield a valid result with finite energy $\mathcal{E}$. We therefore check if the constraints are feasible and give a warning to the user if this is not the case. After

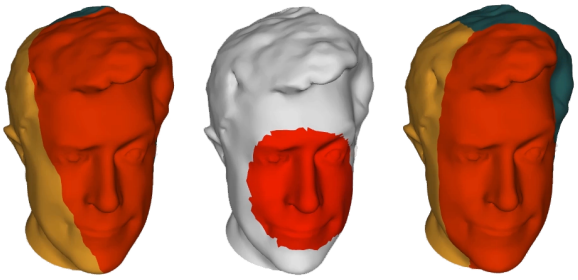


Figure 6: *Our algorithm optionally supports user interaction. The user might not be satisfied with the initial segmentation (left). To prevent seams from cutting through the face, the user marks the region as 'no seam region' (center). Running more α-Expansion iterations with the modified energy yields a result that satisfies the user constraints (right).*

the user made changes to the constraints either a new run of the α-expansion algorithm is performed or additional α expansion iterations, based on the current labeling. We refer to the accompanying video for a demonstration.

6. Deformation

After the mesh has been segmented, some faces might violate constraint (C1), while all component boundaries satisfy (C2) and (C3) (if desired). Figure 7(a) shows faces violating (C1), color coded for a particular component and direction. Our goal is to deform the mesh as little as possible in order to satisfy the constraint (C1) without degrading the overall appearance of the mesh or violate the boundary constraints (C2) and (C3).

A simple way to satisfy (C2) and (C3) is to fix the boundaries of each component. This also lets us treat each component independently. In the following we consider a single component C_i.

6.1. Energy formulation

We face the task of fixing the geometry to comply with (C1) as an energy minimization problem. To this end we formulate a constrained version of the well known as-rigid-as-possible (ARAP) surface deformation energy [SA07].

Let the original geometry of component C_i be represented by the vertex set V and the deformed geometry by V'. We can describe the transformation of triangle $t = (j_0, j_1, j_2)$ as follows: Let E_t be the matrix of edge vectors of t, i.e.,

$$E_t = \left(\mathbf{v}_{j_1} - \mathbf{v}_{j_0}, \mathbf{v}_{j_2} - \mathbf{v}_{j_1}, \mathbf{v}_{j_0} - \mathbf{v}_{j_2}\right), \tag{9}$$

and $E_t = U_t \Sigma_t W_t^\mathsf{T}$ the SVD of this matrix. Furthermore, let $E'_t = U'_t \Sigma'_t W_t'^\mathsf{T}$ be analogously defined for the deformed geometry. Then

$$A_t = U'_t \Sigma'_t W_t'^\mathsf{T} W_t \Sigma_t^+ U_t^\mathsf{T} \tag{10}$$

is a linear transformation mapping the edges of t from the original to the deformed state. More specifically, it is the unique transformation with the property that the normal $\mathbf{n}_t$ maps to zero. Moreover

$$R_t = U'_t W_t'^\mathsf{T} W_t U_t^\mathsf{T} \tag{11}$$

is the closest *orthogonal* transformation to this linear map (with respect to the Frobenius norm). Note that the SVD of the undeformed geometry needs to be computed only once when considering different deformed geometries.

The constraint (C1) for the deformed geometry V' can now be described in terms of the rigid part of A_t, namely R_t, as

$$\mathbf{d}_i^\mathsf{T} R_t \mathbf{n}_t = \mathbf{d}_i^\mathsf{T} \mathbf{n}'_t \geq 0, \quad t \in C_i \tag{12}$$

since A_t does not have any effect on $\mathbf{n}_t$.

6.2. Shape energy

Our deformation energy follows the ARAP approach, i.e. we wish to minimize

$$E(V', R') = \sum_{t \in C_i} \sum_{j,k \in t} w_{jk}^t ||\mathbf{v}'_j - \mathbf{v}'_k - R'_t(\mathbf{v}_j - \mathbf{v}_k)||^2 \tag{13}$$

where w_{jk}^t is the cotan weight for edge (j,k) of triangle t. In contrast to [SA07] we use per face transformations rather than per vertex ones. Not considering any constraints, the main feature of this energy is that it can be easily minimized by alternating between minimizing w.r.t. V', which requires solving a linear system, and minimizing w.r.t. the rotations $\{R'_t\}$, whose solution is $R'_t = R_t$, as defined above.

In the usual setup, a subset of vertex positions is constrained, so that constraints are taken care of in the global step. The main difference in our setup is that we need to minimize the energy such that constraints on the rotations R_t in Eq. (12) are satisfied. Note that if E is minimized, the rotations R'_t have converged against R_t so that satisfying Eq. (12) really means V' satisfies (C1).

6.3. Lagrangian setup

We suggest to optimize an approximation of the *Lagrange dual* form [BV04] of the constrained optimization problem:

$$\max_{\lambda} \inf_{V', R'} \left(E(V', R') - \sum_{t \in C_i} \lambda_t \mathbf{d}_i^\mathsf{T} R'_t \mathbf{n}_t \right). \tag{14}$$

This is only an approximation because the constraint is formulated in terms of the variable R'_t rather than in the rigid part R_t of the actual transformation A_t, which would depend on V'. We discuss the consequences of this approximation later.

Computing the infimum w.r.t. V' is independent of λ and yields the same linear system as when $E(V', R')$ is minimized. Optimizing for the rotations R' is still a local problem, i.e., we can optimize each pair R'_t, λ_t independently.

The KKT conditions for a local optimum require that $\lambda_t \geq 0$ and $\lambda_t \mathbf{d}_i^\mathsf{T} R'_t \mathbf{n}_t = 0$. The first condition leads to $\mathbf{d}_i^\mathsf{T} R'_t \mathbf{n}_t \geq 0$, which together with the second condition yields exactly two cases:

1. $\mathbf{d}_i^\mathsf{T} R'_t \mathbf{n}_t > 0$: This means $\lambda_t = 0$, and the minimization is identical to the standard ARAP case, i.e. the optimum is found for $R'_t = R_t$.
2. $\mathbf{d}_i^\mathsf{T} R'_t \mathbf{n}_t = 0$: This constrains R'_t so that it rotates $\mathbf{n}_t$ into the plane orthogonal to $\mathbf{d}_i$. Among these rotations the optimal one minimizes $E(V', R')$, which means it is the rotation from this set closest (in the Frobenius norm) to A_t. This is equivalent to

$$\min_{R'_t} \|R'_t - R_t\|_F, \quad \mathbf{d}_i^\mathsf{T} R'_t \mathbf{n}_t = 0. \tag{15}$$

Note that if R_t rotates $\mathbf{n}_t$ into $\mathbf{n}'_t$, we wish to further rotate $\mathbf{n}'_t$ so that it is orthogonal to $\mathbf{d}_i$. The shortest path

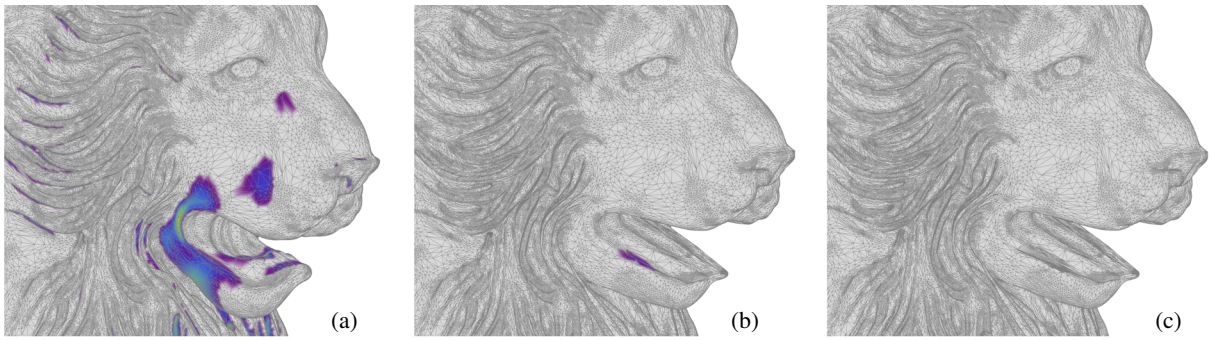


Figure 7: *Constraint violations have to be removed after segmenting the shape (a). Deforming the shape using constraint rotations yields a mesh that is* almost *a height field (b). Remaining error can be eliminated by remeshing (c).*

for this rotation results from rotating around $\mathbf{d}_i \times \mathbf{n}'_t$. The necessary angle of rotation is $\cos^{-1}(1 - \mathbf{d}_i^\mathsf{T}\mathbf{n}'_t)$. Denote the rotation matrix with this axis and angle R_t^* then the optimal transformation is $R'_t = R_t^* R_t$.

In order to compute the optimal transformation, one first computes R_t, checks if $\mathbf{d}_i^\mathsf{T} R_t \mathbf{n}_t > 0$ and than sets $R'_t = R_t$ if true and $R'_t = R_t^* R_t$ else.

6.4. Optimization

Repeating the minimization steps for V' and R' leads to a steady state. For most faces we find $R'_t = R_t$ meaning that the constraint (C1) is necessarily satisfied. For some faces we have $R'_t = R_t^* R_t$ in the converged case meaning it is unclear if, indeed, $\mathbf{d}_i^\mathsf{T} R_t \mathbf{n}_t \geq 0$.

Note that the meshes we subject to this process require rather small deformations by design. This means the least squares fit of triangles to R'_t leads to very small deviations from the constraint. In fact, in most examples in the paper we have taken the mesh resulting from this optimization and directly used it for manufacturing (e.g. computing the tool path for the CNC mill).

In case the violation seems larger than acceptable (this will depend on the application), we suggest the following remeshing procedure: project the surface patch on the plane orthogonal to $\mathbf{d}_i$ and delete the violating triangles. The resulting holes can be triangulate in the plane (we use Triangle [She01] for this task). Finally, we assign height values along $\mathbf{d}_i$ for the newly added vertices based on interpolation. This necessarily fixes all possibly remaining problems. It is of course also possible to use the remeshing method directly without the deformation step. This will, however, result in noticeable artifacts (Figure 8).

Figure 7 shows the most extreme case of constraint violation (after deformation) among the examples we have processed, and an illustration of the post processing step.

6.5. A-priori estimation of the deformation

Recall that we need to estimate whether a region of triangles in a component can be fixed easily. We use the Lagrangian to compute an estimate of the necessary deformation, *without actually performing it*.

At initialization, the triangles contributing to the Lagrangian are exactly the ones violating the constraint. For the initial step, we set the transformation for these triangles to $R'_t = R_t^*$ (note that $R_t = I$ at initialization). We can evaluate

$$E(V,R') = \sum_{\mathbf{d}_i^\mathsf{T}\mathbf{n}_t<0} \sum_{j,k\in t} w_{jk}^t ||(I - R_t^*)(\mathbf{v}_j - \mathbf{v}_k)||^2. \quad (16)$$

The area of a triangle $t = (i,j,k)$ can be expressed as $\frac{1}{2}\sum_{j,k\in t} w_{jk}^t ||(\mathbf{v}_j - \mathbf{v}_k)||^2$ [Hir03]. Equation 16 therefore measures the total projected area of flipped triangles. We use this quantity during segmentation (Section 5) to decide if connected components should be added to a face set.

Remark An alternative approach to our deformation technique is to use a parametrization procedure. Projecting the component along the fabrication direction and optimizing for a bijective mapping in that plane under appropriate boundary constraints will always lead to a valid solution. We implemented this technique using an interior point method suggested by Schüller et al. [SKPSH13]. However, because

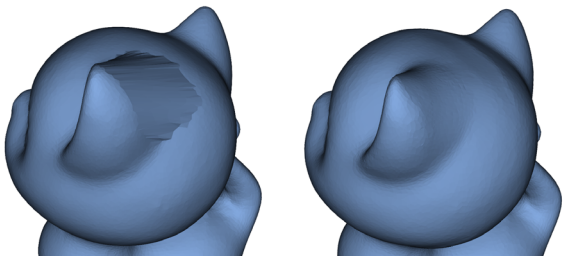


Figure 8: *Direct remeshing of the segmented mesh introduces noticeable artifacts (left). Constrained deformation gives plausible results without artifacts (right).*

this method does not consider the component orthogonal to the projection plane, geometry can get severely distorted. We found our approach to give consistently superior results in our application.

7. Implementation

We implemented the algorithm and the user interface in C++. We use the software library Eigen [GJ*10] for numerical computations and CGAL [CGA] for geometric intersection queries. To sample the face sets, faces violating (C1) and (C2) have to be identified for each direction $\mathbf{d}_i$. While checking for (C1) is straightforward, (C2) is more difficult due to the global nature of the problem. To identify occluded faces, we project all faces of the label face set onto a plane orthogonal to $\mathbf{d}_i$ and record intersecting triangles. Since directions are processed individually the sampling can potentially be accelerated significantly by parallelization. The energy minimization uses a modified version of code for multi-label optimization provided by Delong et al. [DOIB12].

7.1. Performance

The sampling of face sets L_i takes the bulk of computation time. Fortunately, this can be done in a preprocessing step. For the model 'lion-vase' with 200k vertices processing one direction takes between 1.5 and 2 seconds with a mean of 1.83 seconds. For all our examples we used 600 direction, therefore preprocessing takes about 18 minutes. The bottleneck here is the computation of 2d triangle-triangle intersections. To speed up the intersection test we use a spatial data structure by Zomorodian et al. [ZE00] as implemented in CGAL. Therefore, according to the analysis in [ZE00], the sampling step scales almost linearly with respect to the number of faces in practice.

Optimizing Eq. (5) via the α-expansion algorithm takes about 3 minutes for the 'lion vase' with 600 sampled directions and about 1.7 minutes for 300 directions. Using 10 iterations of ARAP deformation amounts to 5 seconds of computation time. For all examples in Figure 15 run times are comparable to the 'lion-vase'. All timings were taken on a 3.9 GHz Quad Core machine.

8. Results

8.1. Multi-piece molds

We manufactured a number of reusable multi-piece molds generated by our technique and used them to cast object in resin and plaster. All results have been computed fully automatically with the same set of parameters and without any user interaction. Our system generated the mold pieces by extruding seam edges and triangulating the planar back sides.

We used a factor of $\lambda = 10^3$ to penalize the number of labels and a size tolerance for connected components of flipped faces of $\gamma = 0.05$ (all meshes have been scaled to fit into the unit cube). All results were generated with 600 adaptively sampled directions. Computation times for all examples range from 20 to 35 minutes. Mold pieces were milled with a conventional 3-axis milling machine using a 3mm drill bit. Results are shown in Figure 15.

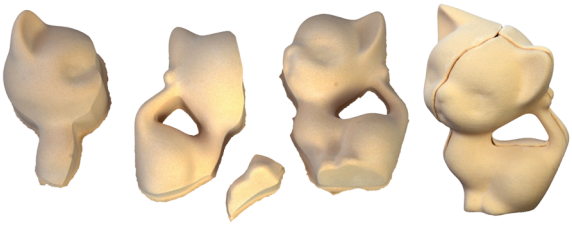


Figure 9: *The kitten model was decomposed into four pieces using our technique and milled with a 3-axis CNC mill. After manually hollowing out the backsides, the pieces were assembled and glued together.*

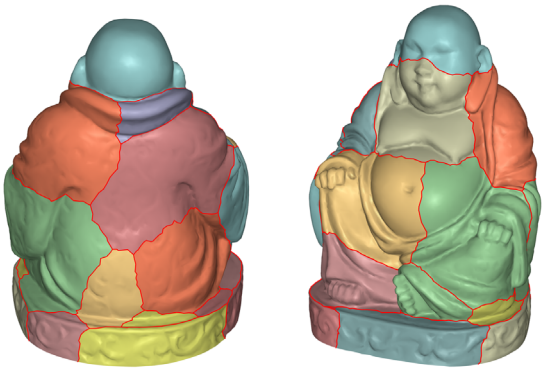


Figure 10: *Segmentation into volume constrained height field patches.*

8.2. Milling

Instead of using mold casting, height fields can also be milled directly using a 3-axis milling machine. In order to assemble the parts, their backsides have to be machined manually. Since the backsides will not be visible, no special care has to be taken in this step. Figure 9 shows an example object assembled from milled height fields, for which the algorithm failed to produce a valid multi-piece mold. Using our technique every 3-axis milling machine turns into an affordable large-scale 3d printer.

8.3. Volume constraint

Using the volume constraint described in Section 4, objects can be decomposed into height-field patches which fit into a predefined cuboid build volume (Figure 10). This is de-

Figure 12: *The bunny has been segmented into two parts with limited overhang angle and printed with the Form 1 STL printer (top) and the Ultimaker 2 (bottom). No support is needed for the surface.*

sirable in situation where either the machining range or the stock material is limited in terms of volume.

γ	Seg. Resid.	Parts	ARAP Resid.
0	2044.04	50	0
0.001	813.2	17	17.9906
0.005	538.9	9	45.127
0.01	512.52	9	49.0432
0.05	389.68	5	71.2212
0.1	380.22	5	107.334
0.2	354.56	4	147.111

Table 1: *Experimental results for the 'lion vase' model.*

8.4. Influence of γ

The main parameter used to find a compromise between the number of parts and the amount of deformation is the parameter γ. To demonstrate the impact of γ on the solution we computed a decomposition of the model 'lion vase' for different parameter values and recorded the residual of the segmentation energy (Eq. (5)) without label costs ($\lambda = 0$). Moreover, we recorded the number of parts and the residual of the ARAP energy (Eq. (13)) to assess the amount of non-rigid deformation due to the least squares solve. This residual manifests itself as distortion in the mesh; triangles might be sheared or scaled. For each γ value we ran the algorithm 20 times using random label orders and picked the result with lowest segmentation energy (Eq. (5)). Figure 7 shows the resulting segmented and deformed models. While values below $\gamma = 0.05$ clearly have too many segments to be useful in practice, a value of 0.2 introduces too much distortion.

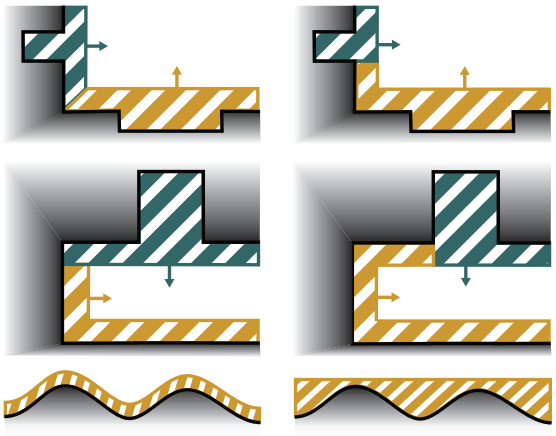


Figure 13: *Depending on the segmentation, mold pieces (colored) might block each other, even though every segment (black) is ray-accessible (top row). If segments are not ray-accessible, mold pieces can be removable, depending on the segmentation (center row).*

8.5. Application to 3d printing

Hu et al. [HLZCO14] recently exploited the fact that decomposing a shape into parts with limited overhang angle will save support material in 3d printing based on fused deposition modeling or stereolithography. We can modify our segmentation algorithm slightly to save support material as well. The goal is to segment a given shape into parts with limited slope angle. We achieve this by adjusting the unary energy to

$$\widetilde{E}_i(l_i) = \begin{cases} 0 & \text{if } \mathbf{d}_{l_i}^\mathsf{T}\mathbf{n}_i \geq -\cos(\alpha_{hang}) \\ \infty & \text{otherwise.} \end{cases} \tag{17}$$

Moreover, we have to account for the base surface of each part since additional support material might be required for non-planar base surfaces. We address this issue by favoring planar seams. term. Since the segments will be fabricated layer by layer, we do not have to account for (C2) and (C3). The modified pairwise energy term is given by:

$$\widetilde{E}_{i,j}(k,m) = \begin{cases} 0 & \text{if } k = m, \\ (\mathbf{d}_k^\mathsf{T}\mathbf{e}_{i,j})^2 + (\mathbf{d}_m^\mathsf{T}\mathbf{e}_{i,j})^2 & \text{otherwise} \end{cases} \tag{18}$$

where $\mathbf{e}_{i,j}$ denotes the edge connecting triangles t_i and t_j. After the shape has been segmented, we triangulate the base planes and print the segments.

9. Discussion

9.1. Global ray accessibility

As noted in Section 3.4, faces might not be ray-accessible, meaning the face might be blocked by another face in fabrication direction. It would be straightforward to form sets

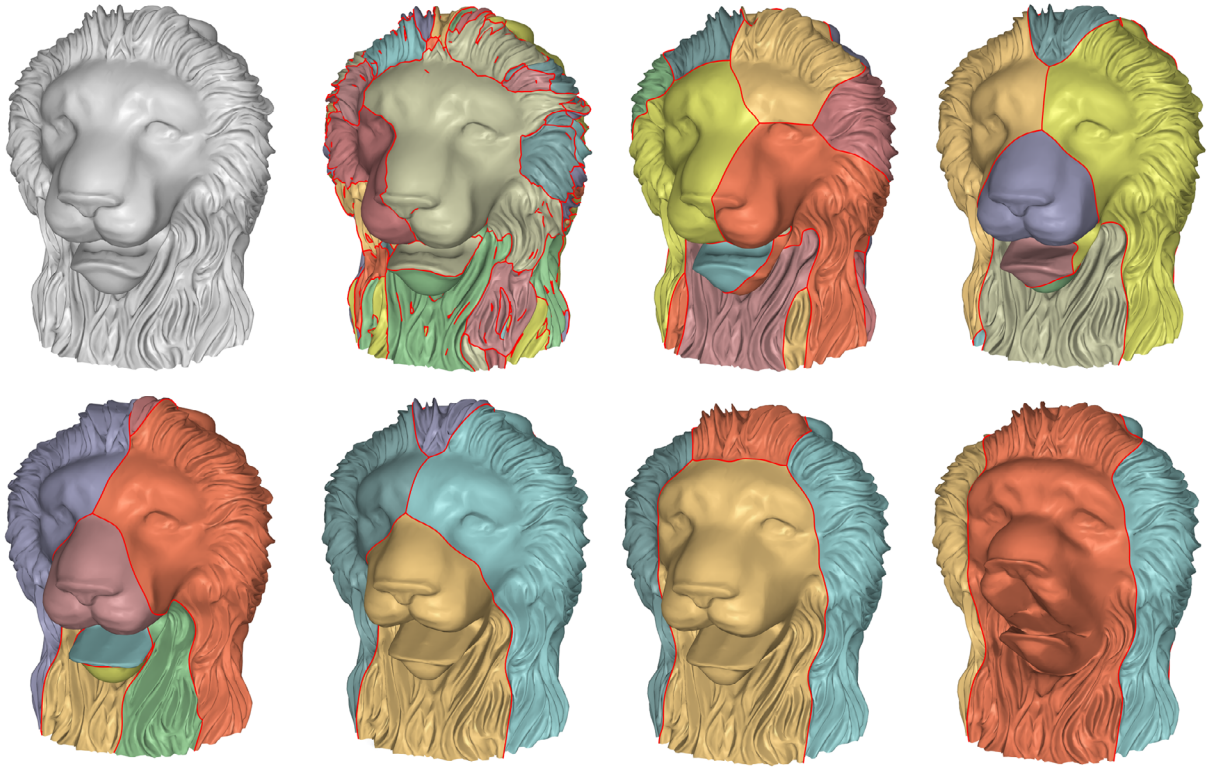


Figure 11: *Several instances of segmented meshes for* $\gamma = 0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2$. *By varying* γ *we can control the number of faces that violate (C1). A value of zero yields no constraint violation (top, center left). Consequently many segment are generated. Choosing* γ *to large yields to much deformation (bottom, very right).*

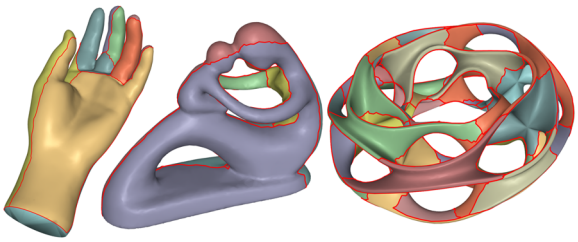


Figure 14: *For these more complex shapes mold pieces cannot be (dis-)assembled due to interlocking.*

$\{L_j\}$ of faces such that each face $f_i \in L_j$ is also ray-accessible in direction $d(L_j)$. However, this approach seems questionable for the following reasons:

- Valid multi-piece molds can often be generated without requiring faces to be ray-accessible. Figure 13 (center row) demonstrates a situation where faces are not ray-accessible w.r.t. their fabrication direction. Depending on the segmentation, mold pieces can be disassembled (left) or not (right). Enforcing that all faces are ray-accessible would limit the solution space, likely resulting in more segments for most objects.
- Even if faces are ray-accessible, it is not guaranteed that this still holds true after the deformation and remeshing step. Incorporating accessibility constraints into the deformation is not straightforward.
- The shape of a mold piece cannot be trivially deduced from the surface segment. For example, if the mold is manufactured with a 3-axis mill, the shape will have a flat base with varying height. This makes a global analysis of the assembly difficult at the stage of segmentation. Figure 13 (bottom, right) illustrates a mold piece with a flat base as opposed to a 'thin' mold piece (bottom, left). Even if using 'thin' mold pieces, disassembly might be impossible, depending on the segmentation (top, left).

9.2. Limitations

In its current form, our algorithm is limited to fairly simple shapes, possibly with small scale details. For more complex geometry our system might fail to produce mold pieces that can be assembled or disassembled (Figures 13, 14) or might generate too many pieces to be useful in practice. While we do believe that some shapes are not suitable for heightfield-based fabrication, we hope to improve on our result in terms of a better deformation algorithm. Simplifying the topology, for example, might help in finding a manufacturable approximation of the input shape.

9.3. Conclusion

We proposed a method to fabricate a freeform geometry with classical manufacturing techniques such as 3-axis milling, casting, and stamping. These techniques are limited to shapes that are height fields. To nonetheless fabricate more general objects we segment them into multiple components that are manufactured separately and then assembled. Strictly enforcing the height field constraint for each component leads to a prohibitively large number of pieces. During segmentation we hence allow small "islands" that violate the constraint in each part. After segmentation these violations are fixed using a deformation step. We employed a variation of the as-rigid-as-possible approach to obtain these deformations. Our technique can also be employed to improve 3d printing by reducing the need for support material or enabling to place it in inconspicuous places.

We fabricated molds and objects for various 3d models. The results demonstrate the practicality of our approach. In particular, we typically obtain only a small number of pieces with a deformation that is not noticeable without comparison.

Acknowledgments

The authors would like to thank Jacob Seibert for his help with mold manufacturing and casting. We are also grateful for the constructive reviewer comments that helped us improve the paper. The meshes are courtesy of AIM@SHAPE, Thingiverse and Stanford Computer Graphics Lab. This work has been supported by the ERC through grant ERC-2010-StG 259550 ("XSHAPE").

References

[BV04] BOYD S., VANDENBERGHE L.: *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 7

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell. 23*, 11 (Nov. 2001), 1222–1239. 5, 6

[CGA] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org. 9

[CR09] CHAKRABORTY P., REDDY N. V.: Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds. *Journal of Materials Processing Technology 209*, 5 (2009), 2464 – 2476. 2

[CSaLM13] CHEN D., SITTHI-AMORN P., LAN J. T., MATUSIK W.: Computing and fabricating multiplanar models. *Computer Graphics Forum (Proceedings of Eurographics 2013) 32*, 2pt3 (2013), 305–315. 2

[DOIB12] DELONG A., OSOKIN A., ISACK H. N., BOYKOV Y.: Fast approximate energy minimization with label costs. *Int. J. Comput. Vision 96*, 1 (Jan. 2012), 1–27. 9

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 9

[HBA12] HILDEBRAND K., BICKEL B., ALEXA M.: crdbrd : Shape Fabrication by Sliding Planar Slices. *Computer Graphics Forum (Eurographics 2012) 31*, 2 (2012). 2

[Hir03] HIRANI A. N.: *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology, 2003. 8

[HLZCO14] HU R., LI H., ZHANG H., COHEN-OR D.: Approximate pyramidal shape decomposition. *ACM Transactions on Graphics, (Proc. of SIGGRAPH Asia 2014) 33*, 6 (2014), to appear. 2, 10

[JKS05] JULIUS D., KRAEVOY V., SHEFFER A.: D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3 (2005), 581–590. 2

[KBM06] KHARDEKAR R., BURTON G., MCMAINS S.: Finding feasible mold parting directions using graphics hardware. *Computer-Aided Design 38*, 4 (2006), 327 – 341. 2

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: Partitioning models into 3D-printable parts. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (Dec. 2012). 2

[Lip13] LIPMAN Y.: Construction of injective mappings of meshes. *CoRR abs/1310.0955* (2013). 3

[LML09] LI W., MARTIN R., LANGBEIN F.: Molds for meshes: Computing smooth parting lines and undercut removal. *Automation Science and Engineering, IEEE Transactions on 6*, 3 (2009), 423–432. 2

[MSM11] MCCRAE J., SINGH K., MITRA N. J.: Slices: A shape-proxy based on planar sections. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 168:1–168:12. 2

[PG04] PRIYADARSHI A. K., GUPTA S. K.: Geometric algorithms for automated design of multi-piece permanent molds. *Computer-Aided Design 36*, 3 (2004), 241 – 260. 2

[PWLSH13] PRÉVOST R., WHITING E., LEFEBVRE S., SORKINE-HORNUNG O.: Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4 (July 2013), 81:1–81:10. 2

[RS90] RAVI B., SRINIVASAN M.: Decision criteria for computer-aided parting surface design. *Computer-Aided Design 22*, 1 (1990), 11–18. 2

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of the Symposium on Geometry Processing* (2007), SGP '07, pp. 109–116. 7

[She01] SHEWCHUK J. R.: Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications 22* (2001), 1–3. 8

[SKPSH13] SCHÜLLER C., KAVAN L., PANOZZO D., SORKINE-HORNUNG O.: Locally injective mappings. *Computer Graphics Forum (Proceedings of Symposium on Geometry Processing) 32*, 5 (2013), 125–135. 8

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph. 31*, 4 (July 2012), 48:1–48:11. 2

[VGB*14] VANEK J., GALICIA J. A. G., BENES B., MĚCH R., CARR N., STAVA O., MILLER G. S.: Packmerger: A 3d print volume optimizer. *Computer Graphics Forum* (2014). 2

[ZE00] ZOMORODIAN A., EDELSBRUNNER H.: Fast software for box intersections. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry* (New York, NY, USA, 2000), SCG '00, ACM, pp. 129–138. 9

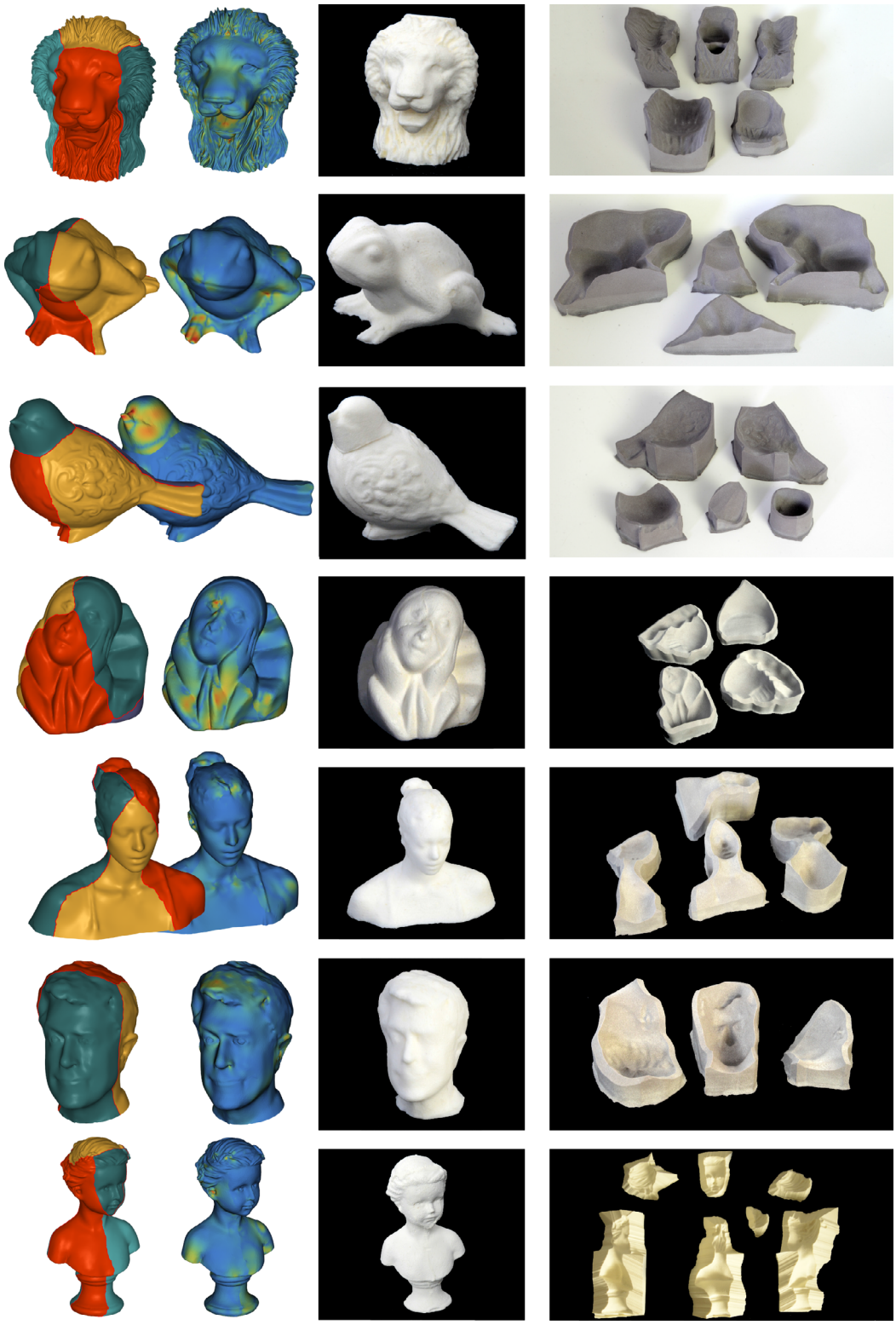


Figure 15: *Several multi-piece molds were generated using our method. Based on the segmented input meshes (first column), the meshes were deformed (second column, scale distortion color coded) and mold pieces were fabricated using CNC milling (last column). These molds where assembled and filled with liquid resin to produce physical replicas (right center column).*