Computers & Graphics (2020)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

# Interactive design of castable shapes using two-piece rigid molds

Oded Stein[a,*], Alec Jacobson[b], Eitan Grinspun[a]

[a]Columbia University, 530 West 120th Street, 10027 New York, USA
[b]University of Toronto, Bahen Center, 40 St. George Street, Room 5266, M5S 2E4 Toronto, Canada

ARTICLE INFO

*Article history*:

*2000 MSC:* 68U05, 68U07

*Keywords:* Computer Graphics, Digital Fabrication, Casting

ABSTRACT

Two-piece rigid molds are particularly amenable to mass-manufacturing. Arbitrary objects cannot be realized with such a mold, because the cast will collide with the mold during removal. Most shapes are far away from being *castable*, i.e., realizable with such a two-piece rigid mold. When starting with an arbitrary shape, large deformations are needed to produce castable shapes. Such large deformations require user interaction together with a design process that is aware of fabrication constraints. We present such a design tool to generate two-piece rigid molds separated by a planar cut for a wide variety of shapes. Our casts can be produced in one single piece from a rigid, reusable mold and do not require further assembly after casting. We use a castability energy that can be optimized with gradient-based methods combined with an elastoplastic deformation based on the as-rigid-as-possible method. The tool enables a designer to deform an input shape significantly and arrive at an output shape that is castable.



## 1. Introduction

A *mold* is a cavity that can be filled with liquid or expanding material that hardens inside of it. The hardened material forms a shape which is called the *cast*. After it has completely hardened, it is removed from the mold. Fabricating casts of objects using molds is a very old method employed by humans for many centuries in art and industry [1].

mold cast

In the food industry the use of molds is widespread, for example to manufacture chocolate sculptures [2] or candy [3]. In metalworking, casting is a popular technique to manufacture many metal structures [4]. Of particular interest is casting with rigid, reusable molds. These kinds of molds are well suited for parallel fabrication, where a large number of molds are simultaneously filled with a material that will harden or expand. After the curing period, the results are removed and the molds can be used again. Prominent examples of this are chocolate bunnies, as seen in Fig. 2.

In this work we study a specific subset of rigid molds inspired by these chocolate bunny molds: *two-piece rigid molds*. The rigidity of the mold means the cast can be removed without deforming the cast or mold. Two-piece rigid molds are easy to assemble and disassemble compared to general many-piece molds, which makes them useful for mass manufacturing. Non-rigid molds require even more care when assembling and disassembling, greatly complicating mass manufacturing.

Using a two-piece rigid mold to manufacture casts is very restrictive. An arbitrary shape is in general *not* castable. Navigating the space of castable shapes can be unintuitive. As a result, many shapes that are produced often do not feature complicated ge-

*Corresponding author: Email: oded.stein@columbia.edu

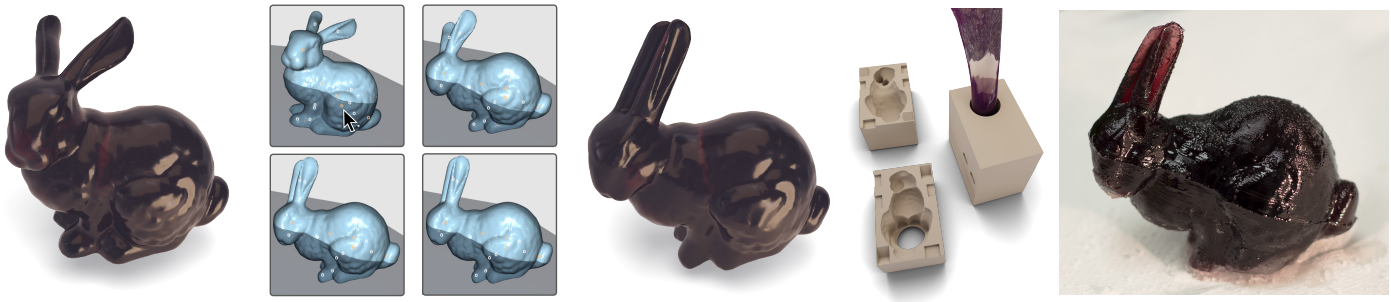
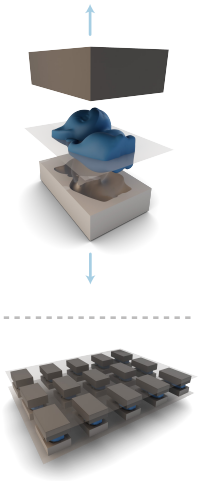



**Fig. 1. Mold design and fabrication of bunny gummy candy. The original mesh (*far left*) is manipulated with the design tool (*left*) to produce a shape that is castable (*center*) with a two-piece rigid mold (*right*). The resulting shape is then fabricated as candy (*far right*).**
**Note that two ears of the bunny are distinct, and have not been merged to each other or the body. Also note that there is almost no visible volume loss.**

ometries (see, for example, the relatively simple commercial chocolate bunny shapes in the inset: they do not look much more complicated than a bas-relief with distinct features such as ears merged to each other or the body and the head flattened [5, 6, 7]).

Existing tools focus primarily on generating molds for a given shape, only insignificantly changing it as post-processing. The challenge with two-piece molds however is that a given shape generally requires significant deformation to become castable. Analyzing a given shape and slighly adjusting it is insufficient, as in general there is no small change that will make the shape castable in a two-piece rigid mold. We propose a *shape design tool* that accounts for both the designer's preferences as well as the fabrication constraints of castability. With this tool it is possible to design shapes that are more complicated than what can be found in the world, such as in chocolate bunnies or candies.



We consider two mold pieces that meet at a plane (see inset). The cast can be removed as a single piece by translating the mold. The mold is a boolean negative of the cast mesh which is cut in half by a plane (the *cut plane*). Such a mold can be trivially extended to a *multi-cavity mold* (not to be confused with a many-piece mold), a mold with multiple identical cavities suitable for parallel fabrication that produces multiple casts at once. There is no easy way to generalize a multi-cavity mold to arbitrarily many pieces with arbitrary removal directions—in general, the different pieces collide with pieces from other cavities when rigidly translated. An example of such a multi-cavity mold designed with our tool can be seen in Fig. 3. While any two-piece mold that can be successfully disassembled can be used for parallel fabrication, we only consider molds that are separated by a simple cut plane. More elaborate cut planes complicate the question of fabricability—restricting ourselves to the problem of planar cuts reduces the problem to a simpler problem with an easier solution. In practice, planar cut molds are ubiquitous, see for example the mold used in Fig. 2.

For most shapes, there does not exist a cut plane that results in an object castable with a two-piece rigid mold. In general, some part of the object will collide with the rigid mold during removal (see Fig. 4).

A simple mathematical condition can be used to characterize castability in two-piece rigid molds that meet at a cut plane. We use this condition to formulate a continuous castability energy to optimize the position of the cut plane, the removal directions of the cast from the mesh, and the shape of the cast object. Optimizing this energy with respect to the vertex position using gradient-based linesearch methods however ignores the look and feel of the object. Especially for large deformations, this approach will not give a satisfactory resulting cast shape that is sufficiently similar to the input shape. In order to preserve the object's character we treat it as a solid volumetric object and use an elastoplastic deformation based on the as-rigid-as-possible (ARAP) method to deform the shape to find a two-piece rigid mold in a way that preserves the shape's character.

The castability energy as well as the elastoplastic deformation form the basis of our interactive tool for designing castable shapes. This design tool allows the user to guide the deformation of any input shape towards an object that is castable in a two-piece rigid mold. The tool operates on a triangle mesh and outputs a castable mesh, as well as a cut plane and removal directions from the mold.

In summary, we

- introduce an energy to quantify how far away a surface is

**Fig. 2. A two-piece rigid mold and the resulting chocolate bunny.**
**Images © Chocoladefabriken Lindt & Sprüngli AG**

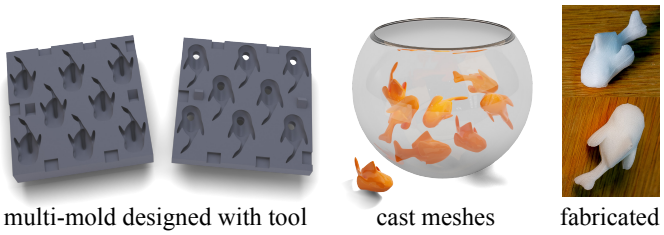



**Fig. 3. A multi-cavity mold for one of the shapes designed with our tool, as well as a visualization of the cast meshes. Two-piece rigid molds are suitable for these kinds of multi-molds.**

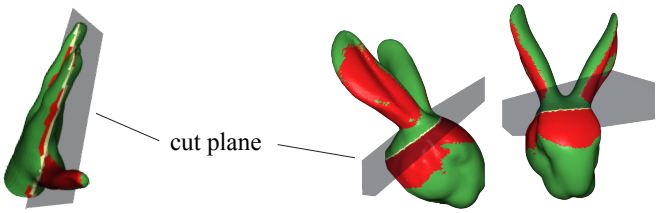



**Fig. 4. In general, an object can't be realized with any two-piece rigid mold. In these examples, the red portion of the objects (according to the removability condition (1)) will collide with the mold during removal. In this example, the directions are normal to the cut plane.**

from being castable with a two-piece rigid mold;

- describe an elastoplastic deformation based on the volumetric ARAP method that deforms an object to make it castable with a two-piece rigid mold;
- present a design tool that enables a user to leverage these tools to design castable shapes starting from arbitrary inputs.

We fabricate some of our molds constructed with our tool using 3D printers, and make casts using modeling foam as well as gummy candy (see Fig. 1).

## 2. Related Work

### *2.1. Molds*

Zhang et al. [8] present a tool to analyze the castability of general objects for rigid injection molding. Chakraborty and Reddy [9] present a tool to generate a two-piece rigid mold and its parting directions for a shape, without modifying the shape. [10] propose an approach to automatically generate multi-piece rigid molds for CAD models. Hu et al. [11] decompose objects into pyramidal shapes that are moldable. Herholz et al. [12] develop an analysis tool that decomposes a shape into many pieces for casting with a many-piece rigid mold. While using a relaxed tolerance allows their algorithm to produce a two-piece mold, their approach is not intended for operating with such a relaxed tolerance, as depicted in the bottom right of [12, Fig. 11]. Their goal is not to create a design process to significantly deform a shape, but to decompose a shape into heightfields that can be molded with light postprocessing. We build on their *analysis* of a shape, further providing *synthesis* of new castable shapes via interactive deformation. The work of Herholz et al. [12] uses [13] to formulate conditions for castability. A similar condition is also used in this work. Malomo et al. [14] relax the rigidity constraint on the mold and introduce a framwework for the automatic design of flexible molds. This approach offers more freedom than the traditional rigid molding, but is harder to perform for a robotic disassembler. By considering flexible molds, they can consider shapes that would be more difficult to realize with rigid molds. Nakashima et al. [15] decompose a shape into many pieces that can each be cast with a two-piece rigid mold, but do not deform a shape to produce a single two-piece rigid mold.

### *2.2. Design for fabrication*

There exists a large body of work on computational designs for fabrication, as seen in Shamir et al. [16]. A few relevant examples are listed here.

Prévost et al. [17] help the user balance 3D shapes in an interactive editing sequence. Their user experience is similar to this tool: the user loads an already existing shape, and the tool guides them in designing a shape that will fulfill certain properties. A tool to take an arbitrary shape and fabricate it inside an extremely constrained volume is proposed by Schüller et al. [18]. Starting with an input shape that does not fit inside a certain constrained volume, the user achieves a bas-relief realization of their shape. Other approaches for designing bas-reliefs can be found, for example, in Weyrich et al. [19]. Tang et al. [20] introduce a design tool for developable surfaces: using their tool, a user can successively cover an input shape with developable strips which are then fitted to the shape. In Duncan et al. [21] two shapes are deformed so that they can be assembled with the same puzzle pieces. Like our tool, they perform specific deformations to their shapes in order to accommodate a certain fabrication method. Li et al. [22] present another way to compute such shape dissections so that they are also hinged: turning the first shape into the second corresponds to turning it *inside-out*.

Jacobson [23] cuts shapes so they can be recursively nested inside each other. For this application, the shapes have to be physically inserted into each other and thus have to be able to be removed without collision. He considers a removal condition similar to (1) used in this work (but does not deform the shape). A similar deformation strategy to the one used in this work is used in Hu et al. [24] to deform models so that they can be printed with minimal support in 3D printers.

## 3. Castability

Three different objects comprise the state of our tool which can be turned into a mold: the cast mesh, the cut plane, and the removal directions. The geometry of the cast is represented by a closed manifold triangle mesh $(V, F)$ with vertices $V$ and faces $F$. The *cut plane* is represented by the 4-vector $\mathbf{p}$; its first three entries are the plane normal, and its fourth entry is the offset of the plane from the origin. The two removal directions are represented by two vectors $\mathbf{d}_1$ and $\mathbf{d}_2$, one for each side of the plane. It is important to notice that the removal directions don't have

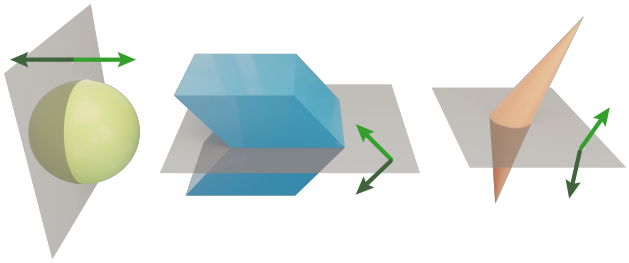


**Fig. 5. The removal directions for the cast object don't have to be the plane normals (*left*). In fact, they can differ from it significantly (*center, right*). All objects in this figure can be produced with two-piece rigid molds where the two pieces meet at a plane. The green arrows indicate the removal direction on either side of the cut plane.**

to be equal to the cut plane normals: they can differ from it significantly as can be seen in Fig. 5 (if the removal directions are not parallel to each other, the shape will be nonsmooth across the cut plane). The collection of mesh, plane and directions is called the *state*.

The goal of the design process is to arrive at a *castable* state starting with an arbitrary input state. A castable state can be turned into a mold by cutting it along the cut plane and turning the two halves into mold pieces. The cast object can then be removed from the mold with a collision-free translation. In fact, each half of a castable state fulfills the following *removability condition* for its respective removal direction:

$$\mathbf{N}_f \cdot \mathbf{d} \geq \tau \quad \forall f \in F\,, \tag{1}$$

where $\mathbf{N}_f$ is the face normal of face $f$, $\mathbf{d}$ is the removal direction, $F$ is the set of all faces, and $\tau \geq 0$ is a tolerance parameter.

Discounting friction, setting $\tau = 0$ guarantees that the object can be removed from the mold. As $\tau = 0$ allows for faces that are perfectly orthogonal to the removal direction, this can lead to problems with friction during the removal process. For that reason, some small $\tau > 0$ can be helpful in practice.

This condition appears in Herholz et al. [12] where it is called (C1). Since they are studying many-piece molds, they formulate additional conditions which are trivial for two-piece rigid molds that meet at a plane. Their approach is not sufficient to avoid global interlocking, as can be seen in [12, Fig. 14], an issue that does not exist for two-piece molds with planar cuts. By restricting the problem to planar cuts, more complicated questions about the castability of two-piece rigid molds can be avoided, leading to a simple tool.

## 4. The Castability Energy

The removability condition (1) can be used to formulate an energy that quantifies how close a state is to being castable. This leads to the definition of the *castability energy* of a state,

$$E\,(V, \mathbf{p}, (\mathbf{d}_1, \mathbf{d}_2)) := \frac{1}{2} \int_\Omega \min\left(\mathbf{N}(x) \cdot \mathbf{d}(x) - \tau, 0\right)^2 \, dA\,, \tag{2}$$

where $\Omega$ is the surface of the mesh, $\mathbf{N}(x)$ is the mesh normal at the point $x$, and $\mathbf{d}(x)$ is the removal direction $\mathbf{d}_1$ or $\mathbf{d}_2$ on the left or right side of the plane $\mathbf{p}$ respectively. Implementation details are provided in Appendix A.

States with zero castability energy are guaranteed to be castable with a certain tolerance $\tau$. In practice, the target tolerance should depend on the tolerances of the fabrication method used—a few example parameters can be seen in Appendix C. The farther away a state is from fulfilling the removability condition (1), the larger its castability energy will be, and minimizing it results in castable states if zero energy can be attained. This fact is used in the design tool to find the optimal plane and the optimal removal directions for the state.

The castability energy is continuous by construction, as it is a composition of continuous functions. The continuity (and piecewise differentiability) makes it amenable for a variety of gradient-based optimization methods.

## 5. Optimizing the Castability Energy

### 5.1. Gradient-based optimization

One way to optimize the castability energy (2), is to use gradient-based optimization methods. In the design tool, for local optimization of (2) with respect to the cut plane position and the removal directions, subgradient descent as well as L-BFGS are used with the line search described by Lewis and Overton [25].

### 5.2. Elastoplastic deformation

Using a gradient-based method to minimize the castability energy with respect to the mesh vertex positions does not yield a satisfactory result for all meshes, as this approach ignores the geometry of the surface, as can be seen in Fig. 6. It is used in the tool for postprocessing only. In order to achieve castability while preserving the character of the surface as the boundary of a solid object, we tetrahedralize the mesh and use an elastoplastic deformation to achieve castability.

We apply the As-Rigid-As-Possible method [26] with a local-global solver to a tetrahedral mesh whose boundary is our surface (see Chao et al. [27]). There are multiple options for generating tetrahedral meshes from surfaces. We use tetgen [28]. Simply deforming the surface with ARAP without considering the volume is not enough to achieve an elastoplastic deformation of the solid, as can be seen in Fig. 8.

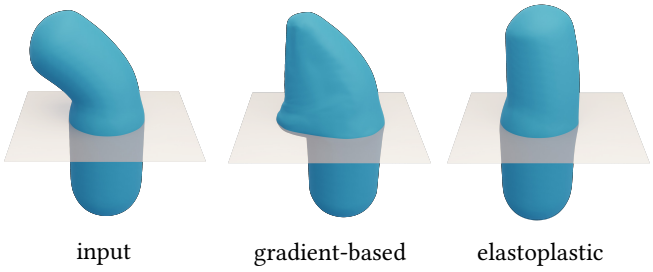



**Fig. 6. Merely minimizing the castability energy using a gradient-based method gives unnatural-looking results (*center*). Using an elastoplastic deformation to a volumetric representation of the object preserves the look and feel of the object (*right*).**

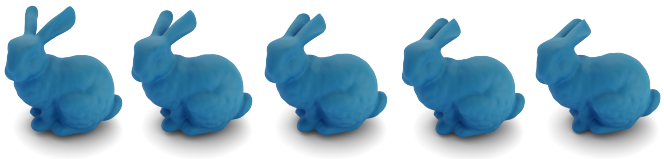


**Fig. 7. From left to right, different stages in the elastoplastic optimization of the bunny. At every frame, the ARAP procedure was restarted. The second image shows the effect of applying the ARAP optimization once only. This is not enough to achieve a castable result.**

A short introduction of the ARAP method (the way it is used by us) follows. Every tetrahedron of the mesh is a *cell*, which means it should transform as rigidly as possible. Let $T$ be the set of all tetrahedra. A rigid transformation for every tetrahedron implies that there is a rotation $R_t$ as well as a translation $c_t$ such that

$$t' = R_t t + c_t \quad \forall t \in T \tag{3}$$

where $t'$ denotes the deformed version of the tetrahedron $t$.

Completely rigid transformations for every tetrahedron do not exist in general. Because of that, we only ask of the deformation to be as rigid as possible. This requirement is formulated as the following quantity, which should be as small as possible:

$$E_A(V', (R_t)_t) = \sum_{t \in T} \sum_{p,q \in t} w_{pq}^t \left\| \mathbf{p}' - \mathbf{q}' - R_t (\mathbf{p} - \mathbf{q}) \right\|^2 \tag{4}$$

where $w_{pq}^t$ denote tetrahedral cotangent weights (involving the cotangent of dihedral angles between triangles) [29, 30]. $p, q \in t$ are vertices in the tetrahedron $t$. The quantity $E_A$ measures the amount by which the tetrahedra violate the constraint (3) (the translations $c_t$ disappear, as they are a constant addition to each vertex in the cell). Optimizing this quantity will result in a deformation that is as-rigid-as-possible (see (7) in Sorkine and Alexa [26]).

$E_A$ is optimized using a local-global approach: first the vertex positions $V'$ are computed, then the rotations $R_t$. The vertex positions are found by solving the linear system that is the gradient of (4) with respect to $V'$. The rotations are found by constructing the covariance matrix for each cell,

$$S_t = \sum_{p,q \in t} w_{pq}^t \mathbf{e}_{pq} \mathbf{e}'^{\mathsf{T}}_{pq} \tag{5}$$

where $\mathbf{e}_{pq}$ is the edge $\mathbf{p} - \mathbf{q}$. The matrix $S_t$ is then composed using singular value decomposition:

$$S_t = U_t \Sigma_t V_t^{\mathsf{T}} . \tag{6}$$

The best-fit rotation can be found by discarding the singular values:

$$R_t = V_t U_t^{\mathsf{T}} . \tag{7}$$

It might be necessary to change the sign of one column of $U_t$ to ensure that the final rotation matrix has positive determinant. (compare to (5) in Sorkine and Alexa [26]).

The optimization with respect to the vertex positions and the optimization with respect to the rotations are then performed alternately and repeated for a sufficient number of times.

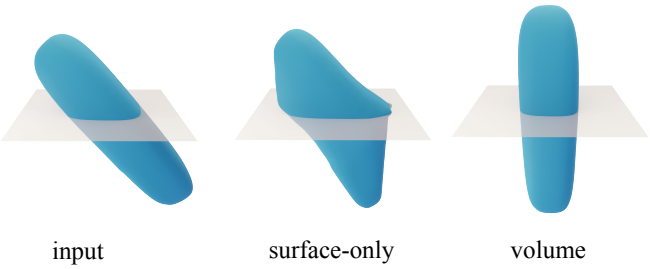



**Fig. 8. Deforming a bar (*left*) with ARAP to make it removable with the removal directions normal to the separation plane. Deforming only the surface (*center*) (similar to the postprocessing used by Herholz et al. [12]) skews the bar significantly, while deforming the entire volume (*right*) merely rotates it.**

### *5.3. Enforcing castability*

So far we have only discussed preserving the shape, but not making it castable. Instead of enforcing castability as a hard constraint on the ARAP problem itself, we ensure castability by injecting it into the rotational step of the ARAP optimization. This is done by going through all the boundary faces of the tetrahedral mesh. If a face fulfills (1), then there is nothing to do and its rotation is simply (7). If (1) is not fulfilled, then an additional rotation is injected into $R_t$ where $t$ is the tetrahedron the face belongs to. We rotate around the axis defined by $\mathbf{d} \times \mathbf{N}$, where $\mathbf{d}$ is the removal direction, and $\mathbf{N}$ is the normal vector of the boundary face, and we rotate until the boundary face fulfills the removability condition. This rotation is used because it is the smallest rotation that leads to the boundary face fultilling (1). The injected rotation is applied after the rotation (7) computed in the local ARAP step, and there could be more than one rotation if a tetrahedron has multiple boundary faces which violate the removability condition (1). In the case of a face that happens to lie on both sides of the cut plane at the same time, castability is not enforced during the ARAP deformation step—these faces will be considered in a postprocessing step.

It may happen that the procedure converges without the castability condition fully enforced. Because of that, the ARAP procedure is performed repeatedly: when the ARAP procedure no longer makes sufficient progress, the initial configuration $V$ is set to the new configuration that was just computed, and the ARAP local-global procedure is run again. In the end, this results in an *elastoplastic deformation*. The deformation is more than just *elastic*—as many elastic deformation steps are performed, the end result is a *plastic* deformation of the object, similar to shaping clay or Play-Doh. The final result is a castable mesh which is implicitly related to the initial surface through its deformation history. Fig. 7 features an example in which the initial configuration is repeatedly reset and ARAP repeatedly run.

This does not take into account any tolerance—a simple postprocessing step can be run once the elastoplastic deformation has finished, which will also address faces that lie on both sides of the cut plane. Here the castability energy (2) is optimized using a gradient-based line search method. In this very last step, only very small deformations of the shape are needed, so the problem described in Fig. 6 does not occur.

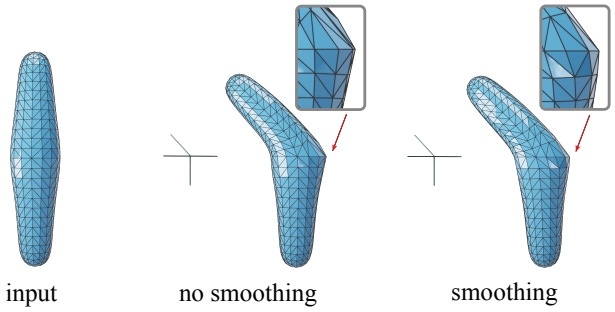



**Fig. 9. Smoothing rotations during ARAP optimization can help produce smoother result meshes.**

A similar approach is used for post-processing in Herholz et al. [12], although only on surfaces and not repeatedly. Their method is only used for polishing a solution. Repeatedly applying a Herholz-style ARAP deformation to a surface mesh does not respect the volumetric character of the object and can lead to very unintuitive deformations, as seen in Fig. 8.

### *5.4. Smoothing rotations*

Enforcing rotations as described in this section can cause nonsmoothness when some tetrahedra have an additional rotation applied during the ARAP optimization, and some tetrahedra do not. This can result in nonsmoothnesses in the cast mesh, as can be seen in Fig. 9. A way to fix this is to smooth all rotations with a simple iterative smoothing. Let $t$ be a tetrahedron, and $u_1, \cdots, u_n \in A_t$ all the tetrahedra adjacent to it. Then the rotations can be smoothed with the following operation:

$$R_t^* = \text{blend}\left((1-\gamma), R_t; \frac{\gamma}{n}, R_{u_1}; \cdots; \frac{\gamma}{n}, R_{u_n}\right) \tag{8}$$

where $\gamma$ is a parameter that controls how much to smooth. Appendix C features a few examples of the parameters used for rotation smoothing strength.

The rotations cannot simply be blended as a linear blend of rotation matrices, since the space of rotation matrices is not closed under linear blending. As a result, the blending is done in the space of quaternions (see, for example, Gramkow [31]).

## 6. The Design Tool

With the continuously optimizable castability energy, the gradient-based optimization, and the elastoplastic deformation, we now have all the tools to put together the design tool for two-piece rigid molds. In addition to choosing the cut plane and the removal directions, if desired, there are two ways in which the user mainly interacts with the tool: pinning parts of the shape to the plane and weighting special regions of the mesh that are especially important to the user.

### *6.1. Pinning points to the cut plane*

Certain parts of a large and complicated mesh that the user absolutely wants to lie on the cut plane can be pinned to the cut plane during the elastoplastic deformation (see Fig. 10, top). In addition to preventing the cast mesh from moving around too much during the optimization, people or animals that are hunched over or curled up can be unraveled onto the cut plane by pinning a few vertices to it. Then, the rest of the features can be rotated to be castable using the elastoplastic deformation. This constrains these vertices to only move on the cut plane. All these parts together enable large deformations that give castable results for non-castable input states. The freedom in pinning points to the cut plane can be used by a designer to achieve a variety of different results (see Fig. 16 top and bottom left). Selecting these points is a very important part of the design process, as different pinned points can lead to very different results starting with the same initial shape.

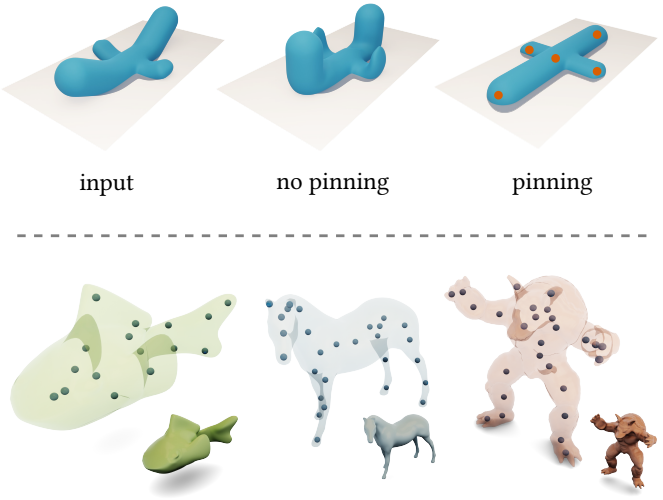



**Fig. 10. Top: with some inputs (*left*), simply rotating faces to fulfill the removability condition as described in Section 5.2 can lead to extreme deformations (*center*). Pinning a few points to the cut plane with ARAP avoids these deformations (*right*).**
**Bottom: examples of some surfaces, and the points on their medial graph, generated with Q-MAT [32]. The points the user wants to pin are often a subset of these points, which makes this a very useful heuristic.**

The parts of the tetrahedral mesh that are pinned to the plane do not need to be chosen by the user. The points that often make the most sense to be fixed are points that lie on the medial surface of a mesh. We use the Q-MAT method [32] to compute a medial graph of the input surface. The points on this medial graph are then used to suggest a small number of points on the tetrahedral mesh to the user to pin to the plane.

This removes the need for the user to pick which vertices of the tetrahedral mesh should be pinned to the plane. When looking at a model, a user may feel that certain mesh features should align to the cut plane. The medial graph of a surface pruned to only a few points is a good tool to identify the most important features that a user might want to fix to the cutting plane (see Fig. 10, bottom). If the suggested points do not suffice, the user can always opt to fix any point on the surface of the shape, or points on the inside of the shape (computed by casting a ray through the shape at the point the user clicked and then fixing the point that is closest to being in between the two locations where the ray first enters and leaves the surface).

### 6.2. Weighting special regions

Sometimes the repeated ARAP optimization can deform certain features that the designer deems especially important during the elastoplastic deformation. This can happen if small features, such as eyes or ridges, are very close to parts of the mesh that will greatly deform in order to make the mesh castable.

A user can prevent this by marking these parts of the mesh as especially important in the tool. The tetrahedra corresponding to faces that have been marked as important get an additional *importance weight* in the ARAP energy (4). For these tetrahedra, the value of $w^t_{pq}$ is doubled every time the user weighs them. This pushes the deformation onto other parts of the mesh, preserving the features that the designer cares about most. An example of this can be seen in Fig. 11.

### 6.3. Putting it all together

The user has multiple ways of interacting with the tool (see Fig. 12). During the design process, the user can manipulate the position of the plane as well as the removal directions. The tool can also find the globally optimal position of the plane by optimizing (2) via random sampling and then local optimization. Faces on the mesh can be painted to weight them for the ARAP optimization (see Section 6.2). Points in the mesh can be selected to pin to the plane (see Section 6.1), either suggested points or points of the user's choosing.

The user can toggle the continuous elastoplastic deformation, employing continuous optimization of the castability energy with respect to removal directions or not. The user can reset the cut plane and directions, or restore the cast mesh to its undeformed state with pinned points. The pins, weights, and other inputs can be varied until a suitable design is obtained. There is an option for postprocessing the mesh where the castability energy (2) is optimized using a gradient-based optimization method to ensure that the final result has zero castability energy with a small fabrication tolerance $\tau > 0$. This postprocessing can't be applied too early, otherwise it leads to unattractive distortions of the object, see Fig. 13.

The mesh is presented to the user at all times. Violations of the removability condition (1) can be indicated visually on the mesh. The cut plane and removal directions are visualized as well. Within the mesh, the suggested points to pin to the plane are faintly visualized, as well as their projections to the cut plane. If a user selects any point to pin, it is visualized as a pink circle, together with its projection to the cut plane. The weighted faces are colored differently from the other faces to set them apart visually. The more a face is weighted, the darker it is rendered.

An example user interacting with the tool to design a castable shape can be seen in Fig. 15 as well as the included video. The main interaction loop is presented in pseudocode form in Algorithm 1.

**Algorithm 1** The main interaction loop as pseudocode

```
10  user picks initial plane or tool finds globally optimal
      initial plane.
20  user picks points of the object to pin to the plane.
30  user marks important parts of the object for preservation.
40  until shape is castable:
50  -- find optimal removal directions by minimizing (2) with
      a gradient-based method.
60  -- do one step of elastoplastic deformation by
      minimizing (4).
70  if the user is not satisfied with the design:
80  -- goto 20.
90  else:
100 -- as postprocessing, minimize (2) with respect to the ver-
      tex positions with a low, nonzero tolerance.
```

## 7. Results

Depending on which points the user decides to pin to which plane, the user guides the design to achieve a variety of different results that are castable using two-piece rigid molds. Fig. 16 top and bottom left show multiple different designs achieved with the same input meshes. In these examples, the different de-

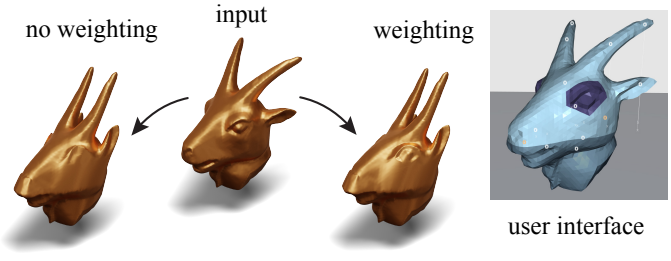



**Fig. 11. For surfaces with particularly important features, like the goat head (*center*), the elastoplastic deformation can destroy these features, as seen for the goat's eye (*left*). In such situations, the user can choose to mark the eye region as especially important. This is displayed in the user interface in purple (*far right*). The deformation will now attempt to preserve the shape of the eye as much as it can (*right*).**

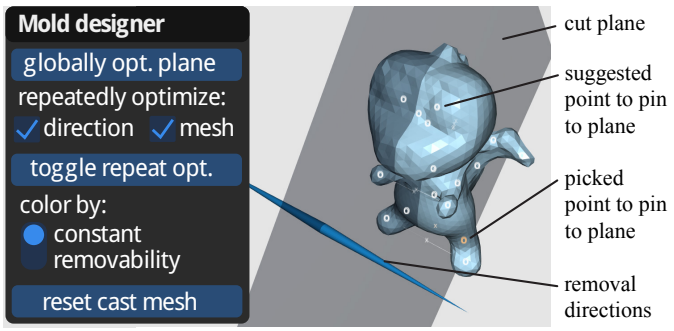



**Fig. 12. A screenshot of the design tool's user interface with controls and the most important UI elements explained.**

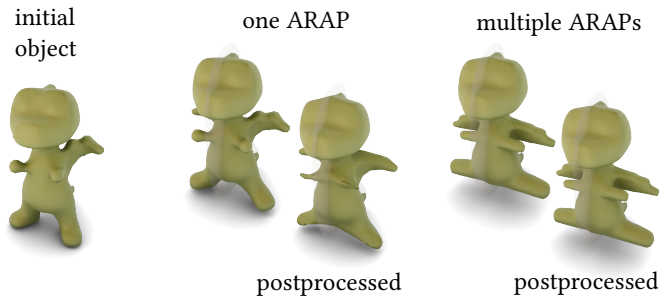



**Fig. 13. Applying postprocessing too early can lead to distorted shapes (*middle*), notice the arms and between the legs. Postprocessing should be used when the elastoplastic deformation has stopped making progress (*right*).**

signs were achieved by pinning different points to different cut planes. If you want to naively cut the bunny in the top of Fig. 16 in two with a plane, as a starting point for making a mold for it, there are multiple possible choices and no clear best choice. Two of these choices are explored in Fig. 16 and lead to different final results. Similarly, the designer of the fish in the top of Fig. 16 faces a difficult decision: would they rather preserve the curve in the fish's tail (the first design) or the exact shape of the tailfin (the second design)? It is impossible to choose both at the same time. If the tail is pinned to a vertical cut plane the designer has decided to preserve the shape of the tailfin, but give up on the curve. If a horizontal cut plane is used, the curve in the tail can be kept, but some deformation of the tail must occur. The hand in the bottom left of Fig. 16 shows a similar dichotomy between horizontal and vertical cut planes. The design tool is not limited to spherical topology of the cast, as can be seen in the bottom right of Fig. 16. However, complicated topologies can make a two-piece rigid mold impossible to manufacture. This is discussed in Section 7.1.

These designs all have one thing in common: they feature intricate geometries that are not normally found in the two-piece molds used in the production of chocolate bunnies or candy. The bunnies in the top of Fig. 16 feature separate, distinct ears that are not cut in half by the cut plane, and a large head that is not flat. The fish from the same figure features multiple distinct appendages and does not look like a bas-relief.

A small overview of runtimes can be seen in Table 1. Note that the ARAP-C time does not solely depend on mesh size, but also on how much deformation happens. The bunny mesh is larger, but it is deformed less than the hand mesh, which makes the tool faster there. These experiments were performed on a Late 2012 iMac with a 3.4 GhZ Intel Core i7 and 32GB RAM. The numbers in this table are not from the design process of the results found in this section, they are seperate experiments. For this section's results' statistics, see Appendix C. More detailed statistics on the actual designs from Figures 16 and 17 can be found in Appendix C.

We designed a number of castable shapes with our method, and computed the molds by taking simple boolean operations with meshes of cubes in a mesh editor. We designed one *multi-mold* which can be used to cast multiple objects at once, it can be seen in Fig. 3. Usually, the designs were first tested on a low-resolution mesh to find appropriate weights or points to pin, and then run on a high-resolution mesh to get the final result. Spouts were inserted manually at aesthetically pleasing locations. These spouts are used to give the foam a place to overflow to, and as a spout to insert the hot liquid candy. Some molds were additionally outfitted with wedging points to make opening the mold easier after the material has hardened. These are small slits in the mold itself, distant from the actual casts, in which a spatula or screwdriver head can be inserted (see Fig. 14).

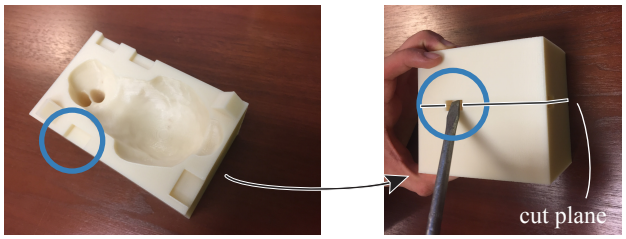



**Fig. 14. Inserting small dents into each mold at the cut plane makes it easy to wedge open a mold with a screwdriver after the material inside has hardened.**

The molds were printed in a Makerbot Replicator Z18 printer with Cool Gray filament, the layer heights varying from 0.1mm to 0.2mm and 5% infill density, as well as a dimension uPrint plus with white ABSplus thermoplastic, fine layer height and low infill density.

The resulting molds were then filled with Smooth-On FlexFoam-iT! 17 modeling foam after sealing the 3D mold with 3 layers of sealing agent and a layer of foam mold release. The results of these castings can be seen in Figs. 17 and 3. One of the molds was used to cast gummy candy in the shape of a bunny. The result of that experiment can be seen in Fig. 1. The exact recipe can be found in Appendix B.

### *7.1. Limitations*

There are limitations to the design tool. The additional rotations applied during the ARAP optimization to ensure castability cannot rotate faces for more than $\frac{\pi}{2}$. If such a rotation is needed to deform a shape in order to make it castable, the face will rotate the wrong way during the elastiplastic deformation, as the optimization finds the shortest rotation of any face

| mesh | # verts | 1st ARAP | 1st directions | optimization |
|---|---|---|---|---|
| beam | 464 | 0.355 | 0.00926 | 5.3 |
| dragon | 4290 | 1.85 | 0.0117 | 26.2 |
| hand | 7234 | 41.3 | 0.0189 | 368 |
| bunny | 14290 | 2.62 | 0.0529 | 195 |

**Table 1. Runtimes in seconds for a few input meshes, rounded to 3 significant digits. The third and fourth columns list the time for the first elastoplastic deformation step and castability energy minimization with respect to directions after the pinning to plane stage. The *optimization* column lists the time for one complete optimization run that was manually ended to apply postprocessing.**
**The shapes in this table are the same as in Fig. 8 (beam), Fig. 12 (dragon), Fig. 16 (hand), Fig. 1 (bunny).**

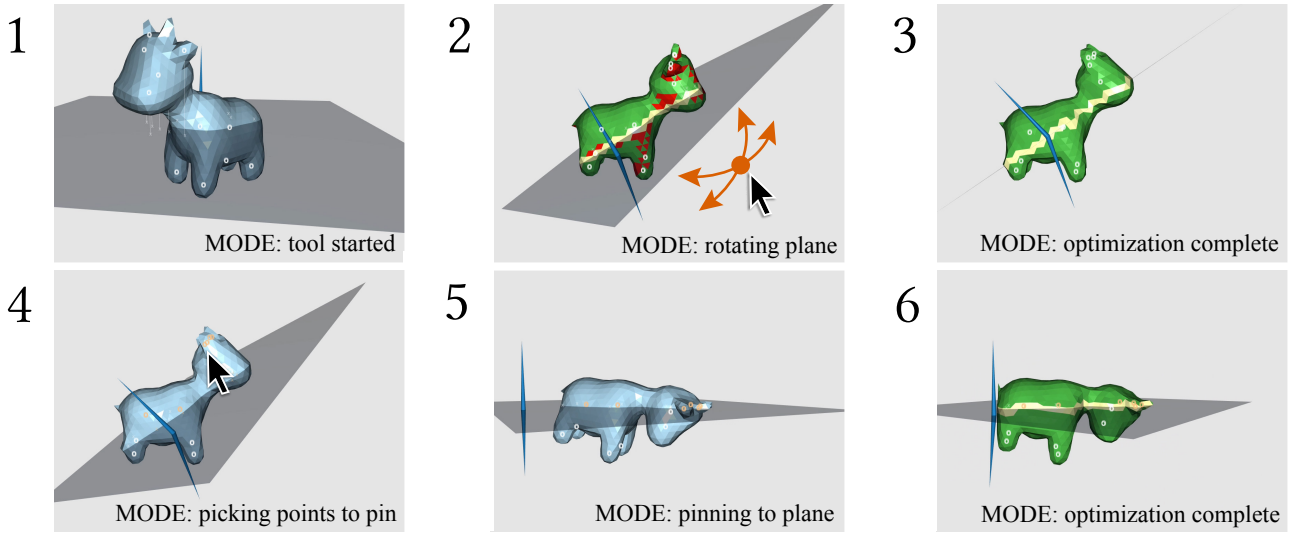



**Fig. 15. An example design using our tool, as described in Section 6.3. The user loads the tool with a cow mesh (1). The user realizes that the current state is not castable (the red parts of the mesh will collide with the mold during removal), tries to change the position of the cut plane, and then decides to just pick the globally optimal plane (2). The user runs the deformation to find a castable state (3). The user is unhappy with the resulting deformation of the horns and ears of the cow. The user decides to select some of the suggested points in the horns and ears that should be pinned to the cut plane to get a different result (4). These points are then pinned to the cut plane (5). The deformation is run again to find a final castable state (6) that the designer is satisfied with.**

to make it fulfill the removability condition (1). This means the design tool is not suitable for states with faces that violate the removability condition by more than $\frac{\pi}{2}$.

Another failure mode exists when there is no reasonable rotation at all to make the mesh castable. Sometimes the shape forms little pockets of "air" on one side of the plane. This can't be resolved at all with rotations. To resolve this, one would have to translate the shape until the pockets disappear, or perform mesh surgery to remove them.

An example of these failure modes can be seen in Fig. 18, left. In practice, they can usually be avoided by pinning the right parts of the shape to the plane, which can unravel and simplify it—suitable designs for the hand mesh can be seen in Fig. 16.

For certain topologies, it is also simply impossible to find a two-piece rigid mold. One example of this is a Hopf link of two tori. While a single torus can be cut in half so that each half is removable from a rigid mold, the fact that two tori are linked means that each individual torus cut plane bisects the other torus in a way that will not make the other one castable (see Fig. 18, right).

## 8. Conclusion & Future Work

We have characterized two-piece rigid molds joined at a plane mathematically, and introduced a design tool to generate two-piece rigid molds for arbitrary shapes. Some of these results were manufactured.

In future work, we would like to further investigate injection molding and focus on the physics of casting, similar to Nakashima et al. [15]. For injection molding, there are a lot of physics to consider that have not been discussed in this work relating to the fluid dynamics of the liquid casting material. It would also be interesting to produce actual molds for mass-manufacturing designed with our tools in an industrial setting (similar to the ones seen in Fig. 2).

It would be interesting to offer even more control over the final object, such as a bounding volume for the resulting mold, minimal feature size, etc. This would make the tool more useful for designers in practice. Volume preservation for large deformations has been successfully studied, for example, in Stomakhin et al. [33], and it would be interesting to apply their approach to our elastoplastic deformation.

The performance of the optimization scheme could be improved by providing a more optimized implementation, and by using unexplored technologies such as a GPU-based implementation. Such an improved implementation could be the focus of additional application-oriented research. An alternative optimization scheme to the one introduced in Section 5 is also of potential interest for future work. An approach optimizing all degrees of freedom simultaneously to generate better minima, or a way to fit a hard castability constraint directly into the optimization are exciting future research directions.

Additional tools for user interaction, such as merging different parts of the object to remove the air bubbles from Fig. 18, sculpting tools for general shape editing, and a bas-relief tool to flatten parts of the shape are interesting avenues for future work as well. The future tool could, for example, suggest parts of the shape to be merged, based on the detection of those regions which violate the castability condition severely, or where air bubbles occur.

Exploring different shapes that are more complicated than just planes, such as more general height fields modeled by polynomials or splines is another promising avenue for future work. This would make some of our results fabricable with less deformation from the input shape.

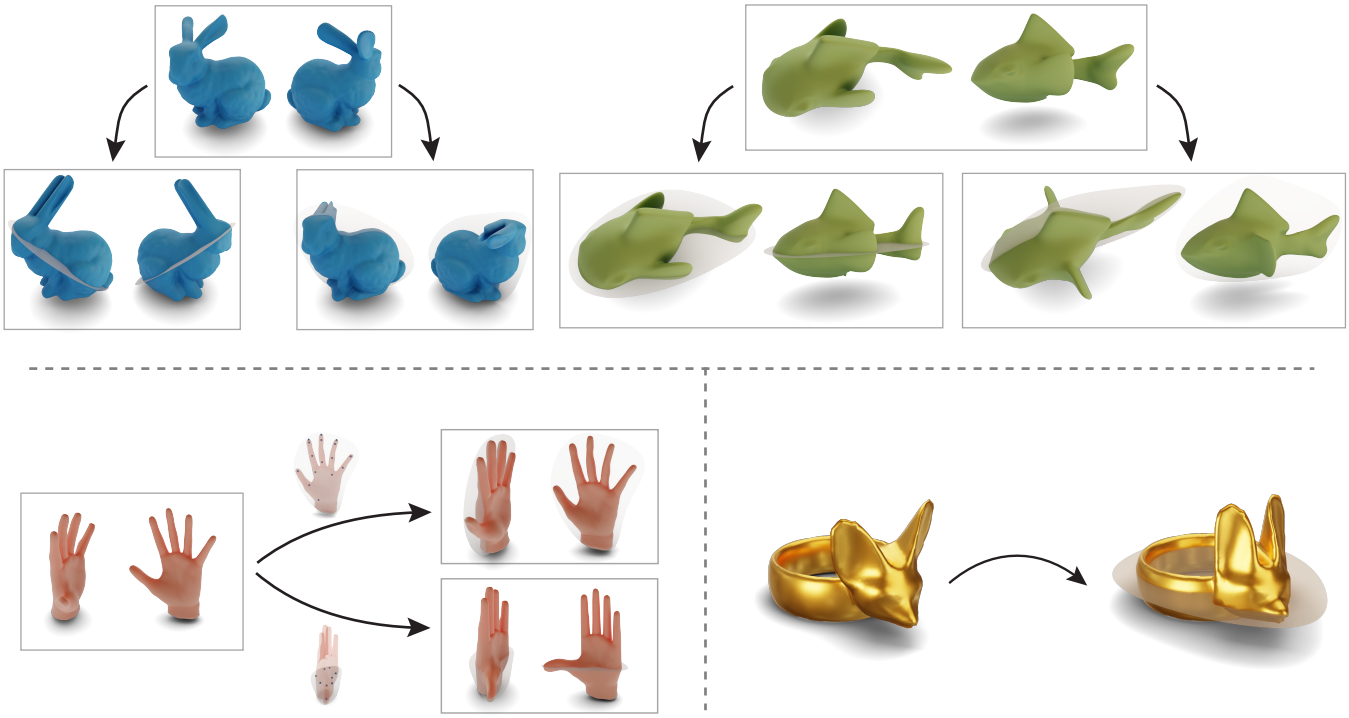


**Fig. 16. Top: two input meshes and the different castable results that can be obtained using the design tool. In both examples, the user pinned different points to different cut planes as a design decision and achieved completely different results.**
**Bottom left: two different result designs starting from the same input hand mesh. The faint pictures above the arrows show which points have been pinned to the cut plane for each design.**
**Bottom right: an example of a cast with non-spherical topology designed with the tool.**

We believe that this work provides a stepping stone towards building a variety of tools for the computational design of molds. Molds and casts are widely used in the world, and computational design in this space has the potential to have a great impact.

## 9. Acknowledgements

We would like to thank Henrique Maia and Sandra Chiritescu for helping with the fabrication process, David Levin for interesting discussions on casting and molds, Sarah Kushner, John Kanji, and Chang Xiao for proofreading, and the authors of Herholz et al. [12] and Li et al. [32] for sharing their code with us. We thank Thingiverse (goat head [34], dragon [35], fox ring [36]), libigl Jacobson et al. [37] (horse, lion, hand, kitten, armadillo, fertility), Keenan Crane Crane [38] (Spot the cow, fish), and the Stanford 3D Scanning Repository Graphics [39] (Stanford bunny) for their 3D models. We also thank Chocoladefabriken Lindt & Sprüngli AG for allowing the use of their images and videos.

This work was funded in part by NSERC Discovery Grants (RGPIN2017-05235), NSERC DAS (RGPAS-2017-50938), and the Canada Research Chairs Program. Eitan Grinspun was supported in part by a Fields Institute Fellowship. This work is funded in part by the National Science Foundation Award NSF CCF-17-17268.

## References

[1] Dall'Orto, G. Mold images in first page insert. Wikimedia Commons; 2009. URL: https://commons.wikimedia.org/wiki/File:3312_-_Athens_-_Sto%C3%A0_of_Attalus_Museum_-_Moulds_-_Photo_by_Giovanni_Dall%27Orto,_Nov_9_2009.jpg.

[2] Lindt Chocolate World, . How is the lindt goldbunny actually being made? 2016. URL: https://www.youtube.com/watch?v=9uuVuQKPdeU.

[3] Wilton, . Creating candy containers with two-piece wilton candy molds. 2014. URL: https://www.youtube.com/watch?v=g9rllaTWhd8.

[4] Campbell, J. Complete Casting Handbook. 1 ed.; Elsevier; 2011.

[5] Evan-Amos, . Chocolate-easter-bunny.jpg. Wikimedia Commons; 2011. URL: https://commons.wikimedia.org/wiki/File:Chocolate-Easter-Bunny.jpg.

[6] Platus, . Chocolate bunny image. pixabay.com; 2016. URL: https://pixabay.com/en/easter-chocolate-oster-easter-bunny-1256649/.

[7] Türk Jun., J. Schokohase für ostern, vollmilchschokolade, february 2010.jpg. Wikimedia Commons; 2010. URL: https://commons.wikimedia.org/wiki/File:Schokohase_f%C3%BCr_Ostern,_Vollmilchschokolade,_February_2010.jpg.

[8] Zhang, C, Zhou, X, Li, C. Feature extraction from freeform molded parts for moldability analysis. The International Journal of Advanced Manufacturing Technology 2010;48(1):273–282. URL: https://doi.org/10.1007/s00170-009-2273-7. doi:10.1007/s00170-009-2273-7.

[9] Chakraborty, P, Reddy, NV. Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds. Journal of Materials Processing Technology 2009;209:2464–2476.

[10] Lin, AC, Quang, NH. Automatic generation of mold-piece regions and parting curves for complex cad models in multi-piece mold design. Computer-Aided Design 2014;57:15 – 28. URL: http://www.sciencedirect.com/science/article/pii/S0010448514001420. doi:https://doi.org/10.1016/j.cad.2014.06.014.


**Fig. 17. Multiple casts fabricated via our method. The far-left column shows the input mesh; the left column shows the result mesh after applying our method; the right column shows the 3D-printed molds; the far right column shows the foam casts (dime for scale). The shapes undergo large deformations to become castable, but remain close to the input in terms of character.**


[11] Hu, R, Li, H, Zhang, H, Cohen-Or, D. Approximate pyramidal shape decomposition. ACM Transactions on Graphics 2014;33(6):213:1–213:12. URL: http://doi.acm.org/10.1145/2661229.2661244. doi:10.1145/2661229.2661244.

[12] Herholz, P, Matusik, W, Alexa, M. Approximating Free-form Geometry with Height Fields for Manufacturing. Computer Graphics Forum (Proceedings of Eurographics) 2015;34(2):239–251. doi:10.1111/cgf.12556.

[13] Lipman, Y. Bijective mappings of meshes with boundary and the degree in mesh processing. CoRR 2013;abs/1310.0955. URL: http://arxiv.org/abs/1310.0955. arXiv:1310.0955.

[14] Malomo, L, Pietroni, N, Bickel, B, Cignoni, P. Flexmolds: Automatic design of flexible shells for molding. ACM Transactions on Graphics 2016;35(6):223:1–223:12. URL: http://doi.acm.org/10.1145/2980179.2982397. doi:10.1145/2980179.2982397.

[15] Nakashima, K, Auzinger, T, Iarussi, E, Zhang, R, Igarashi, T, Bickel, B. Corecavity: Interactive shell decomposition for fabrication with two-piece rigid molds. ACM Transactions on Graphics 2018;37(4).

[16] Shamir, A, Bickel, B, Matusik, W. Computational tools for 3d printing. In: ACM SIGGRAPH 2016 Courses. 2016,.

[17] Prévost, R, Whiting, E, Lefebvre, S, Sorkine-Hornung, O. Make it stand: Balancing shapes for 3d fabrication. ACM Transactions on Graphics 2013;32(4):81:1–81:10. URL: http://doi.acm.org/10.1145/2461912.2461957. doi:10.1145/2461912.2461957.

[18] Schüller, C, Panozzo, D, Sorkine-Hornung, O. Appearance-mimicking surfaces. ACM Transactions on Graphics 2014;33(6):216:1–216:10. URL: http://doi.acm.org/10.1145/2661229.2661267. doi:10.1145/2661229.2661267.

[19] Weyrich, T, Deng, J, Barnes, C, Rusinkiewicz, S, Finkelstein, A. Digital bas-relief from 3d scenes. ACM Transactions on Graphics 2007;26(3). URL: http://doi.acm.org/10.1145/1276377.1276417. doi:10.1145/1276377.1276417.

[20] Tang, C, Bo, P, Wallner, J, Pottmann, H. Interactive design of developable surfaces. ACM Transactions on Graphics 2016;35(2):12:1–12:12. URL: http://doi.acm.org/10.1145/2832906. doi:10.1145/2832906.

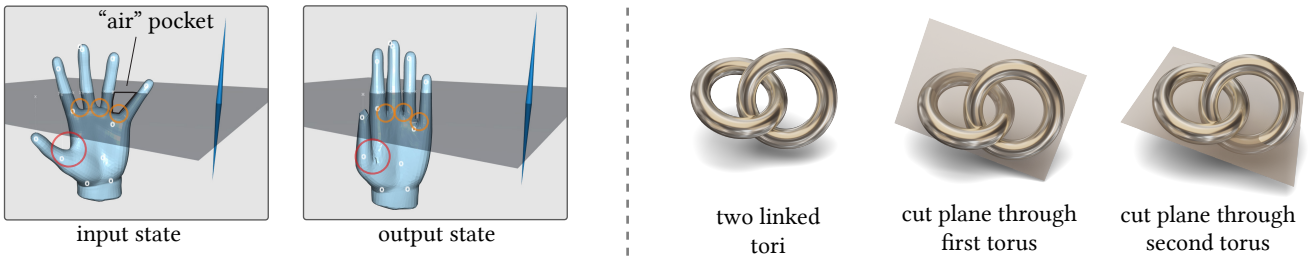



**Fig. 18. Left: the part of the hand mesh that is marked with a red circle contains faces that need to be rotated by more than $\frac{\pi}{2}$ in order to achieve a castable result. Regions marked with an orange circle can't be rotated at all to make the mesh castable, because they form little pockets on one side of the plane. Right: a link of two tori can't be made castable with a two-piece rigid mold without changing the topology. A cut plane that makes one of the individual tori castable will bisect the other one in such a way as to make it not castable.**

| mesh | # of vertices | # of suggested points | time | rotation smoothing strength | # of smoothing iterations | Hausdorff distance |
|---|---|---|---|---|---|---|
| 1st hand design | 4780 | 16 | 3m 55s | 0.05 | 1 | 0.15683 |
| 2nd hand design | 4780 | 12 | 11m 31s | 0.05 | 1 | 0.10267 |
| Fox ring | 4183 | 16 | 9m 44s | 0.01 | 3 | 0.075551 |
| 1st bunny design | 14290 | 16 | 4h 50m 39s | 0.1 | 5 | 0.28608 |
| 2nd bunny design | 14290 | 8 | 1h 35m 16s | 0.1 | 5 | 0.22522 |
| 1st fish design | 10786 | 16 | 27m 38s | 0.01 | 2 | 1.0316 |
| 2nd fish design | 10786 | 16 | 2h 21m 38s | 0.02 | 5 | 0.10777 |
| Lion | 12409 | 8 | 2h 2m 48s | 0.2 | 5 | 0.073108 |
| Spot | 11533 | 16 | 1h 58m 13s | 0.1 | 5 | 0.29717 |
| Armadillo | 14587 | 32 | 10m 47s | 0.01 | 1 | 0.19943 |

**Table C.2. A few statistics on the designs from Figs. 16 and 17. The Hausdorff distance is measured between the final result and the original and then scaled by the original's bounding box diagonal. The time is measured from the pinning until the design is complete.**


[21] Duncan, N, Yu, LF, Yeung, SK, Terzopoulos, D. Approximate dissections. ACM Transactions on Graphics 2017;36(6):182:1–182:13. URL: http://doi.acm.org/10.1145/3130800.3130831. doi:10.1145/3130800.3130831.

[22] Li, S, Mahdavi-Amiri, A, Hu, R, Liu, H, Zou, C, Van Kaick, O, et al. Construction and fabrication of reversible shape transforms. In: SIGGRAPH Asia 2018 Technical Papers. SIGGRAPH Asia '18; New York, NY, USA: ACM. ISBN 978-1-4503-6008-1; 2018, p. 190:1–190:14. URL: http://doi.acm.org/10.1145/3272127.3275061. doi:10.1145/3272127.3275061.

[23] Jacobson, A. Generalized matryoshka: Computational design of nesting objects. Computer Graphics Forum 2017;36(5):27–35. URL: https://doi.org/10.1111/cgf.13242. doi:10.1111/cgf.13242.

[24] Hu, K, Jin, S, Wang, CC. Support slimming for single material based additive manufacturing. Computer Aided Design 2015;65(C):1–10. URL: http://dx.doi.org/10.1016/j.cad.2015.03.001. doi:10.1016/j.cad.2015.03.001.

[25] Lewis, AS, Overton, ML. Nonsmooth optimization via quasi-newton methods. Mathematical Programming Series A 2013;141(1-2):135–163.

[26] Sorkine, O, Alexa, M. As-rigid-as-possible surface modeling. In: Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing. 2007, p. 109–116.

[27] Chao, I, Pinkall, U, Sanan, P, Schröder, P. A simple geometric model for elastic deformations. ACM Transactions on Graphics 2010;29(4):38:1–38:6. URL: http://doi.acm.org/10.1145/1778765.1778775. doi:10.1145/1778765.1778775.

[28] Si, H. Tetgen - a quality tetrahedral mesh generator and a 3d delaunay triangulator. 2018. URL: http://wias-berlin.de/software/index.jsp?id=TetGen.

[29] Meyer, M, Desbrun, M, Schröder, P, Barr, AH. Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege, HC, Polthier, K, editors. Visualization and Mathematics III. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-662-05105-4; 2003, p. 35–57.

[30] Jacobson, A. Algorithms and interfaces for real-time deformation of 2d and 3d shapes. Ph.D. thesis; ETH Zürich; 2013.

[31] Gramkow, C. On averaging rotations. International Journal of Computer Vision 2001;42(1):7–16. URL: https://doi.org/10.1023/A:1011129215388. doi:10.1023/A:1011129215388.

[32] Li, P, Wang, B, Sun, F, Guo, X, Zhang, C, Wang, W. Q-mat: Computing medial axis transform by quadratic error minimization. ACM Transactions on Graphics 2015;35(1):8:1–8:16. URL: http://doi.acm.org/10.1145/2753755. doi:10.1145/2753755.

[33] Stomakhin, A, Howes, R, Schroeder, D, Teran, J. Energetically consistent invertible elasticity. Eurographics Symposium on Computer Animation (SCA) 2012;.

[34] hugoelec, . Goat 5k yuanmingyuan 12 zodiac animals. Thingiverse; 2013. URL: https://www.thingiverse.com/thing:42256.

[35] macouno, . Dorus the dragon (supportless). Thingiverse; 2013. URL: https://www.thingiverse.com/thing:40635.

[36] Angus, . Fennec fox ring. Thingiverse; 2013. URL: https://www.thingiverse.com/thing:60402.

[37] Jacobson, A, Panozzo, D, et al. libigl: A simple C++ geometry processing library. 2018. URL: http://libigl.github.io/libigl/.

[38] Crane, K. Keenan's 3d model repository. 2018. URL: https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/.

[39] Graphics, S. The stanford 3d scanning repository. 2014. URL: http://graphics.stanford.edu/data/3Dscanrep/.

[40] Food Network, . Homemate gummy bears recipe. 2018. URL: https://www.foodnetwork.com/recipes/food-network-kitchen/homemade-gummy-bears-3686862.

[41] Jacobson, A. gptoolbox - geometry processing toolbox. 2018. URL: https://github.com/alecjacobson/gptoolbox.

## Appendix A. Implementing the castability energy

The castability energy (2) can be implemented by looping over all triangles in a triangle mesh and summing up the contributions from each triangle individually. Standard gradients for triangle area and normal exist and can be used here. For a triangle with vertices $\mathbf{u}, \mathbf{v}, \mathbf{w}$, normal $\mathbf{N}$ and area $A$, the gradient of the area is

$$\nabla_{\mathbf{u}} A = \frac{1}{2}\mathbf{N} \times (\mathbf{w} - \mathbf{v}), \tag{A.1}$$

and the gradient of the normal is

$$\nabla_{\mathbf{u}} N = \frac{1}{A}\left((\mathbf{w} - \mathbf{v}) \times \mathbf{N}\right)\mathbf{N}^T. \tag{A.2}$$

Note that treating every face intersecting the plane as two disjoint faces on either side of the plane does not actually require a partition of the mesh data structure or dealing with meshes more complicated than triangle meshes, it can be handled entirely within the energy calculation: a nondegenerate triangle intersected by a plane will have at least one side of the plane that only contains one triangle vertex; that side together with the plane again forms a triangle. The integral can thus be computed as an integral over the small triangle for one side, or as an integral over the original triangle minus an integral over the small triangle for the other side.

## Appendix B. Gummy candy recipe

This is the recipe that was used to create the gummy from Fig. 1. The ingredients are:

- 1 cup of Welch's concord grape juice
- 4 tablespoons of Domino's pure cane granulated sugar
- 4 tablespoons of Knox's unflavored gelatin

All ingredients are given in US customary units.

Mix the ingredients and heat in a saucepan on low heat for a short time, about a minute, until all solids are fully dissolved. Take care not to brown the sugar. Grease the mold with canola oil to prevent sticking. Fill the hot liquid into the mold. Apply some pressure to get the liquid into all nooks and crannies.

Let the candy set for about 3 hours in the fridge at 40 degrees Fahrenheit. Remove the candy from the mold and let it air dry a bit. Sugar can be used to prevent it from sticking to surfaces after removal.

This recipe is adapted from the following one: [40].

## Appendix C. Result statistics

This section of the appendix features a few statistics on the design process for the shapes in Figures 16 and 17. They can be found in Table C.2. The table features the times for the high-resolution versions that give high quality results, timings for smaller meshes with details on the exact length of different steps of the process can be found in Table 1. The Hausdorff distance was approximated using gptoolbox [41], upsampled twice before calculating. For the final vertex position optimization mentioned in Algorithm 1, the tolerance $\tau = 0.02$ was used.