Interactive Design of Castable Shapes Using Two-Piece Rigid Molds

ODED STEIN, Columbia University
ALEC JACOBSON, University of Toronto
EITAN GRINSPUN, Columbia University

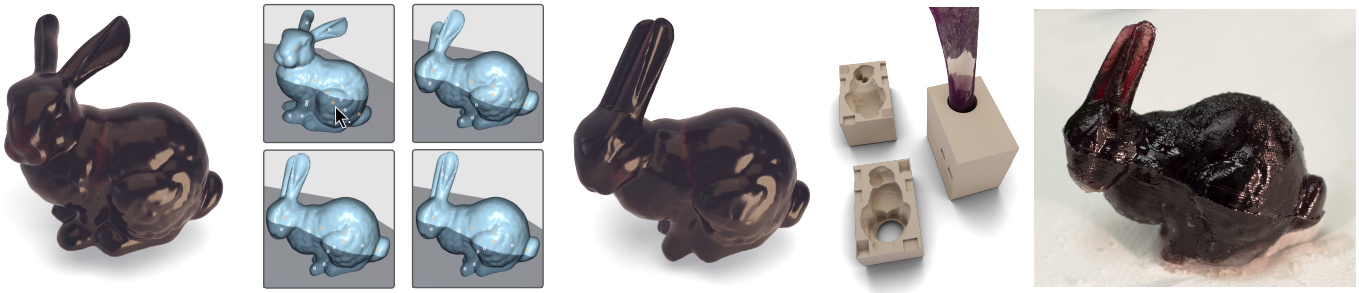


Fig. 1. Mold design and fabrication of bunny gummy candy. The original mesh (*far left*) is manipulated with the design tool (*left*) to produce a shape that is castable (*center*) with a two-piece rigid mold (*right*). The resulting shape is then fabricated as candy (*far right*).

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 form 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 than 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.

(c) Giovanni Dall'Orto, wikimedia.org

In the food industry the use of molds is widespread, for example to manufacture chocolate sculptures [Lindt Chocolate World 2016] or candy [Wilton 2014]. In metalworking, casting is a popular technique to manufacture many metal structures [Campbell 2011]. Of particular interest is casting with rigid, reusable molds. These kinds of molds are well suited for parallel fabrication, where a large amount 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 geometries (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 [Evan-Amos 2011; Platus 2016; Türk Jun. 2010]).

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 wild, such as in chocolate bunnies or candies.

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.

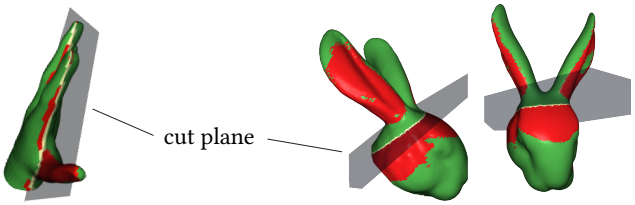
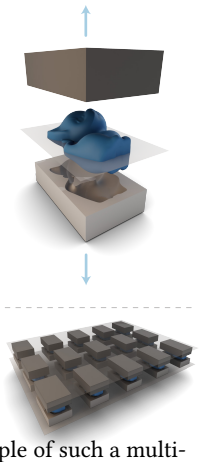


Fig. 4. In general, an object can't be realized with any two-piece rigid mold. In these examples, the red portion of the objects (according to the removability condition (1)) will collide with the mold during removal. In this example, the directions are normal to the cut plane.



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 translation direction does not necessarily need to be normal to that plane. 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.

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 poisition 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 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 the failure of a surface to be castable with a two-piece rigid mold;
- describe an elastioplastic 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. [2010] present a tool to analyze the castability of general objects for rigid injection molding. Chakraborty and Reddy [2009] present a tool to generate a two-piece rigid mold and its parting directions for a shape, without modifying the shape. [Lin and Quang 2014] propose an approach to automatically generate multi-piece rigid molds for CAD models. Hu et al. [2014] decompose objects into pyramidal shapes which are moldable. Herholz et al. [2015] 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 [Herholz et al. 2015, 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. [2015] uses

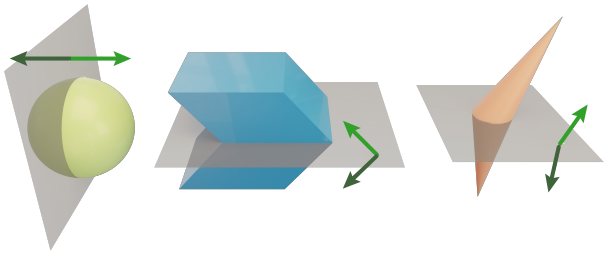


Fig. 5. The removal directions for the cast object don't have to be the plane normals (*left*). In fact, they can differ from it significantly (*center, right*). All objects in this figure can be produced with two-piece rigid molds where the two pieces meet at a plane. The green arrows indicate the removal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

[Lipman 2013] to formulate conditions for castability. A similar condition is also used in this work. Malomo et al. [2016] 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. [2018] 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. [2016]. A few relevant examples are listed here.

Prévost et al. [2013] 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. [2014]. 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 approches for designing bas-reliefs can be found, for example, in Weyrich et al. [2007]. Tang et al. [2016] 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.

Jacobson [2017] cuts shapes so they can be recursively nestled 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 [2015] 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 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. [2015] 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 [Herholz et al. 2015, Fig. 14], an issue which does not exist for two-piece molds.

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,

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

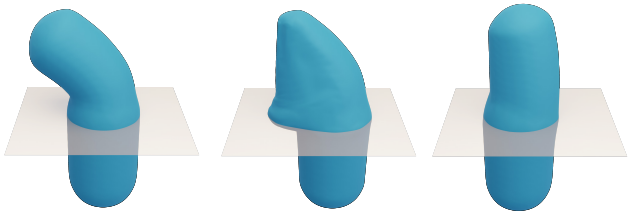


Fig. 6. Merely minimizing the castability energy (*center*) using a gradient-based method does not give natural-looking results. To preserve the the look and feel of an object, an elastoplastic deformation can be applied to a volumetric representation of it (*right*).

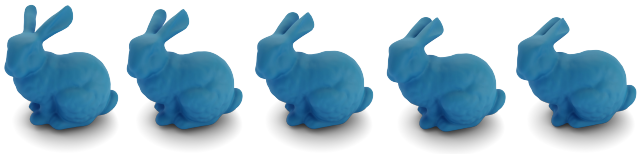


Fig. 7. From left to right, different stages in the elastoplastic optimization of the bunny. At every frame, the ARAP procedure was restarted.

where Ω 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.1.

States with zero castability energy are guaranteed to be castable with a certain tolerance τ. 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 subgradient descent as well as L-BFGS are used with the line search described in Lewis and Overton [2013] for local optimization of (2) with respect to the cut plane position and the removal directions.

5.2 Elastoplastic deformation

However, using a gradient-based method to minimize 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 (ARAP) method [Sorkine and Alexa 2007] to a tetrahedral mesh whose boundary is our surface (see Chao et al. [2010]). There are multiple options for generating tetrahedral meshes from surfaces. We use tetgen [Si 2018]. 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.

We treat every tetrahedron of the tetrahedral mesh as a cell that should be transformed as rigidly as possible, and use a local-global solver similar to Sorkine and Alexa [2007] from there. Let T be the set of all tetrahedra. If every tetrahedron $t \in T$ is rigidly deformed, then there is a rotation matrix R_t such that:

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

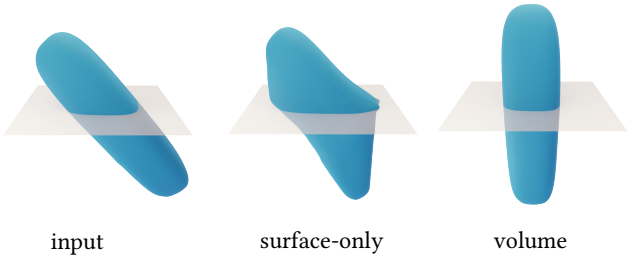


Fig. 8. Deforming a bar (*left*) with ARAP to make it removable with the removal directions normal to the separation plane. Deforming only the surface (*center*) (similar to the postprocessing used by Herholz et al. [2015]) skews the bar significantly, while deforming the entire volume (*right*) merely rotates it.

where t' is the deformed tetrahedron.

In most cases it will be impossible to find a completely rigid deformation that makes a mesh castable. Because of that the deformation is only required to be as rigid as possible, i.e., to minimize

$$E_A(V', (R_t)_t) = \sum_{t\in T} \sum_{p,q\in t} w_{pq}^t \| \mathbf{p}_{n+1} - \mathbf{q}_{n+1} - R_t\,(\mathbf{p}_n - \mathbf{q}_n)\|^2 \tag{4}$$

where w_{pq}^t are tetrahedral cotangent weights [Jacobson 2013; Meyer et al. 2003]. Sorkine and Alexa [2007] show that this energy can be optimized using a local-global approach, where the rotations are computed by forming the covariance matrix for each cell,

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

where $\mathbf{e}_{pq} = \mathbf{p} - \mathbf{q}$. This is followed by a singular value decomposition $S_t = U_t \Sigma_t V_t^{\mathsf{T}}$. Discarding the singular values then yields a rotation $R_t = V_t U_t^{\mathsf{T}}$ (this might require changing the sign of one column in U to ensure positive determinant).

The vertex positions can be computed by minimizing the global energy in (4) with respect to the vertex positions. The optimization with respect to the rotations and with respect to the vertex positions can then be repeated until convergence.

5.3 Enforcing castability

So far we have completely disregarded castability. Directly enforcing castability on the result leads to a complicated constrained optimization problem. Because of that, we choose to enforce castability in a weaker way during the rotational step of the ARAP optimization. For a boundary face of the tetrahedral mesh, we can check if the removability condition (1) is fulfilled or not. If it is fulfilled, then there is nothing to do. If it is not fulfilled, we rotate the tetrahedron that this face belongs to *until the removability condition is fulfilled.* We rotate around the axis $\mathbf{d} \times \mathbf{N}$, where $\mathbf{d}$ is the removal direction and $\mathbf{N}$ is the normal of that boundary face. The rotation is chosen because it is the smallest rotation that leads to a face fulfilling (1). This additional rotation is applied after the rotation R_t which was computed in the local ARAP step, and there may be multiple rotations if a tetrahedron has multiple boundary faces violating the removability condition. For faces that lie on both sides of the cut

plane, we don't enforce removability — a postprocessing step will take care of them.

This may converge without the castability condition fully enforced. For this reason, this ARAP procedure is applied repeatedly: once converged, the initial configuration is set to the new configuration, and the ARAP local-global optimization is run again. This results in an elastoplastic deformation. The deformation is not simply *elastic* — since we run multiple elastic deformation steps that are each elastic, we end up *plastically* deforming the object, similar to shaping clay or play dough. This gives us a castable mesh that is implicitly related to the original mesh through its 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.

A similar approach is used for post-processing in Herholz et al. [2015], 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:

$$R_t^* = \text{blend}\left((1-\gamma), R_t; \frac{\gamma}{n}, R_{u_1}; \cdots; \frac{\gamma}{n}, R_{u_n}\right) \tag{6}$$

where γ is a parameter that controls how much to smooth.

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 [2001]).

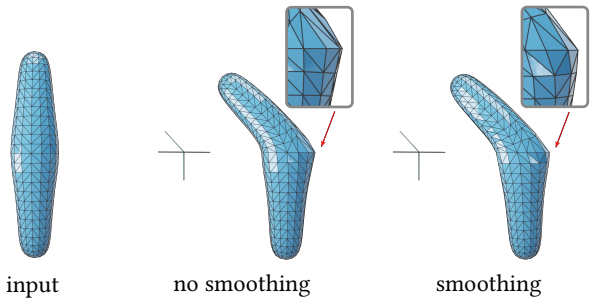


Fig. 9. Smoothing rotations during ARAP optimization can help produce smoother result meshes.

6 THE DESIGN TOOL

With the continuously optimizable castability energy, the gradient-based optimization and the elastoplastic deformation we now have all 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). 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 Figs. 15, 16).

The parts of the tetrahedral mesh that are pinned to the plane don't 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 [Li et al. 2015] to compute a

Fig. 10. Taking an arbitrary input (*left*) and rotating faces to better conform to the removability condition could lead to some extreme deformations (*center*). Pinning a few key points to the cut plane (*right*) will avoid these deformations.

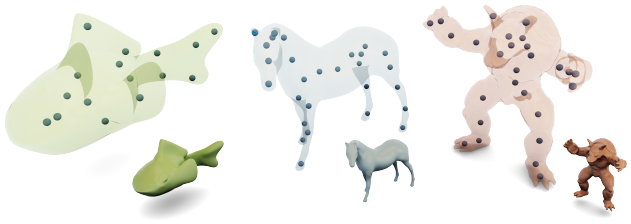


Fig. 11. Examples of some surfaces, and the points on their medial graph, generated with Q-MAT [Li et al. 2015]. The points the user wants to pin are often a subset of these points, which makes this a very useful heuristic.

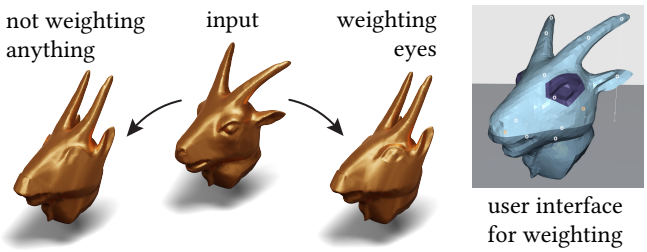


Fig. 12. Designing a mold for a goat head shape (*center*). Running the elastoplastic deformation as such destroys the shape of the goat's eye (*left*). As a result, the user chooses to mark the eye region as especially important, which is displayed in purple (*far right*). Running the deformation will attempt to preserve the shape of the eyeball as much as possible (*right*).

medial graph of the input surface. The points on this medial graph are then used to suggest a small amount 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 (ref. Fig. 11). 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 farthest away from the two points where the ray hits 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 tets, the value of w_{pq}^t 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. 12.

6.3 Putting it all together

The user has multiple ways of interacting with the tool (see Fig. 13). 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.

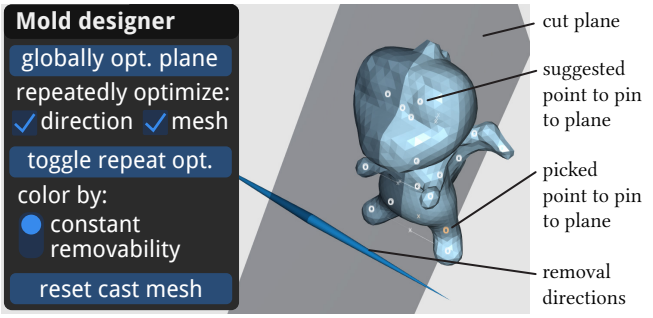


Fig. 13. A screenshot of the user interface of the design tool with controls and the most important UI elements explained.

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 postprocessign, minimize (2) with respect to the vertex
      positions with a low, nonzero tolerance.
```

The user can toggle the continuous elastoplastic deformation with or without continuous optimization of the castability energy with respect to removal directions. 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 nonzero tolerance.

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. 14. The main interaction loop is presented in pseudocode form in Algorithm 1.

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

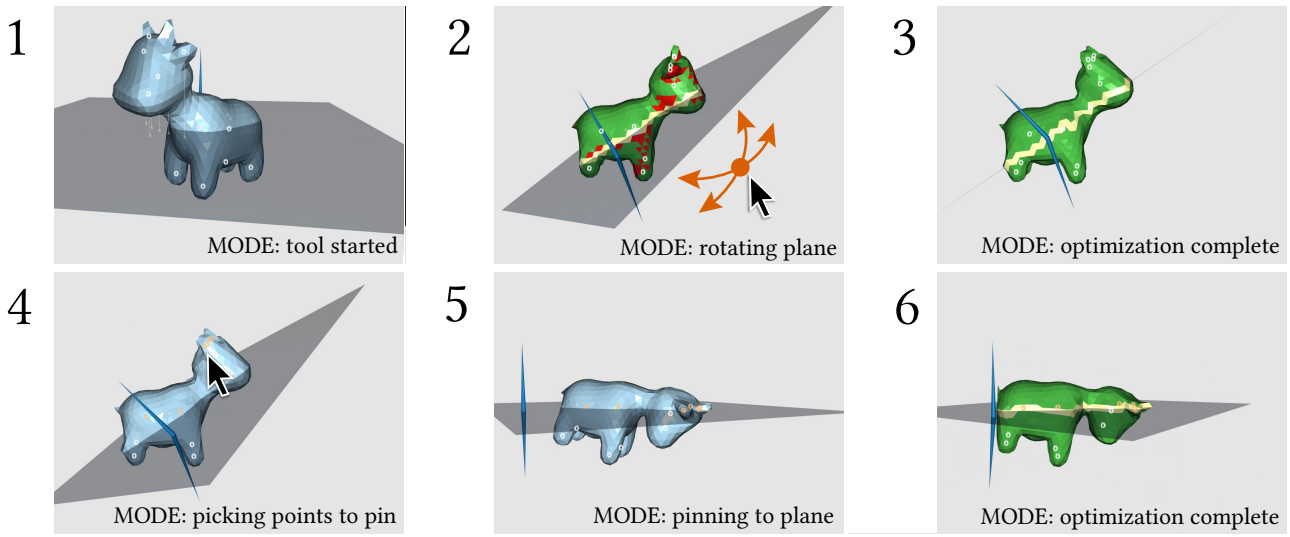


Fig. 14. An example design using our tool, as described in Section 6.3. The user loads the tool with a cow mesh (1). The user realizes that the current state is not castable (the red parts of the mesh will collide with the mold during removal), tries to change the position of the cut plane, and then 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.

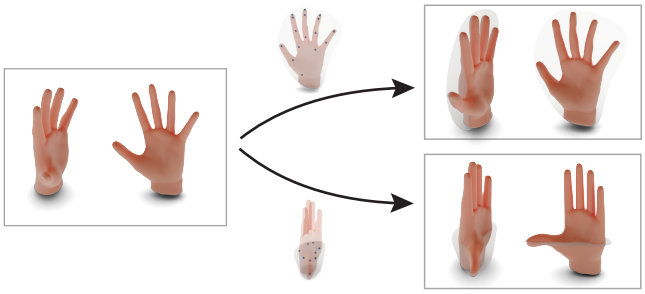


Fig. 15. Two different result designs starting from the same input hand mesh. The faint pictures above the arrows show which points have been pinned to the cut plane for each design.

mesh	# vertices	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.

are castable using two-piece rigid molds. Fig. 16 and Fig. 15 show multiple different designs achieved with the same input meshes. In these examples the different designs were achieved by pinning different points to different cut planes. For the bunny in Fig. 16, there are two ways to cut the bunny that make it almost castable, except for the ears, the tail, and a part of the legs. Each of these cuts leads to a different final design. Similarly, the designer of the fish in 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 Fig. 15 shows a similar dichotomy between horizontal and vertical cut planes.

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 Fig. 16 features 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.

We designed a number of molds 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

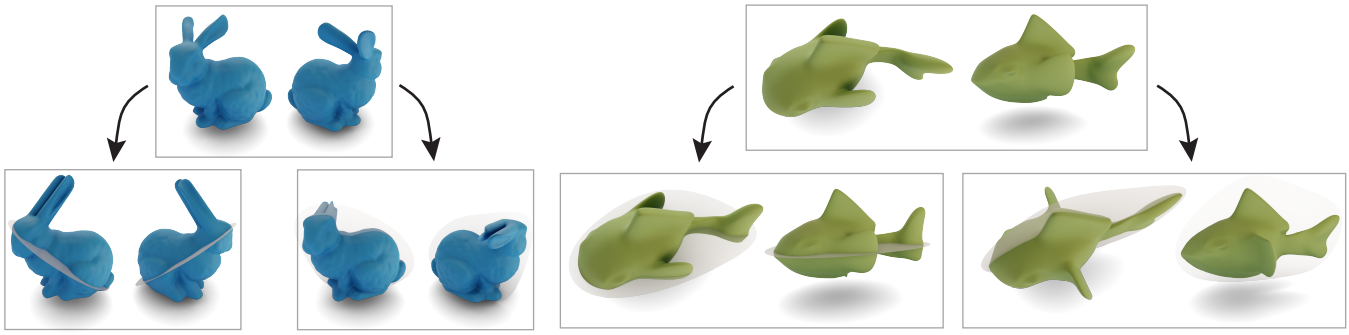


Fig. 16. Two input meshes and the different castable results than can be obtained using the design tool. In both examples the user pinned different points to different cut planes as a design decision and achieved completely different results.

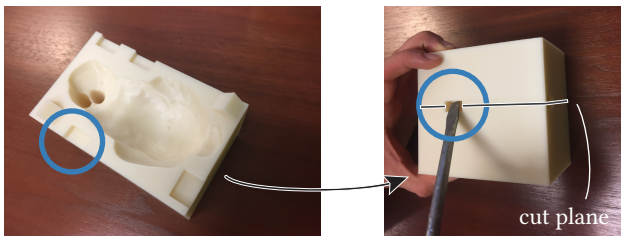


Fig. 17. Inserting small dents into each mold at the cut plane makes it easy to wedge open a mold with a screwdriver after the material inside has hardened.

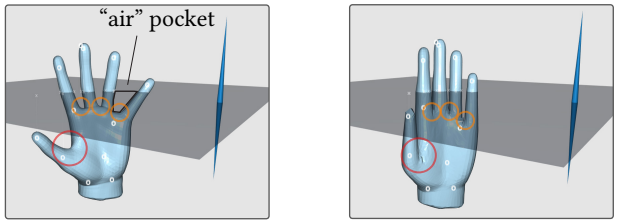


Fig. 18. The part of the hand mesh that is marked with a red circle contains faces that need to be rotated by more than $\frac{\pi}{2}$ in order to achieve a castable result. Regions marked with an orange circle can't be rotated at all to make the mesh castable, because they form little pockets on one side of the plane.

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. 17).

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. 19 and 3. One of the molds was used to cast gummy candy in the shape of a bunny. The result for that experiment can be seen in Fig. 1. The exact recipe can be found in Appendix A.2.

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 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. 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. 15.

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. 20).

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

Fig. 19. 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.

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. [2018]. 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 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. [2012], and it would be interesting to apply their approach to our elastoplastic deformation. 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

two linked tori | cut plane through first torus | cut plane through second torus

Fig. 20. 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.

detection of those regions which violate the castability condition severely, or where air bubbles occur. 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).

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 for helping with the fabrication process, David Lavin for interesting discussions on casting and molds, Sarah Kushner and John Kanji for proofreading, and the authors of Herholz et al. [2015] and Li et al. [2015] for sharing their code with us.

This work is funded in part by the National Science Foundation Award NSF CCF-17-17268.

REFERENCES

2015. Support Slimming for Single Material Based Additive Manufacturing. *Comput. Aided Des.* 65, C (Aug. 2015), 1–10. DOI: http://dx.doi.org/10.1016/j.cad.2015.03.001

John Campbell. 2011. *Complete Casting Handbook* (1 ed.). Elsevier.

Pritam Chakraborty and Nallagundla Venkata Reddy. 2009. Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds. 209 (03 2009), 2464–2476.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A Simple Geometric Model for Elastic Deformations. *ACM Trans. Graph.* 29, 4, Article 38 (July 2010), 6 pages. DOI: http://dx.doi.org/10.1145/1778765.1778775

Evan-Amos. 2011. Chocolate-Easter-Bunny.jpg. Wikimedia Commons. (April 2011). https://commons.wikimedia.org/wiki/File:Chocolate-Easter-Bunny.jpg

Food Network. 2018. Homemate Gummy Bears Recipe. (2018). https://www.foodnetwork.com/recipes/food-network-kitchen/homemade-gummy-bears-3686862

Claus Gramkow. 2001. On Averaging Rotations. *International Journal of Computer Vision* 42, 1 (01 Apr 2001), 7–16. DOI: http://dx.doi.org/10.1023/A:1011129215388

Philipp Herholz, Wojciech Matusik, and Marc Alexa. 2015. Approximating Free-form Geometry with Height Fields for Manufacturing. *Computer Graphics Forum (Proc. of Eurographics)* 34, 2 (2015), 239–251. DOI: http://dx.doi.org/10.1111/cgf.12556

Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. 2014. Approximate Pyramidal Shape Decomposition. *ACM Trans. Graph.* 33, 6, Article 213 (Nov. 2014), 12 pages. DOI: http://dx.doi.org/10.1145/2661229.2661244

Alec Jacobson. 2013. *Algorithms and Interfaces for Real-Time Deformation of 2D and 3D Shapes.* Ph.D. Dissertation. ETH Zürich.

Alec Jacobson. 2017. Generalized Matryoshka: Computational Design of Nesting Objects. *Comput. Graph. Forum* 36, 5 (Aug. 2017), 27–35. DOI: http://dx.doi.org/10.1111/cgf.13242

Adrian S. Lewis and Michael L. Overton. 2013. Nonsmooth optimization via quasi-Newton methods. *Math. Program. Ser. A* 141 (2013), 135–163.

Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. 2015. Q-MAT: Computing Medial Axis Transform By Quadratic Error Minimization. *ACM Trans. Graph.* 35, 1, Article 8 (Dec. 2015), 16 pages. DOI: http://dx.doi.org/10.1145/2753755

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

Lindt Chocolate World. 2016. How is the LINDT GoldBunny actually being made? (2016). https://www.youtube.com/watch?v=9uuVuQKPdeU

Yaron Lipman. 2013. Bijective Mappings Of Meshes With Boundary And The Degree In Mesh Processing. *CoRR* abs/1310.0955 (2013). arXiv:1310.0955 http://arxiv.org/abs/1310.0955

Luigi Malomo, Nico Pietroni, Bernd Bickel, and Paolo Cignoni. 2016. FlexMolds: Automatic Design of Flexible Shells for Molding. *ACM Trans. Graph.* 35, 6, Article 223 (Nov. 2016), 12 pages. DOI: http://dx.doi.org/10.1145/2980179.2982397

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. 2003. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and Mathematics III*, Hans-Christian Hege and Konrad Polthier (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 35–57.

Kazutaka Nakashima, Thomas Auzinger, Emmanuel Iarussi, Ran Zhang, Takeo Igarashi, and Bernd Bickel. 2018. CoreCavity: Interactive Shell Decomposition for Fabrication with Two-Piece Rigid Molds. *To appear in ACM Transactions on Graphics* 37, 4 (2018).

Platus. 2016. Chocolate Bunny Image. pixabay.com. (March 2016). https://pixabay.com/en/easter-chocolate-oster-easter-bunny-1256649/

Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Trans. Graph.* 32, 4, Article 81 (July 2013), 10 pages. DOI: http://dx.doi.org/10.1145/2461912.2461957

Christian Schüller, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Appearance-mimicking Surfaces. *ACM Trans. Graph.* 33, 6, Article 216 (Nov. 2014), 10 pages. DOI: http://dx.doi.org/10.1145/2661229.2661267

Ariel Shamir, Bernd Bickel, and Wojciech Matusik. 2016. Computational Tools for 3D Printing. In *ACM SIGGRAPH 2016 Courses.*

Hang Si. 2018. TetGen - A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator. (2018). http://wias-berlin.de/software/index.jsp?id=TetGen

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing.* 109–116.

A. Stomakhin, R. Howes, D. Schroeder, and J. Teran. 2012. Energetically Consistent Invertible Elasticity. *Eurographics Symposium on Computer Animation (SCA)* (2012).

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive Design of Developable Surfaces. *ACM Trans. Graph.* 35, 2, Article 12 (Jan. 2016), 12 pages. DOI: http://dx.doi.org/10.1145/2832906

Josef Türk Jun. 2010. Schokohase für Ostern, Vollmilchschokolade, February 2010.jpg. Wikimedia Commons. (February 2010). https://commons.wikimedia.org/wiki/File:Schokohase_f%C3%BCr_Ostern,_Vollmilchschokolade,_February_2010.jpg

Tim Weyrich, Jia Deng, Connelly Barnes, Szymon Rusinkiewicz, and Adam Finkelstein. 2007. Digital Bas-relief from 3D Scenes. *ACM Trans. Graph.* 26, 3, Article 32 (July 2007). DOI: http://dx.doi.org/10.1145/1276377.1276417

Wilton. 2014. Creating Candy Containers with Two-Piece Wilton Candy Molds. (2014). https://www.youtube.com/watch?v=g9rllaTWhd8

Chunjie Zhang, Xionghui Zhou, and Congxin Li. 2010. Feature extraction from freeform molded parts for moldability analysis. *The International Journal of Advanced Manufacturing Technology* 48, 1 (01 Apr 2010), 273–282. DOI: http://dx.doi.org/10.1007/s00170-009-2273-7

A APPENDIX

A.1 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

$$\nabla_{\mathbf{u}} A = \frac{1}{2}\mathbf{N} \times (\mathbf{w} - \mathbf{v}), \tag{7}$$

and the gradient of the normal is

$$\nabla_{\mathbf{u}} N = \frac{1}{A}\left((\mathbf{w} - \mathbf{v}) \times \mathbf{N}\right)\mathbf{N}^T. \tag{8}$$

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.

A.2 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 this one: [Food Network 2018].