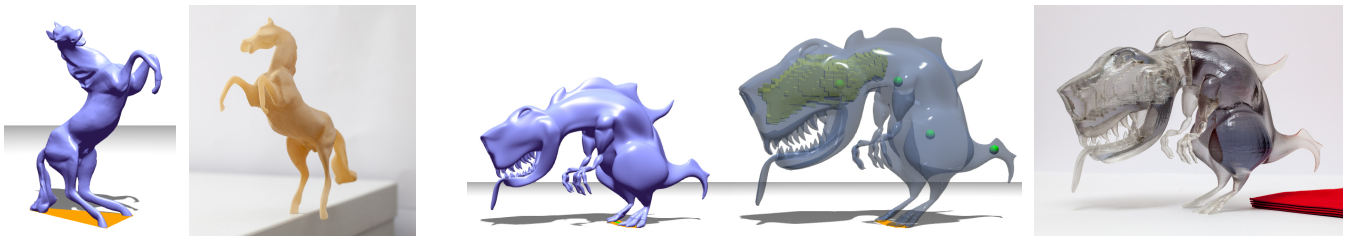
Make It Stand: Balancing Shapes for 3D Fabrication

Romain Prévost[1] Emily Whiting[1] Sylvain Lefebvre[2] Olga Sorkine-Hornung[1]

[1]ETH Zurich [2]INRIA



(a) (b) (c) (d)

Figure 1: *Our algorithm iterates between carving and deformation to reach the final, balanced result. (a) The original horse model does not stand on its hind legs and requires using the tail as a third support. (b) Our approach deforms the horse to make it stand on a* single *hind leg. (c,d) The user scaled up the head of the T-Rex. Our optimizer succeeds in finding the delicate balance of a massive head and body on a tiny base of support. It deforms and carves the model (yellow region visible by transparency) to precisely position the center of mass.*

Abstract

Imbalance suggests a feeling of dynamism and movement in static objects. It is therefore not surprising that many 3D models stand in impossibly balanced configurations. As long as the models remain in a computer this is of no consequence: the laws of physics do not apply. However, fabrication through 3D printing breaks the illusion: printed models topple instead of standing as initially intended. We propose to assist users in producing novel, properly balanced designs by interactively deforming an existing model. We formulate balance optimization as an energy minimization, improving stability by modifying the *volume* of the object, while preserving its surface details. This takes place during interactive editing: the user cooperates with our optimizer towards the end result. We demonstrate our method on a variety of models. With our technique, users can produce fabricated objects that stand in one or more surprising poses without requiring glue or heavy pedestals.

CR Categories: I.3.7 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; Physically based modeling

Keywords: Static equilibrium, structural stability, 3D printing, optimization, interactive shape modeling

Links: DL PDF WEB VIDEO

1 Introduction

Balance is a delicate compromise between shape, weight and pose. It is therefore difficult to visually assess whether a given object is properly balanced and will stand in a stable, up-right position. Artists, designers and architects use this to their advantage to produce surprising and elegant designs that seem to defy gravity [Smithson 1968; von Meiss 1990; Kedziora 2011]. A well-known design principle to this effect, *asymmetric balance* [Lauer and Pentak 2011], consists in achieving balance by contrasting sizes and weights on either side of a composition. When considering physical objects, this process is not only concerned with aesthetics, but also with structural soundness: the weights of each part must exactly compensate each other, balancing the design in its intended, stable pose.

With the advent of 3D printing technologies, it becomes very simple to produce physical realizations of 3D models. Unfortunately, they most often fail to stand, making it mandatory to glue the printed objects onto a heavy pedestal. The delicate process of balancing, already difficult with physical objects, is very challenging when manipulating geometry in a 3D modeler. There is usually no indication of gravity, support or weight. Volumes are only represented by their boundaries, making it tedious to exploit degrees of freedom such as carving the inside of an object to change its weight distribution.

In this paper, we propose to assist users in modifying existing 3D models to create novel, balanced designs. Using our approach, the user interactively edits a shape and cooperates with our optimizer towards the final result. The optimizer constantly improves the design to ensure that, after printing, it will stand on its intended basis with the chosen orientation. This is especially well suited for the modeling of surprising asymmetric balance configurations, such as the printed horse model in Figure 1.

The input to our algorithm is a surface mesh representing a solid object, a number of desired contact points and the desired orientation (i.e., gravity direction). We exploit two main degrees of freedom when modifying the model: our algorithm *carves* and *deforms* the object to improve its equilibrium. We seek to minimize deviations from the intended shape, and therefore the algorithm searches for a compromise between removing matter from the interior and de-

forming the surface. We explore two modes of balancing: (*i*) standing on a flat surface, and (*ii*) orientation of suspended objects. In both cases the user specifies as input the base of support or the attachment points. Our method enables to simultaneously optimize for several desired modes, e.g. several standing orientations.

Contributions. Our approach redistributes weight by jointly optimizing for the interior and the surface shape of an object. We approximate the interior volume with a voxel grid, each voxel being tagged as full or empty. We propose a simple yet effective algorithm to quickly carve voxels to improve stability given the current shape. We express shape deformation as a least-squares optimization, preserving shape details while moving the center of gravity closer to its stable configuration. Our optimizer automatically performs blended translations, rotations and scalings of the shape's *volume* to improve the design. An important aspect of our formulation is the proper derivation of the change in center of mass due to the redistribution of weight.

We demonstrate our approach on a variety of examples, most of which we printed as physical objects. As can be seen throughout the paper and the accompanying video, our results provide surprising yet stable configurations. These models all stand on their own without glue or magnets.

2 Related work

The process of turning a virtual model into a physical object is not straightforward, and several design constraints have to be followed. This led researchers to propose methods automatically checking whether an object can be fabricated, as well as approaches to assist the user throughout the fabrication process. Telea and Jalba [2011] study the printability of an object by considering a voxel discretization of its volume. The system detects potential issues such as too thin features, bridges and tunnels. 3D printers can only print relatively small objects; Luo et al. [2012] propose an algorithm that automatically cuts a model into smaller, printable parts that can be later assembled. Beyond 3D printing, alternative tools for rapid prototyping have been introduced, such as hand-guided 2D cutting tools [Rivers et al. 2012b] and sculpting with visualization guidance [Rivers et al. 2012a]. While we focus on automated 3D printers, our stability optimization is applicable to alternative fabrication materials and technologies.

Rather than post-processing models, a number of recent papers further assist the user in modeling shapes that would be physically sound once fabricated. In SketchChair [Saul et al. 2011], the user sketches a profile which is automatically extruded into a chair that can be fabricated from assembled laser-cut wood panels. The user is presented with a real-time simulation of the physics of the chair, so that she can visualize any equilibrium or stability issues with the final model. Umetani et al. [2012] extend the principle of SketchChair and propose an interface automatically *suggesting* to the user how to modify her current design so that it would meet specific fabrication requirements, such as stability and durability. The system follows the gradient of the objective function to reach valid regions of the configuration space.

The stress–relief algorithm [Stava et al. 2012] automatically modifies a shape to reinforce its physical realization. The approach considers fragile areas of the model and performs three types of corrections: thickening tube-like sections of the mesh, hollowing parts to reduce strain on fragile parts, and adding struts to support remaining excessive weight. These improvements are considered in sequence. The algorithm does not optimize for stability: It is assumed that a balanced model is provided. It however rejects any change to the model that would compromise its balance. Therefore, a model with a delicate balance is likely to pose a challenge. In contrast, our method iterates between carving and deformation to reach equilibrium constraints. We exclude the use of struts or heavy pedestals to keep the shape as close as possible to the original intent of the modeler. Note that we neglect internal stresses and joint mechanics, assuming a solid rigid material of sufficient strength, but the approach of Stava et al. could be combined with ours to improve part robustness. Fabrication of articulated models such as posable characters [Bächer et al. 2012; Calì et al. 2012] or mechanical toys [Zhu et al. 2012] must handle structural constraints related to friction: with too much resistance the parts will not move, but insufficient friction will cause joints to fail under self-weight when setting poses. We deal with a different aspect of the stability problem to prevent global toppling of the printed model.

While not targeted at fabrication, several approaches propose to optimize the pose or the shape of models under mechanical objectives. Static stability of character models was investigated in Mesh-Puppetry [Shi et al. 2007]. This approach targets plausible rather than feasible poses; in particular, the center of mass is approximated by a sum of masses located at the barycenter of the skeleton bones. The underlying formulation of the method is quite different from ours, since Mesh-Puppetry optimizes for joint angles and skinning weights, while we optimize for the inner and outer surface through deformation and carving. Derouet-Jourdan et al. [2010] optimize the parameters of user-drawn curves so that once simulated under gravity, their shape matches the user sketch. In architectural geometry, stability optimization has been studied for the design of self-supporting structures. Whiting et al. [2009; 2012] and Vouga et al. [2012] optimize structures composed of rigid blocks under the constraint of compression-only static equilibrium. These methods consider stability of an assemblage of parts consisting of simple solids; in contrast, we deform meshes of arbitrary complexity and additionally carve interior voids. More generally, the field of shape optimization also considers the problem of modifying shapes to enforce a number of mechanical properties [Allaire 2001]. These approaches however rely on computationally expensive iterative finite element simulations which do not afford for interactive feedback [Stava et al. 2012].

3 Method overview

Given an input surface representing a solid model, our goal is to deform the volume such that the center of mass will reach a feasible target. We manipulate the volume in two ways: we *carve* the model interior to create inner voids, and we *deform* the original surface. As mentioned, we support two balancing modes: shapes that are meant to stand on a flat surface, and the resting position of shapes suspended by a string. In both cases the problem amounts to modifying the volume so as to reach a target center of mass. Our method also supports multiple targets, as described later.

Standing mode. Feasible standing positions for a model are assessed by the location of the center of mass $\mathbf{c}$ in relation to the intended base of support. The base of support is defined by the convex hull of all the points of the mesh touching the ground, which we refer to as the *support polygon*. This is determined by the set of the lowest mesh vertices with respect to the gravity direction. Optionally we flatten faces surrounding the support, within a threshold, in order to enlarge support polygons which are too small to result in a stable configuration.

To maintain static equilibrium, the center of mass of an object must project along the gravity direction *into* the support polygon. However, while positioning the centroid at the boundary of the support polygon is sufficient for feasibility, we further optimize the stability by setting the ideal center of mass $\mathbf{c}^\star$ to project within a "safe

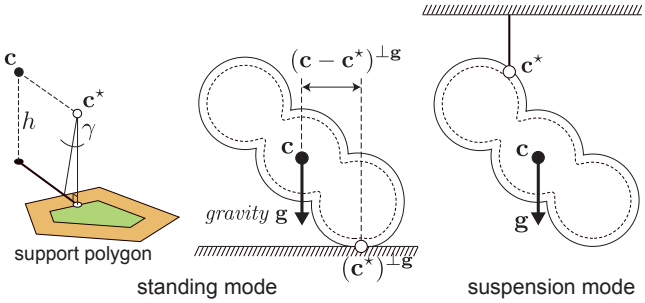


Figure 2: *(Left) The support polygon for standing models (orange) is chosen as the set of the lowest vertices relative to the gravity direction. The horizontal component of the target center of mass* $\mathbf{c}^\star$*, for which we optimize, is the projection of* $\mathbf{c}$ *onto the "safe region" (green) of the support polygon, defined by the toppling angle* γ*. We assume that the height of* $\mathbf{c}^\star$ *is equal to that of* $\mathbf{c}$ *when we define the safe region. (Right) For suspended models the target center of mass is the selected suspension point.*

region" of the base. We ensure the model will not trivially topple when pushed by small external forces by adding a stability offset from the support boundaries, determined by a target toppling angle γ (Figure 2).

In this work, we assume that the support polygon is chosen by the user and remains fixed throughout optimization. Therefore, all surface points touching the ground will remain at their chosen location.

Suspension mode. In this mode the object is meant to be suspended by a string. The user is free to choose the suspension point anywhere on the surface, as well as the desired orientation of the object. To achieve equilibrium the center of mass has to project along gravity onto the suspension point. We therefore select the suspension point as the target center of mass $\mathbf{c}^\star$. The user is responsible for selecting a valid configuration where the cord does not intersect the mesh and the equilibrium, once reached, remains stable.

3.1 Problem formulation

The input to our method is a triangle mesh $\mathcal{M}$ representing the outer surface of the model. Our approach generates two output meshes, $\mathcal{M}_O$ and $\mathcal{M}_I$, defining the outer and inner surfaces of the result: $\mathcal{M}_O$ is a deformed version of the original mesh while $\mathcal{M}_I$ is the inner surface representing the boundaries of interior voids.

Given an unstable model, we look to carve the interior volume for an optimal mass distribution, and simultaneously allow deformation of the original surface. More precisely, given a target center of mass $\mathbf{c}^\star$, we solve the following optimization problem:

$$\underset{\mathcal{M}_I,\mathcal{M}_O}{\operatorname{argmin}} \; (1-\mu)\, E_{CoM}(\mathcal{M}_I,\mathcal{M}_O) + \mu\, E_{\mathcal{M}}(\mathcal{M}_O), \qquad (1)$$

balance — preservation of shape

where $E_{\mathcal{M}}(\mathcal{M}_O)$ is the shape-preserving term, namely, it measures the deviation of the outer shape $\mathcal{M}_O$ from the input shape $\mathcal{M}$ (detailed in Section 5); E_{CoM} measures the projected distance to the target center of mass:

$$E_{CoM}(\mathcal{M}_I,\mathcal{M}_O) = \frac{1}{2}\left\|(\mathbf{c}(\mathcal{M}_I,\mathcal{M}_O) - \mathbf{c}^\star)^{\perp\mathbf{g}}\right\|^2. \qquad (2)$$

Vector $\mathbf{g}$ is the gravity direction ($\|\mathbf{g}\| = 1$) and $^{\perp\mathbf{g}}$ denotes the perpendicular projection onto the support plane along the gravity vector. The weight μ is used to balance between the two objectives; we describe how it is set in Section 5.3.

Center of mass. The center of mass $\mathbf{c}(\mathcal{M}_I,\mathcal{M}_O)$ is fully determined by the geometry of the two surfaces, $\mathcal{M}_I$ and $\mathcal{M}_O$, under the assumption of uniform density. Let Ω be the domain trapped between $\mathcal{M}_I$ and $\mathcal{M}_O$ such that $\partial\Omega = \mathcal{M}_I \cup \mathcal{M}_O$; assume uniform mass density ρ. The mass m and the center of mass $\mathbf{c}$ are defined by volume integrals:

r_i: coordinates in space

$$m(\mathcal{M}_I,\mathcal{M}_O) = \textstyle\int_\Omega \rho\, dr, \;\; \mathbf{c}(\mathcal{M}_I,\mathcal{M}_O) = \frac{1}{m(\mathcal{M}_I,\mathcal{M}_O)} \int_\Omega \rho\mathbf{r}\, dr\,.$$

Using the divergence theorem, these integrals can be expressed as 2D surface integrals over $\partial\Omega$ (see Appendix for details); the center of mass is a rational function of the inner and outer mesh vertices:

$$m = \frac{\rho}{6}\sum_{f\in\mathcal{F}} ((\mathbf{v}_j-\mathbf{v}_i)\times(\mathbf{v}_k-\mathbf{v}_i))\cdot(\mathbf{v}_i+\mathbf{v}_j+\mathbf{v}_k)\,, \qquad (3)$$

$$\mathbf{c} = \frac{\rho}{24\,m}\sum_{f\in\mathcal{F}} ((\mathbf{v}_j-\mathbf{v}_i)\times(\mathbf{v}_k-\mathbf{v}_i)) \star g(\mathbf{v}_i,\mathbf{v}_j,\mathbf{v}_k), \qquad (4)$$

where $\mathcal{F}$ is the combined set of faces of $\mathcal{M}_I$ and $\mathcal{M}_O$ (note that face orientation is important: the normals of $\mathcal{M}_O$ should point outwards and those of $\mathcal{M}_I$ should point inwards), $f = (i,j,k)$ is a triangle, $\mathbf{v}$ are the vertex positions, and $\star$ is the component-wise product. The vector-valued function g is:

$$g(\mathbf{v}_i,\mathbf{v}_j,\mathbf{v}_k) = \mathbf{v}_i\star\mathbf{v}_i+\mathbf{v}_i\star\mathbf{v}_j+\mathbf{v}_j\star\mathbf{v}_j+\mathbf{v}_j\star\mathbf{v}_k+\mathbf{v}_k\star\mathbf{v}_k+\mathbf{v}_k\star\mathbf{v}_i.$$

This formulation allows us to compute the gradient of the center of mass analytically, which will be used in the optimization.

Multiple bases. In the case of multiple bases, the objective becomes a sum of the projected center of mass distances to the targets:

$$(1-\mu)\sum_k \frac{1}{2}\left\|(\mathbf{c}(\mathcal{M}_I,\mathcal{M}_O) - \mathbf{c}_k^\star)^{\perp\mathbf{g}_k}\right\|^2 + \mu\, E_{\mathcal{M}}(\mathcal{M}_O). \qquad (5)$$

Note that for each base k the projection is along the corresponding gravity vector $\mathbf{g}_k$. Any number of bases are possible, but the existence of a solution is conditioned by a non-empty intersection of the extruded support polygons.

Given the above formulation, we introduce a number of approximations to make the problem tractable. Namely, we use a voxelization to represent the inner volume, and introduce handles to control deformation in a well-behaved subspace.

Mass carving. We compute a voxel grid $\mathcal{V}$ as a discretized representation of the volume inside $\mathcal{M}$. Voxels are a natural representation for solid objects, they are less prone to difficulties such as tracking a surface and avoiding self-intersections, which is critical in this case, as we are constrained by the outer hull and in its close proximity. Our approach assigns a binary fill value α_i to each voxel (1 means full and 0 means empty); the whole set of these values is denoted by α. By carving, we obtain a discretized inner surface defined by the boundaries of the filled regions (Figure 4). Section 4 describes the carving phase in detail.

Shape deformation. Instead of using all the degrees of freedom of the mesh (i.e. vertices) we manipulate deformation handles. We expect small changes impacting large portions of the model to be the most useful to optimize for stability, and therefore reduce the unnecessarily large number of DOFs. We use the simple linear blend skinning (LBS) deformation model with bounded biharmonic weights [Jacobson et al. 2011] for parameterizing the deformation of the solid shape, where users place a small number of handles within the shape's volume. The degrees of freedom then become the affine transformations at the handles (which are interpolated to

handles at which deformation occurs

the rest of the shape by LBS); we denote them by $\mathcal{H}$. To measure the quality of the deformation, we use the deformation energy from Laplacian editing [Sorkine et al. 2004] as the shape-preserving surface energy term $E_{\mathcal{M}}$. Section 5 describes the deformation approach in detail.

Our formulation. Given the above representations, our final energy formulation in terms of the voxel variables and deformation handles is:

$$\underset{\alpha,\mathcal{H}}{\operatorname{argmin}}\ (1-\mu)\,\frac{1}{2}\left\|(\mathbf{c}\,(\mathcal{M}_I(\alpha,\mathcal{H}),\mathcal{M}_O(\mathcal{H}))-\mathbf{c}^\star)^{\perp\mathbf{g}}\right\|^2 + \\ +\,\mu\,E_{\mathcal{M}}\,(\mathcal{M}_O(\mathcal{H}))\,. \tag{6}$$

Note that α determines the carving of the inner volume and hence the geometry of the inner surface $\mathcal{M}_I$, and $\mathcal{H}$ determines the deformation of both the outer surface $\mathcal{M}_O$ and the voxel grid (and $\mathcal{M}_I$).

3.2 Stability optimization

Our objective (6) has a complex form: it mixes discrete (binary) variables α for the voxel fills with continuous variables $\mathcal{H}$ for the transformations of the handles. We therefore approach the stability optimization with an iterative process that alternates between inner volume and object geometry optimization. In the first phase, we aim to find the optimal mass distribution inside the object, assuming a fixed surface mesh $\mathcal{M}_O$. In the second phase, we aim to deform the object to improve stability while preserving features. This can be seen as a block coordinate descent method (see e.g. [Ortega and Rheinboldt 1970]): we alternate between fixing $\mathcal{H}$ and optimizing (6) by varying α, and vice versa, aiming to decrease the energy in each step. In this alternating optimization, the choice of empty voxels α is recomputed at every iteration. There is therefore no limitation for carved voxels to remain empty, and vice versa.

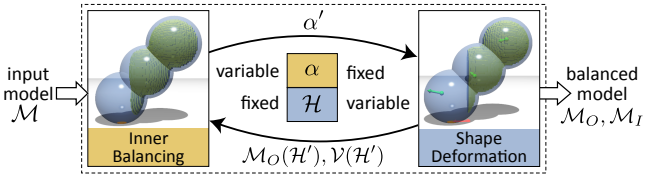


Figure 3: *Our stability optimization alternates between two phases: inner carving and shape deformation. The output is two meshes representing the deformed outer surface $\mathcal{M}_O$ and the inner surface (the boundary of interior voids) $\mathcal{M}_I$, respectively.*

4 Inner carving

During the carving stage we fix deformation parameters $\mathcal{H}$, such that the outer surface geometry $\mathcal{M}_O$ is constant, and we change the balance of the object by carving the inside of the shape, i.e., by varying α. The volume discretization (voxel grid) is computed once at the beginning; later the grid deforms together with the surface during the deformation stage (but the lattice connectivity is kept fixed, i.e., we do not remesh the volume). The resulting distortion is taken into account when computing the center of mass.

Since $\mathcal{H}$ is fixed at this stage, the optimization (6) simplifies to the first term:

$$E_{CoM}(\alpha) = \left\|(\mathbf{c}(\alpha)-\mathbf{c}^\star)^{\perp\mathbf{g}}\right\|^2. \tag{7}$$

Here we simplified the notation: $\mathbf{c}(\alpha) = \mathbf{c}(\mathcal{M}_I(\alpha), \mathcal{M}_O)$.

First, we set α_0 such that all the voxels are filled. Throughout the carving process we never empty the voxels located within a distance $t_{\min}$ from $\mathcal{M}_O$ to ensure a minimal thickness to the outer *hull*. $t_{\min}$ is set by the user according to the physical material properties.

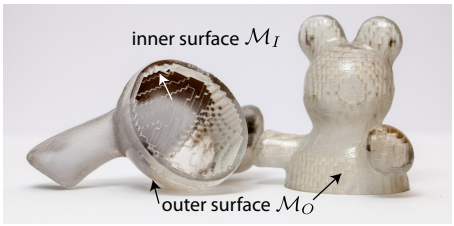


Figure 4: *Cross-section displaying inner and outer surfaces. The outer boundary is the smooth surface mesh $\mathcal{M}_O$. The inner boundary (neighboring empty voxels) is the inner surface $\mathcal{M}_I$.*

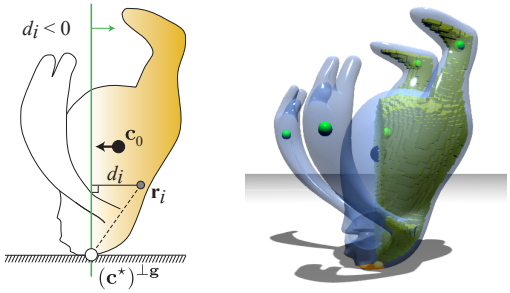


Figure 5: *Optimized voxel fill for a given surface mesh. (Left) The original center of mass (black) is moved closer to the target by carving the inside. Removing voxels to the* left $(d_i < 0)$ *of the plane cannot further improve stability, while removing voxels on the* right $(d_i > 0)$ *potentially improves E_{CoM}. (Right) A model carved by our algorithm. The yellow volume inside the model is empty.*

We use the following fast heuristic to approximate the optimal voxel carving. Given the current center of mass $\mathbf{c}_0 := \mathbf{c}(\alpha_0)$, imagine cutting a plane through $(\mathbf{c}^\star)^{\perp\mathbf{g}}$ that is perpendicular to $(\mathbf{c}_0-\mathbf{c}^\star)^{\perp\mathbf{g}}$ (Figure 5). The filled voxels located in the half-space plane that does *not* contain $\mathbf{c}_0$ are left unchanged: carving these would only result in $\mathbf{c}_0$ moving further away from the target. However, carving the voxels located in the other half-space potentially brings $\mathbf{c}_0$ closer to $\mathbf{c}^*$ and may decrease E_{CoM}.

Hence, we sort the voxels in decreasing order of their *signed* distance to the cutting plane:

$$d_i = (\mathbf{r}_i - \mathbf{c}^\star)\cdot(\mathbf{c}_0-\mathbf{c}^\star)^{\perp\mathbf{g}}, \tag{8}$$

where $\mathbf{r}_i$ is the centroid of voxel i. Voxels with negative distance are ignored. We carve the voxels with positive d_i one by one, and keep track of the value of the energy E_{CoM}. When all the voxels in the queue are processed, we pick the carve pattern α that corresponded to the lowest energy value. Note that we do not stop as soon as the energy is not decreasing anymore: Removals of voxels located on the positive side of the plane but across the center of mass may compensate each other, while removing each voxel independently may temporarily increase E_{CoM}. Also note that carving voxels in this order ensures that we do not create "floating" components.

Our heuristic fixes the center of mass for the definition of the cutting plane. While a better solution may be found by removing single voxels and recomputing the center of mass, the plane constraint greatly benefits performance. Note that we always evaluate and record the true energy E_{CoM} when removing a voxel; computing

the center of mass $\mathbf{c}(\alpha)$ can be efficiently done thanks to the incremental nature of the process. As evident from Equation 4, when removing the new voxel we just need to update the previous value by subtracting the integrals of those $\mathcal{M}_I$'s faces that disappear and add the integrals of the new faces; this is conveniently achieved by subtracting the integrals over *all* the (oriented) faces of the carved-out voxel.

The balancing process above can be iterated by recomputing the new center of mass and the cutting plane, and sorting once more the voxels that remained filled. At each iteration we either decrease the energy or detect that too few voxels were removed and stop. Figure 5 shows a balancing result.

Multiple bases. If several targets are specified (Equation 5), we define a plane for each one, and hence have multiple distances d_i; we use their sum (only the positive d_is participate) as the sorting value and proceed as described above.

At the end of the carving stage, we have the updated inner surface $\mathcal{M}_I$ as a mesh bordering between the empty and the filled voxels. The vertices of this mesh are vertices of the voxel grid and they will be deformed by transforming the handles in the deformation stage.

5 Shape deformation

If carving is sufficient for feasibility, our algorithm will stop after the first iteration without any deformation to the outer surface. However, the most interesting and delicate models require deformation to achieve balance.

During this stage, α and the connectivities of the $\mathcal{M}_I, \mathcal{M}_O$ meshes are fixed. We deform both the outer surface $\mathcal{M}_O$ and the inner volume, outlined by $\mathcal{M}_I$, to improve the balance and further decrease the energy in (6). We use the linear blend skinning (LBS) deformation model, whose parameters are affine transformations $\mathcal{H}$ at a number of manipulation handles, placed by the user in the beginning of the session. Specifically, if $\mathbf{v}$ is a vertex of $\mathcal{M}_O$ or $\mathcal{M}_I$ or the inner volume grid $\mathcal{V}$, its deformed position is defined by

$$\mathbf{v} = \sum_{j=1}^{N} w_j(\tilde{\mathbf{v}})\, \mathcal{H}_j(\tilde{\mathbf{v}}), \tag{9}$$

where $\tilde{\mathbf{v}}$ is the initial, rest pose position of $\mathbf{v}$. We assume there are N handles, the weight function of handle j is $w_j : \mathcal{V} \to [0,1]$, and $\mathcal{H}_j$ is the affine transformation assigned to handle j. The weight functions are precomputed in on the initial voxel grid $\mathcal{V}$; the initial surface $\mathcal{M}$ is contained in $\mathcal{V}$, so that the values of the w_js are interpolated onto $\mathcal{M}$'s vertices from the surrounding voxel vertices. We use bounded biharmonic weights [Jacobson et al. 2011], which are shown to be smooth and produce intuitive deformations. The weights sum up to 1 at each vertex and they interpolate the handles, namely, $w_j = 1$ at handle j and all other weights w_k $(k \neq j)$ are equal 0 there.

Overall, parameterizing the deformation as in Equation 9 serves two purposes: it confines our optimization to a well-behaved shape subspace, and it drastically reduces the number of degrees of freedom. Note that the LBS formula is very fast and embarrassingly parallel to compute.

Handle placement. The number of handles N needed is typically small (between 2 and 6) and easy to select through standard UI tools. For each handle, the user selects a screen-space point in the mesh interior, and the depth position is set to the center of the model's volume there (Figure 6). Our optimization is not sensitive to noise in the initial handle positions since the overall deformation is low frequency (see Figure 10).

It is important that the chosen support remains fixed. In addition to the user-selected deformation handles, we define all the voxels in the base (or multiple bases) of support as an additional handle and keep the associated affine transformation $\mathcal{H}_{base}$ fixed.

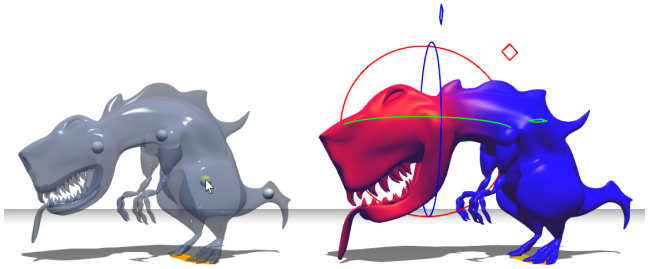


Figure 6: *Handle placement. (Left) Locations of user-selected handles are indicated by spheres. Only a small number of handles are required. Handles are automatically placed in the center of the volume. (Right) Mesh deformation by manipulating a handle; the weight function of the handle is visualized. Handles may translate, scale and rotate. Here, the user scaled up the head of the T-Rex*

We restrict the affine transformations $\mathcal{H}_j$ to translation, uniform scaling and rotation, thereby eliminating shears which tend to distort the shape. Furthermore, rotations can be interactively set by the user, but we do not optimize for them due to the nonlinear parameterization of rotations in 3D. We optimize for the translation and scaling parameters, so that in (9), $\mathbf{v}$ is a linear function of $\mathcal{H}$. Thus for each handle j, $\mathcal{H}$ has four scalar parameters (one scalar for uniform scale and three for translation); overall $\mathcal{H}$ is a vector of size $4N$. Denoting the column vector with all $\mathcal{M}_O$'s vertex coordinates stacked together by V_O, and likewise V_I for $\mathcal{M}_I$'s vertex coordinates, we can rewrite the LBS formula (9) in matrix form as

$$V_O = M_O \mathcal{H}, \;\; V_I = M_I \mathcal{H}, \tag{10}$$

where the matrices M_O, M_I are *fixed* and comprised of some combinations of the initial positions $\tilde{\mathbf{v}}$ and the skinning weights w_j. For additional details on how to construct these matrices, see e.g. [Jacobson et al. 2012].

5.1 Deformation energy

Equipped with our deformation model, we now come to minimize our energy (6) by varying $\mathcal{H}$. To simplify notation when α is fixed, we have

$$\min_{\mathcal{H}} \; (1-\mu)\, E_{CoM}(\mathcal{M}_I(\mathcal{H}), \mathcal{M}_O(\mathcal{H})) + \mu\, E_{\mathcal{M}}(\mathcal{M}_O(\mathcal{H})) \tag{11}$$

The second term $E_{\mathcal{M}}$ allows us to control the shape preservation with respect to the initial model $\mathcal{M}$. We use the Laplacian Editing energy [Sorkine et al. 2004], which quantifies the deformation of the outer surface using differential coordinates and implicit rotations. The energy is quadratic in the mesh vertex positions and can be written as

$$E_{\mathcal{M}}(\mathcal{M}_O) = \frac{\lambda}{2}\, V_O^T M_{\text{Lap}} V_O, \tag{12}$$

where the scale factor λ compensates for the difference in dimensionality between E_{CoM} and $E_{\mathcal{M}}$, and M_{Lap} is a sparse symmetric positive semidefinite matrix whose entries only depend on the initial mesh $\mathcal{M}$. The purpose of λ is to normalize the weight μ across meshes, but this is not critical since μ adapts during optimization (see Section 5.3). We currently set λ manually, once per mesh. For details on how to construct M_{Lap} please refer to [Sorkine et al.

2004]. An important fact is that $E_{\mathcal{M}}$ is invariant to translation and global uniform scaling of the geometry V_O; it is also invariant to moderate rotation, within the bounds possible for linear variational deformation [Botsch and Sorkine 2008]. Hence this quadratic energy provides a reasonable tradeoff between efficient minimization and shape preservation properties.

By combining (10) and (12), we get the deformation energy term

$$E_{\mathcal{M}}(\mathcal{M}_O(\mathcal{H})) = \frac{\lambda}{2}\, \mathcal{H}^T \left(M_O^T M_{\text{Lap}} M_O \right) \mathcal{H}. \qquad (13)$$

5.2 Gradient descent

The energy in (11) is a rational expression in $\mathcal{H}$ due to the E_{CoM} term, hence its optimization is nonlinear. The subspace reduction in (9) leads to a small number of unknowns $4N$, with a small dense matrix in (13) as a result. We thus employ gradient descent with analytically computed gradients. While a more sophisticated optimization method (e.g. Newton) is possible, it may hinder interactivity; with gradient descent, we perform few iterations while the user is interactively manipulating the mesh, to maintain a reasonable framerate, and can iterate further towards convergence once the user lets go of the UI.

We can compute the gradient of the energy w.r.t. the handle transformation parameters $\mathcal{H}$ using simple derivation rules:

$$\frac{\partial E}{\partial \mathcal{H}} = (1-\mu)\, \frac{\partial E_{CoM}}{\partial \mathcal{H}} + \mu\, \frac{\partial E_{\mathcal{M}}}{\partial \mathcal{H}}. \qquad (14)$$

The center of mass is a function of the inner and outer surface vertex positions: $\mathbf{c} = \mathbf{c}(V_I(\mathcal{H}), V_O(\mathcal{H}))$; we omit the argument for the sake of brevity and just write $\mathbf{c}$. Then the partial derivatives for each energy term are given by:

$$\begin{aligned}
\frac{\partial E_{CoM}}{\partial \mathcal{H}} &\overset{(2)}{=} \left((\mathbf{c}-\mathbf{c}^\star)^{\perp \mathbf{g}} \right)^T \frac{\partial \mathbf{c}}{\partial \mathcal{H}} \\
&= \left((\mathbf{c}-\mathbf{c}^\star)^{\perp \mathbf{g}} \right)^T \left(\frac{\partial \mathbf{c}}{\partial V_I} \frac{\partial V_I}{\partial \mathcal{H}} + \frac{\partial \mathbf{c}}{\partial V_O} \frac{\partial V_O}{\partial \mathcal{H}} \right) \\
\frac{\partial V_I}{\partial \mathcal{H}} &\overset{(10)}{=} M_I \\
\frac{\partial V_O}{\partial \mathcal{H}} &\overset{(10)}{=} M_O \\
\frac{\partial E_{\mathcal{M}}}{\partial \mathcal{H}} &\overset{(13)}{=} \lambda\, \mathcal{H}^T \left(M_O^T M_{\text{Lap}} M_O \right)
\end{aligned}$$

Please refer to the Appendix for details on computing the center of mass gradients $\partial \mathbf{c}/\partial V_I$, $\partial \mathbf{c}/\partial V_O$. Note that if several targets are specified (as in Equation 5), this only affects the E_{CoM} term, and we simply sum up the gradients of each gravity direction. The step size in the gradient descent is dynamically set so as to not increase the energy. This is further described Section 5.3.

Once the gradient descent converges, we obtain the new outer and inner surfaces, and we also deform the voxel grid using the optimized parameters $\mathcal{H}$, according to the LBS formula (9). This gives us the modified voxel positions, with which we can restart the carving process (Section 4).

5.3 Convergence and balance improvement

We have experimented with varying the target $(\mathbf{c}^\star)^{\perp \mathbf{g}}$ and the weighting parameter μ (Equation 11) in order to help the method find a better balanced solution. In particular, the target can be recomputed after each carving and deformation iteration as the projection of the *current* (rather than the starting) center of mass onto the support polygon; this helps the method to find a balanced position quicker, since the distance between $\mathbf{c}^{\perp \mathbf{g}}$ and the support polygon generally decreases.

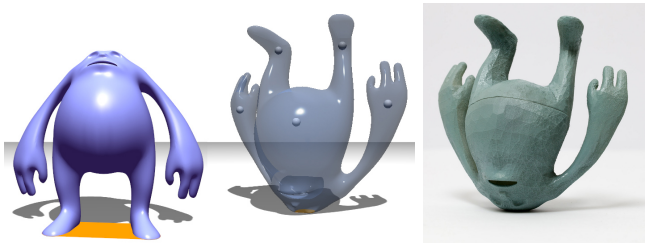


Figure 7: *(Left) Original model. (Middle) The user wishes to balance the model in an upside-down pose. She selects a number of handles for the optimizer. (Right) The carved, deformed model is balanced and stands on its head after printing.*

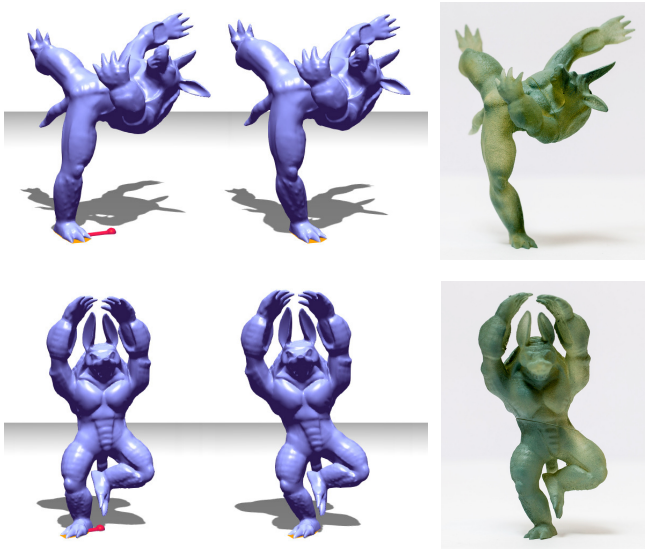


Figure 8: *Armadillo model with different starting poses. (Left) Unstable input. (Middle) Balanced result. (Right) Printed model.*

We heuristically adapt the step size and μ during optimization. The step size is initialized with a default value (1.0). At each step of the gradient descent we check whether the energy decreased by at least 1%. If not, we step back and decrease the step size by a fixed ratio (0.8). If the step size reaches a pre-defined minimum (0.4) before stability is achieved, we consider that the shape preservation term prevents further optimization and decrease μ by a fixed amount (0.05). The step size is reset to its initial value. If μ reaches zero before stability is achieved, we consider that the configuration cannot be further improved and stop.

Throughout this process, advanced users may at any time take control over μ: Its influence is rather intuitive and allows a better balance at the cost of more deformation.

6 Results

Our approach offers precise balance of models after 3D printing. The user may specify balance by the model orientation, and in the case of suspended models, also by the attachment point. We applied our algorithm to a variety of objects including character models, sculptures and toys, and fabricated a number of the final results.

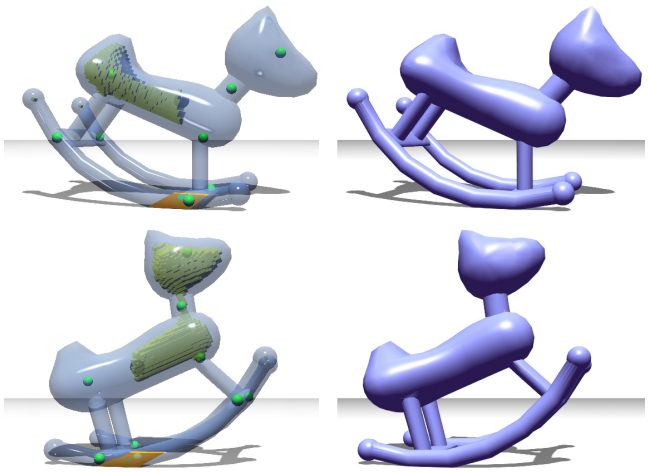


Figure 9: *Curved base. While many stable orientations are possible for the rocking horse, we can choose a specific orientation as the stable resting pose of the model. Here we achieve two completely different orientations by carving different parts of the model.*

Incorporating the exact center of mass in our formulation is particularly important for precarious examples, such as the horse and T-Rex in Figure 1. The weight of the body must be balanced on a proportionally tiny base of support, where there is little tolerance for error. Figure 7 also shows the combined use of carving and deformation to achieve an unexpected headstand pose. In Figure 8 the balanced result alters the angle of the supporting leg. Despite an overall shift in the position of the upper body, the surface features remain visually unchanged from the original. Figure 8 is a case where deformation is particularly necessary: The volume of the hull is comparable to the inner volume since only voxels in the chest could be removed without surpassing thickness limits. Carving alone could not achieve balance of the final model. In contrast, in Figure 9 the optimizer mostly relies on carving to modify the rest pose of a children's rocking horse.

Interaction is an important part of discovering designs with unusual balance. Our chosen deformation scheme allows the user to easily change the configuration of a model, so that the algorithm cooperates with the user to find the best compromise. An example is the T-Rex (Figure 1) where the user scaled up the head. The corresponding handle is locked and the optimizer exploits the other degrees of freedom to recover balance. Further examples are shown in the accompanying video. It should be noted that while the user can easily re-position handles for exploring new designs, the handle positions do not need to be very precisely selected for the optimization (see Figure 10).

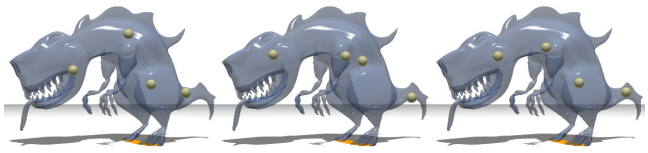


Figure 10: *Handles do not have to be precisely positioned. The left image reveals the handles chosen by the user. We produce two other sets of handles by adding Gaussian noise to their initial positions in all directions. We optimize all three for stability (deformation+carving) and, for each mesh vertex, we compute the average distance between its positions in the three results. On average we displace the handles by* 6.7% *of the longest mesh dimension. The average distance between vertices in the results is* 0.33%*, an order of magnitude less.*

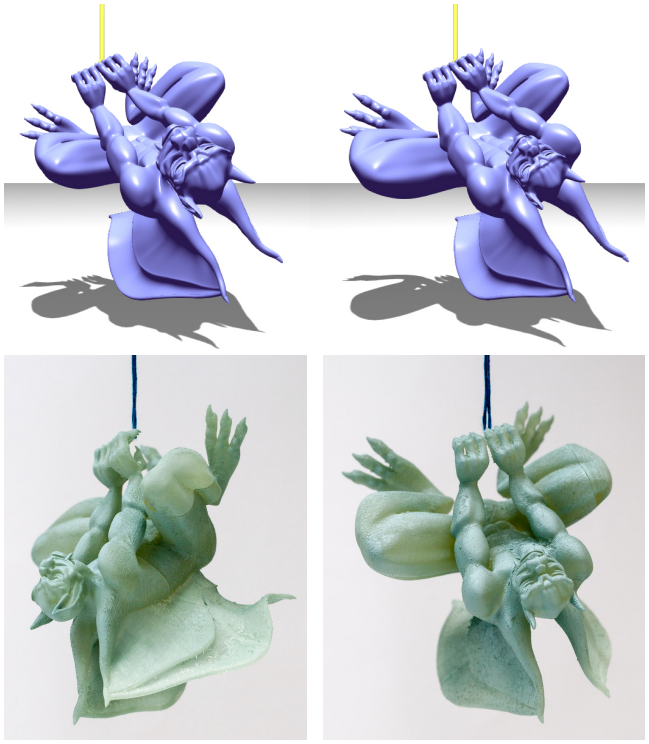


Figure 11: *Gargoyle hanging from one of its claws. (Top-Left) Input that would rotate away from the intended orientation. (Top-Right) Optimized result: the new center of mass aligns with the suspension point along the gravity direction. (Bottom) The printed, optimized model.*

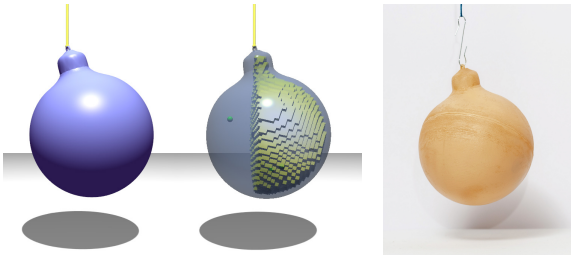


Figure 12: *Christmas tree ornament hanging asymmetrically as a result of inner voids.*

In addition to standing models, our approach applies to suspended designs. The user may specify the angle at which the model hangs as well as the point of attachment. Figures 11 and 12 depict models hanging at unintuitive angles through a combination of inner carving and deformation.

Our approach generalizes to multiple bases as previously discussed. Figure 13 shows a result for a teddy bear model that balances in two configurations.

3D printing. We fabricated prototypes using a high-resolution Objet 3D printer [http://objet.com/]. Models were printed with rigid Vero^{TM} material. In order to create hollow models, it was necessary to create cuts in the geometry and print in multiple assemblable

Figure 13: *(Top) This Teddy bear has been edited and optimized to stand in two different positions. (Bottom) The printed object stands in both configurations. We intentionally printed in transparent material to reveal the uneven distribution of weight inside.*

pieces. This provides access to the interior so that support material may be removed. Note that support material is necessary only in the printing bed, not in finished form. Since only a single cut was required for all the models, we divided them into two halves manually using standard software tools. The cut locations have been chosen to facilitate clean-up of the empty parts after printing. The cuts could be better hidden, for instance following insights from mesh parameterization [Sheffer and Hart 2002]. Other printing technologies could make this step unnecessary, for example, with Z-Corp powder-based printers cuts are unnecessary; instead, a hole is required for the powder to exit the enclosed voids.

3D printers cannot print walls thinner than a given threshold. We set the minimal hull thickness $t_{\min}$ above this value, with a security margin. Our algorithm does not guarantee that deforming voxels will preserve $t_{\min}$, however it was the case on all our printed models. If needed, the carving algorithm can be modified to force all deformed voxels within a $t_{\min}$ distance of the hull to be included.

Validation. Figure 14 shows the effect of optimizing the same model while selectively disabling terms in our problem formulation (6) and the optimization process. Disabling carving (c) introduces significant scaling to achieve the necessary weight reduction of the top spheres. Disabling the shape-preserving Laplacian energy (d) introduces significant distortion in the bottom sphere where translation can now freely occur. The difference between our result (b) and (d) shows the importance of scaling: It affords for significant changes in weight distribution while preserving local details. Disabling both carving and Laplacian (e) leads to an unacceptable distortion of the initial model.

Model	# tri.	# vox.	time/iter.	# iter.	stands
Rocking toy	5k	28k	35 ms	43	0
3-spheres	12k	83k	88 ms	128	80
Armadillo (top)	86k	22k	87 ms	277	120
T-Rex	95k	38k	84 ms	148	94
Horse	172k	27k	114 ms	206	51

Table 1: *Optimization performance. For each model we specify both the number of iterations to convergence (into the safe region), and the iteration at which the model stands (inside the support polygon). Note that reported times are for the optimization only, we do not include rendering time.*

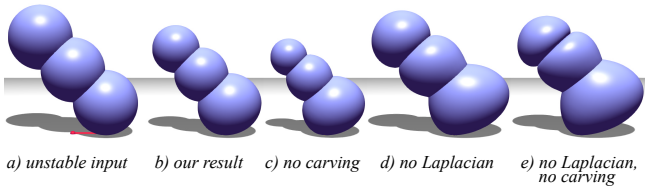


Figure 14: *Results obtained when selectively disabling the Laplacian deformation energy term $E_{\mathcal{M}}$ and the carving step in our formulation. Please refer to the text for details.*

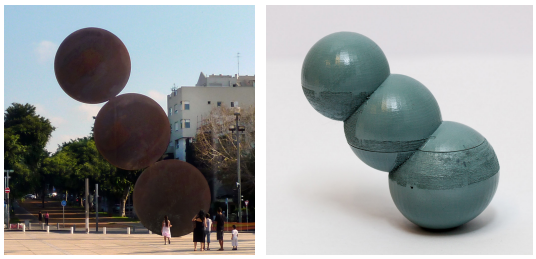


Figure 15: *(Left) Sculpture where significant support structure was required to make it stand. (Right) Our 3-sphere model mimics the unintuitive balance and stands without hidden supports.*

We recorded the performance for a number of models including time per iteration and total iterations to convergence. Results are reported in Table 1.

Limitations and future work. A poor choice of handles will likely result in aesthetically unpleasing solutions. Such bad choices would be handles all close to the base, or too few handles, not allowing for enough degrees of freedom (see Figure 16, *right*). It is possible in extreme cases for user deformations to lead to collisions between different model parts, as shown Figure 16, *left*. An area of future work could involve prevention of self-intersections for both the inner and outer surface meshes. Although we do not explicitly prevent them, in practice we did not observe this to happen even for large deformations. An extension could be to combine our deformation step with interference-aware modeling [Harmon et al. 2011].

Our focus has been on printing single material models. It would be possible to generalize our method to several materials. Varying den-

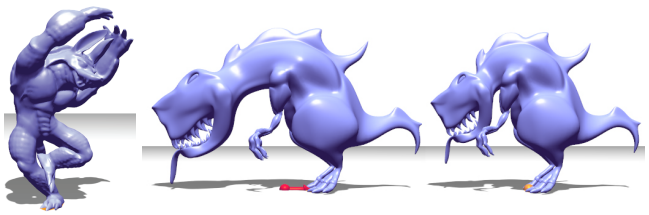


Figure 16: *Limitations. (Left) Depending on the starting pose, large deformations may be necessary, which could result in self-intersections, e.g. the Armadillo's foot intersects his leg. (Right) When the degrees of freedom are too limited, unpleasing deformations may result. Here a single handle was placed in the head, causing the deformation to concentrate in the neck region.*

Figure 17: *Printed models standing together.*

sities in the mass distribution would further reduce the deformation of the surfaces. Due to limitations in current printing technologies we keep this for future work.

7 Conclusion

We have demonstrated how to ensure stability of printed models so that they may be physically realized. Our approach combines stability objectives with shape preservation measures in order to respect the user's original design, whilst making it feasible without the need for aids such as an oversized base or the addition of stilts. We introduced a novel algorithm that leverages the use of interior voids to manipulate mass distribution without affecting the exterior appearance of the model. We alternate between interior carving and shape deformation to arrive at a final stable result. We have demonstrated effectiveness with numerous results including character models and everyday objects, many requiring particularly delicate balance conditions to stand in equilibrium. Further, we extended our technique to objects that stand on multiple bases.

Acknowledgments

We thank Daniele Panozzo and Philippe Block for insightful discussions; Ladislav Kavan for providing source code for bounded biharmonic weights; Amit Bermano, Bernd Bickel and Markus Buehler for their help with producing the 3D models; and Gioacchino Noris, Ronnie Gänsli and Steven Poulakos for their help in painting and photographing the printed 3D models. We thank Alec Jacobson and Ladislav Kavan for the Dancing Armadillo models. Other models are from `www.turbosquid.com`: T-Rex (by *csirkeFrs*), Gargoyle (by *csirkeFrs*), Mr Humpty (by *ArtbySmitty*); and `http://archive3d.net/`: Horse (by *Gian Lorenzo*) and Christmas ornament (by *Labrouste Henri*). The Teddy Bear model is provided courtesy of the AIM@SHAPE Shape Repository. This work was supported in part by an SNF award 200021_137879, ERC grant ShapeForge (StG-2012-307877), ERC grant iModel (StG-2012-306877) and a gift from Adobe Research. Emily Whiting is supported by the ETH Zurich Postdoctoral Fellowship.

References

ALLAIRE, G. 2001. *Shape optimization by the homogenization method*, vol. 146 of *Applied Mathematical Sciences*. Springer.

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph. 31*, 4.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Trans. Vis. Comput. Graph. 14*, 1, 213–230.

CALÌ, J., CALIAN, D. A., AMATI, C., KLEINBERGER, R., STEED, A., KAUTZ, J., AND WEYRICH, T. 2012. 3D-printing of non-assembly, articulated models. *ACM Trans. Graph. 31*, 6, 130:1–130:8.

DEROUET-JOURDAN, A., BERTAILS-DESCOUBES, F., AND THOLLOT, J. 2010. Stable inverse dynamic curves. *ACM Trans. Graph.*, 137:1–137:10.

HARMON, D., PANOZZO, D., SORKINE, O., AND ZORIN, D. 2011. Interference aware geometric modeling. *ACM Trans. Graph. 30*, 6, 137:1–137:10.

JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4, 78:1–78:8.

JACOBSON, A., BARAN, I., KAVAN, L., POPOVIĆ, J., AND SORKINE, O. 2012. Fast automatic skinning transformations. *ACM Trans. Graph. 30*, 4, 77:1–77:10.

KEDZIORA, J., 2011. Balancing sculptures. Display at the Kinetica Art Fair.

LAUER, D. A., AND PENTAK, S. 2011. *Design Basics*. Wadsworth Publishing.

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

ORTEGA, J., AND RHEINBOLDT, W. 1970. *Iterative solution of nonlinear equations in several variables*. Academic Press.

RIVERS, A., ADAMS, A., AND DURAND, F. 2012. Sculpting by numbers. *ACM Trans. Graph. 31*, 6, 157:1–157:7.

RIVERS, A., MOYER, I. E., AND DURAND, F. 2012. Position-correcting tools for 2D digital fabrication. *ACM Trans. Graph. 31*, 4, 88:1–88:7.

SAUL, G., LAU, M., MITANI, J., AND IGARASHI, T. 2011. Sketchchair: an all-in-one chair design system for end users. In *Proc. 5th Intl. Conf. Tangible, Embedded, and Embodied Interaction*, 73–80.

SCHNEIDER, P. J., AND EBERLY, D. H. 2002. *Geometric Tools for Computer Graphics*. Morgan Kaufmann Publishers.

SHEFFER, A., AND HART, J. C. 2002. Seamster: inconspicuous low-distortion texture seam layout. In *Proc. Visualization*, IEEE Computer Society, 291–298.

SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., AND GUO, B. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. 26*, 3.

SMITHSON, R., 1968. Gyrostasis. Permanent collection of the Hirshhorn Museum.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proc. Symposium on Geometry Processing*, 179–188.

STAVA, O., VANEK, J., BENES, B., CARR, N. A., AND MECH, R. 2012. Stress relief: improving structural strength of 3D printable objects. *ACM Trans. Graph. 31*, 4, 48.

TELEA, A., AND JALBA, A. 2011. Voxel-based assessment of printability of 3D shapes. In *Proc. ISMM*, 393–404.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4.

VON MEISS, P. 1990. *Elements of architecture: from form to place*. Routledge.

VOUGA, E., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2012. Design of self-supporting surfaces. *ACM Trans. Graph. 31*, 4, 87:1–87:11.

WHITING, E., OCHSENDORF, J., AND DURAND, F. 2009. Procedural modeling of structurally-sound masonry buildings. *ACM Trans. Graph. 28*, 5, 112:1–112:9.

WHITING, E., SHIN, H., WANG, R., OCHSENDORF, J., AND DURAND, F. 2012. Structural optimization of 3D masonry buildings. *ACM Trans. Graph. 31*, 6, 159:1–159:11.

ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. 2012. Motion-guided mechanical toy modeling. *ACM Trans. Graph. 31*, 6, 127:1–127:10.

Appendix

Mass and center of mass. The intermediate step for computing the exact center of mass ((3), (4)) is the transformation of 3D volume integrals into 2D surface integrals over the boundary thanks to the divergence theorem. In our case, the integration domain is the volume between the inner surface $\mathcal{M}_I$ and the mesh surface $\mathcal{M}_O$, such that we end up with integrals over those two surfaces. After discretization over the mesh triangles [Schneider and Eberly 2002]:

$$m = \frac{\rho}{3} \sum_{\mathcal{F}} \mathbf{n}_{\mathcal{F}} \cdot \iint_{\mathcal{F}} [x\ y\ z]^T \, dS$$

$$\mathbf{c} = \frac{\rho}{2m} \sum_{\mathcal{F}} \mathbf{n}_{\mathcal{F}} \star \iint_{\mathcal{F}} \left[x^2\ y^2\ z^2\right]^T \, dS$$

where $\star$ is the component-wise product of two vectors, $\mathbf{n}_{\mathcal{F}}$ is the normal of the triangle and we sum over all the triangles. dS is the area element on a triangle.

Mass and center of mass gradients. Following from the final formulations in (3) and (4), the gradient of mass w.r.t. vertex position is obtained by summing up the contributions of the derivatives of all its incident faces:

$$\frac{\partial m}{\partial \mathbf{v}_p} = \frac{\rho}{6} \sum_{\mathcal{N}_p} \frac{\partial \left([(\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i)] \cdot [\mathbf{v}_i + \mathbf{v}_j + \mathbf{v}_k] \right)}{\partial \mathbf{v}_p}$$

where $\mathcal{N}_p$ are the faces incident on $\mathbf{v}_p$, with vertex indices i, j, k. The gradient of the center of mass w.r.t. vertex position is:

$$\frac{\partial \mathbf{c}}{\partial \mathbf{v}_p} = \frac{1}{m} \left(\frac{\partial (m\mathbf{c})}{\partial \mathbf{v}_p} - \frac{\partial m}{\partial \mathbf{v}_p} \mathbf{c} \right)$$

$$\frac{\partial (m\mathbf{c})}{\partial \mathbf{v}_p} = \frac{\rho}{24} \sum_{\mathcal{N}_p} \frac{\partial \left([(\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i)] \star [\, g(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k) \,] \right)}{\partial \mathbf{v}_p}$$

We use the following relations:

$$\frac{\partial [\mathbf{v}_i + \mathbf{v}_j + \mathbf{v}_k]}{\partial \mathbf{v}_p} = \mathbf{I}, \quad \frac{\partial [\, g(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k) \,]}{\partial \mathbf{v}_p} = \mathrm{diag}\,(\mathbf{v}_i + \mathbf{v}_j + \mathbf{v}_k)$$

$\partial m / \partial V_O$, $\partial m / \partial V_I$ and $\partial \mathbf{c} / \partial V_O$, $\partial \mathbf{c} / \partial V_I$ are obtained by stacking the above derivatives w.r.t vertices together in a matrix form.