Reduced-Order Shape Optimization Using Offset Surfaces

Przemyslaw Musialski* Thomas Auzinger* Michael Birsak* Michael Wimmer* Leif Kobbelt†

*Vienna University of Technology †RWTH Aachen University

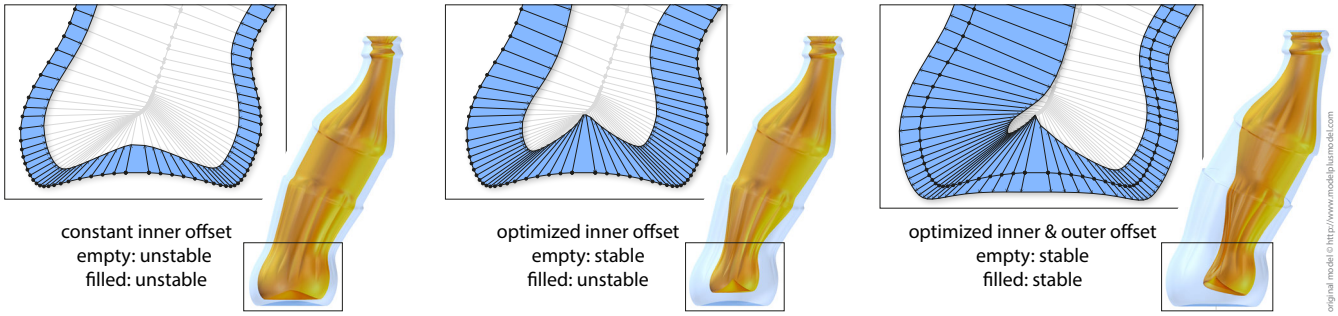

Figure 1: *We introduce a method for reduced-order shape optimization of 2-manifolds that uses offset surfaces to deform the shape. Left: a bottle model is generated using offset surfaces with constant offsets. The resulting object is unable to stand. Center: the offsets are optimized such that the bottle can stand if empty, however, if filled it is unstable. Right: the model is optimized to stand both empty and filled. In order to account for that, offset surfaces are added inside and outside of the original shape.*

Abstract

Given the 2-manifold surface of a 3d object, we propose a novel method for the computation of an offset surface with varying thickness such that the solid volume between the surface and its offset satisfies a set of prescribed constraints and at the same time minimizes a given objective functional. Since the constraints as well as the objective functional can easily be adjusted to specific application requirements, our method provides a flexible and powerful tool for shape optimization. We use manifold harmonics to derive a reduced-order formulation of the optimization problem, which guarantees a smooth offset surface and speeds up the computation independently from the input mesh resolution without affecting the quality of the result. The constrained optimization problem can be solved in a numerically robust manner with commodity solvers. Furthermore, the method allows simultaneously optimizing an inner and an outer offset in order to increase the degrees of freedom. We demonstrate our method in a number of examples where we control the physical mass properties of rigid objects for the purpose of 3d printing.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: geometry processing, geometric design optimization, shape optimization, reduced-order models, physical mass properties, digital fabrication

1 Introduction

Today's geometric modeling software (e.g., Blender) allows for interactive design or customization of 3d geometric shapes, and many of them can now be fabricated at home using a low-cost 3d-printer. However, most such items are created in an ad-hoc fashion, i.e., their geometric and physical aspects are usually assumed intuitively or determined empirically with a series of trial-and-error iterations. While this might work well in some cases, it can also turn into a tedious and especially a costly procedure. For this reason, in the approaching age of personal digital fabrication, there is a growing demand for computational tools that not only enable ordinary users to design and 3d-print their everyday objects, but also allow them to optimize their designs for practical usability.

In this paper, we provide a novel method for shape optimization of geometric objects defined by 2-manifold surface meshes. The particular problem we are dealing with is to find a new shape that is as similar to an input shape as possible, but which at the same time satisfies various global goals. An example of such a goal is depicted in Figure 1: the bottle is intended to stand upright in a desired position when filled, however, it would fall over given its current shape. In order to prevent this, our method automatically adjusts the shape of the object, but simultaneously, it tries to preserve its volume, smoothness, and the overall detailed appearance.

Technically, we formulate this task as a continuous constrained shape optimization problem, which balances shape preservation against given design goals. We express the shape using *offset surfaces*—a simple yet powerful technique where the surface is parameterized by a vector of offset values applied to each vertex in a certain direction. This parameterization allows deforming the surface both locally and globally, depending on the chosen displacements. We describe the details in Section 4.

Since we want to preserve the characteristics of the input shape under deformations as well as possible, we favor displacements of the interior surface of the object if feasible, and penalize displacements of the outer surface explicitly. Additionally, and most importantly, we apply only low-frequency changes to the shape, which is perceptually less noticeable than high-frequency modifications, but still has a large influence on global properties, such as the object's volume and thus mass.

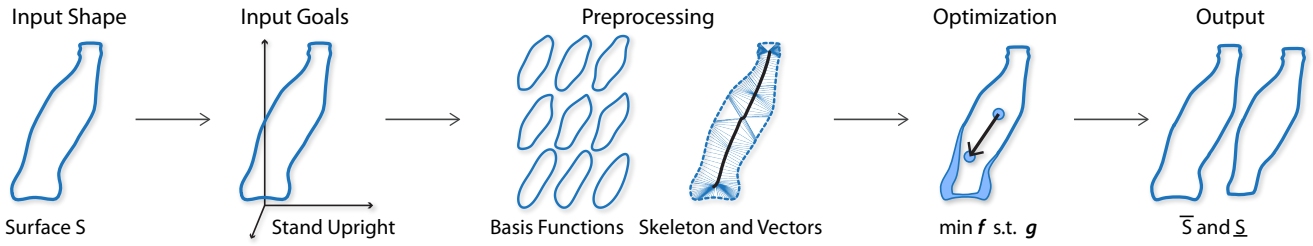

Figure 2: *Overview of our processing pipeline. The input is a shape $\mathcal{S}$ and a desired goal, e.g., stable standing in a given pose. In the preprocessing stage, a set of manifold harmonic basis functions Γ_k, a contracted skeleton mesh $\underline{\mathcal{S}}$, and a vector field $\mathbf{V}$ are computed. In the next stage, the surfaces are optimized with respect to the given goals $\mathfrak{f}$ and constraints $\mathfrak{g}_i$, and finally, an output shape $\overline{\underline{\mathcal{S}}}$ is generated.*

In order to account for these requirements, we decompose the geometry of the input shape into its particular spectral bands using the *manifold harmonic basis* [Vallet and Lévy 2008]. In our context, this representation turns out to be remarkably versatile: by projecting the surface offsets into a subspace of this basis, we obtain an efficient representation of low-frequency modifications that, at the same time, leave high-frequency details largely intact. Moreover, this approach also serves as an intrinsic regularizer that greatly lowers the number of degrees of freedom, turning out to be a powerful order-reduction technique of the otherwise highly underdetermined optimization problem. We provide the details in Section 5.

Our proposed solution is fast, lightweight, numerically stable, and easy to integrate into common 3d modeling software, making it well suitable for practical optimization of global shape properties: for instance, enforcement of mass moments [Bächer et al. 2014] or structural optimization [Lu et al. 2014]. While the latter usually also involves the finite-element discretization of the shape and needs to be evaluated numerically, the former can be expressed as integrals over the object's surface. Since these can be elegantly computed analytically, we utilize them to demonstrate the applicability of our proposed reduced-order shape optimization approach in Section 6.

In summary, our contributions are the following:

- We provide a flexible and robust novel framework for the continuous optimization of 2-manifold surfaces, which aims to satisfy global objectives by displacing an offset surface derived from a given initial shape.
- We provide an elegant and efficient basis-reduction approach that is numerically robust and speeds up the computation considerably by making the number of optimization variables independent of the mesh resolution.
- We demonstrate the applicability of our approach by controlling the mass properties of a rigid body enclosed between an inner and an outer offset surface.

2 Related Work

Design Optimization Problems. Design optimization problems aim at the automatic computation of structural or mechanical designs that suit some desired (global) goals. They have been studied in computational industrial design as form-finding problems, as well as in statics and mechanical engineering as structural optimization problems [Haftka and Gürdal 1992]. In computer graphics, recent approaches provide algorithms for improving models for 3d printing, like the computation of structural stability [Stava et al. 2012], worst-case structural analysis [Zhou et al. 2013], cost-effective material usage [Wang et al. 2013], or optimization of both the strength and weight of printed objects [Lu et al. 2014].

Various methods have been proposed that integrate computational design into interactive modeling tools in order to solve specific problems. For instance, interactive systems for various manufacturing planning tasks, like garment editing [Umetani et al. 2011], design of physically valid furniture [Umetani et al. 2012], articulated 3d-printed models [Bächer et al. 2012], mechanical assemblies, like toys [Zhu et al. 2012] and various moving characters [Coros et al. 2013; Thomaszewski et al. 2014], or construction of inflatable balloons with desired shapes [Skouras et al. 2012] have been proposed. Another example is material design, where the specification of a desired deformation behavior can be given *a priori*, and the composite material with respective elastic behavior is computed by optimization [Bickel et al. 2010]. This approach was also extended to the construction of balloons with prescribed shape [Skouras et al. 2012], and for the manufacturing of synthetic clones of human faces [Bickel et al. 2012].

Reduced-Order Models. Order-reduction approaches aim at lowering the dimensionality of the parametric space of a computational problem while preserving its input-output behavior as much as possible. Their goal is in general to gain performance. In the area of physically based deformation and animation, the idea of using vibration modes of a body for order reduction has been proposed by Pentland and Williams [1989]. This approach has been further explored for efficient interactive animation [Kim and James 2009] and shape deformation [von Tycowicz et al. 2013]. In geometry processing, space and surface deformation methods based on skinning between handles or cages are reduced-order approaches. The idea is to express the problem with respect to few handles that define a subspace, and to interpolate or approximate the overall deformation [Botsch and Kobbelt 2004; Sumner et al. 2007; Jacobson et al. 2011]. In mesh processing, multi-resolution modeling [Kobbelt et al. 1998] can be seen as an example of order reduction.

Computation of Mass Properties. The computation of global mass properties has a long tradition in the modeling and CAD communities, since from the beginning of computerized modeling, exact parameters of modeled objects were of great importance for animation, simulation, as well as manufacturing. Two early works pioneered the idea to utilize Gauss's Divergence Theorem for the computation of mass moments of polyhedral objects [Messner and Taylor 1980], and parametric bi-cubic spline patches [Timmer and Stern 1980]. Recently, it has been utilized for the optimization of mass properties of 3d-printed models [Prévost et al. 2013; Bächer et al. 2014; Christiansen et al. 2015], which we also demonstrate in this paper.

3 Problem Formulation

In this section we present the general concept of our shape optimization approach. The basic idea is to interpret the shape as a solid enclosed between two surfaces, where each can be deformed by the application of offsets with spatially varying thickness.

Definitions and Notation. As input to our method, we expect a 3d shape represented as a closed oriented 2-manifold surface $\mathcal{S} =$

$(\mathcal{X}, \mathcal{T})$, which in practice is a triangle mesh composed of n vertices $\mathbf{x} \in \mathcal{X}$ and a set of triangles $\mathcal{T}$. Usually, the surface is the boundary of a solid body. Alternatively, the input can be an inner and an outer surface enclosing a solid between them.

Depending on the desired optimization task, our method can output a single offset surface or two complementary offset surfaces, one oriented to the outside and one to the inside of the input shape, where we denote the outer one as $\overline{\mathcal{S}}$, the inner one as $\underline{\mathcal{S}}$, or both together as $\underline{\overline{\mathcal{S}}}$. For simplicity, we explain the concepts by using the outer surface $\overline{\mathcal{S}}$, except where both surfaces are involved. Figure 2 provides an overview.

Offset Surfaces. Each output surface is created by adding an offset value δ in a direction $\boldsymbol{\nu}$ at each surface vertex $\mathbf{x}$, such that $\overline{\mathbf{x}} = \mathbf{x} + \delta\boldsymbol{\nu}$, where $\boldsymbol{\nu} \in \mathbf{V}$ is a direction vector from a vector field $\mathbf{V}$, $\delta \in \mathbb{R}$ is a scalar that provides the magnitude and the sign of the shift, and $\overline{\mathbf{x}} \in \overline{\mathcal{X}}$ is a vertex of the offset surface $\overline{\mathcal{S}}$. One obvious choice for $\mathbf{V}$ would be the surface normal field $\mathbf{N}$, however, general offset surfaces are not limited to this choice, and we use a vector field derived from iterative surface contraction as described in Section 4.

In the following, the vertices $\mathbf{x}$ of a mesh are organized in a matrix $\mathbf{X}$, displacement vectors $\boldsymbol{\nu}$ in $\mathbf{V}$, and the scalar offsets δ in the vector $\boldsymbol{\delta}$.

Optimization Problem. Assuming a given vector field $\mathbf{V}$, our goal is to find the n optimal offsets δ such that the shape satisfies a given objective. In order to remain general, we first define a template functional for the resulting optimization problem as

$$E_S = \min_{\boldsymbol{\delta}} f(\boldsymbol{\delta}) \quad \text{s.t.} \quad g_i(\boldsymbol{\delta}) \leqslant 0 \quad \text{and} \quad \boldsymbol{\delta}_l \preceq \boldsymbol{\delta} \preceq \boldsymbol{\delta}_u\,, \qquad (1)$$

where f is the objective function, g_i are additional hard equality and/or inequality constraints, and $\boldsymbol{\delta}_l$ and $\boldsymbol{\delta}_u$ are lower and upper boundary constraints. For instance, f could be the goal to lower the z-position of the center of mass of an object, and g could be the constraints to keep the center of mass in a certain xy-region. Boundary constraints are needed to prevent offset surfaces from intersecting each other or from reaching other implausible values.

The functions f and g_i are usually non-linear in the deformation of the surfaces, making the problem a non-linear programming task (NLP), which can be generally approached with existing standard numerical routines. However, a significant disadvantage of this formulation is that the problem is highly underdetermined, i.e., there usually exist many offset surfaces that satisfy the equation. Even worse, the number of offsets δ equals the number of vertices (i.e., n), and therefore it is high-dimensional and scales badly with increasing mesh resolution. In other words, the problem is redundant and expensive to solve, hence, barely suitable for practical purposes.

As a major contribution of this paper, we introduce a reduced-order formulation that provides a remedy for these issues. Our solution is to project the problem onto a lower-dimensional basis determined by *manifold harmonics* [Vallet and Lévy 2008], where we can solve it more efficiently using standard numerical routines—independently of the number of input vertices. In Section 5 we present the details of this approach.

4 Offset Directions and Bounds

One significant problem of offset surfaces for 2-manifolds is that too large displacements may cause the target surface to penetrate itself and lose its 2-manifoldness, becoming unusable for practical applications (cf. Figure 3, middle).

Such self-intersections can be either local or global. Local foldovers appear if, at any point of the surface, the orthogonal distance

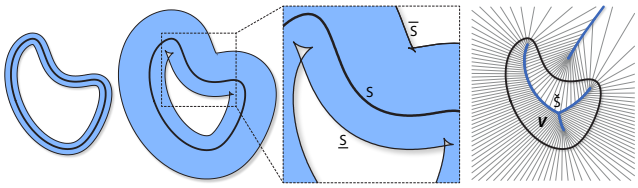

Figure 3: *Left to right: inner and outer offset surfaces with increasing offset to the original. Middle: note the self-intersections where offsets get too large. Right: medial axis as a global bound for maximal offset.*

of the offset to the surface becomes equal or higher than the radius of the maximal principal curvature κ_{max}. Global self-intersections happen if offsets from distant regions of the surface intersect in the interior or within concavities. These problems can be approached using the *shape skeleton* as the upper bound for the displacements (cf. Fig 3, right). If it is appropriately chosen, i.e., it approximates the medial axis, which is the set of all centers of spheres that touch $\mathcal{S}$ in at least two points, and the offsets do not exceed it, both global and local self-intersections can be avoided.

Shape Skeleton. Since we are working with piecewise linear triangle meshes, in practice, the medial axis is neither easy to compute exactly, nor is it necessarily smooth at sharp corners. For this reason, we strive for an approximation that is robust to compute and provides smooth skeletons.

We adapt the idea presented by Tagliasacchi et al. [2012], where the original surface is contracted iteratively using constrained Laplacian smoothing until it converges to a skeleton $\check{\mathcal{S}}$. The advantage of this method over other solutions is that it is much less sensitive to surface detail and that it provides quite smooth skeleton approximations, even at sharp corners of the input shape (cf. Fig. 4, left). Recursive application of the contraction scheme results in a mesh that converges to a one-dimensional curve, but in early contraction stages, it forms a so-called *meso-skeleton* $\check{\mathcal{S}}$, which approximates the medial surface. That surface is especially useful in the case of complex objects (cf. Fig. 4, right), since it helps keep a one-to-one correspondence with the vertices.

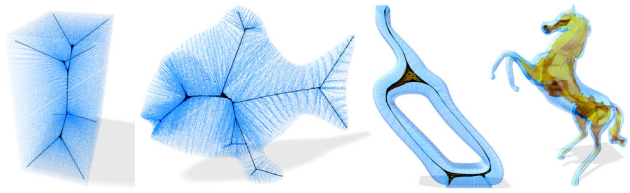

Figure 4: *Left: Two examples of skeletons with a direct one-to-one correspondence between outer and inner vertices and the resulting vector field. Right: two meso-skeletons (cf. Section 4).*

4.1 Offset Vectors

Inner Offset Vectors. By keeping track of the correspondences of the input mesh vertices $\mathbf{x}$ to the skeleton vertices $\check{\mathbf{x}}$, we obtain a surjective correspondence between $\mathcal{S}$ and $\check{\mathcal{S}}$ (Fig. 4). We can now derive the vectors for inner offsets as $\underline{\boldsymbol{\nu}} = \check{\boldsymbol{\nu}}/\|\check{\boldsymbol{\nu}}\|$, with $\check{\boldsymbol{\nu}} = (\check{\mathbf{x}} - \mathbf{x})$. Please note that these vectors point inwards, and are not necessarily normal to the surface. However, since they were created by consecutive contraction, they are rather smooth and provide naturally suitable trajectories for surface offsets.

Outer Offset Vectors. For the direction of outer offsets we have, among others, the choice of taking the outward-pointing normal vectors $\overline{\boldsymbol{\nu}} = \mathbf{n}$ of the input mesh, or taking the inverse contraction vectors $\overline{\boldsymbol{\nu}} = -\underline{\boldsymbol{\nu}}$. Usually we favor the latter option, since the

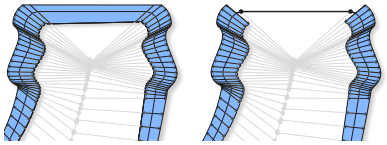

Figure 5: *Left: surfaces are offset independently. Right: the hole has been constrained to have no offsets.*

contraction vector field is of rather low frequency and thus less sensitive to fine details of the input surface than the normals, which can exhibit many high-frequency fluctuations.

4.2 Offset Bounds

Maximal Bounds. Given the offset vectors, we can use them to provide constraints for minimal and maximal displacements δ_l and δ_u. In the case of inner offset surfaces $\underline{S}$, we obtain the individual maximum displacement values as $\underline{\delta_u} = \left[\, |\check{\nu}_1| \, |\check{\nu}_2| \ldots |\check{\nu}_n| \,\right]^T$, which is the 'safe' distance to the skeleton. The choice of the outer offsets $\overline{\delta_u}$ is usually not as critical, since we try to keep the shape of the object as similar to the input as possible. We therefore set it to a low value, which can be adjusted for particular models individually. However, self-intersections can still happen as shown in Fig. 3, right.

Minimal Bounds. In the case of a double surface $\overline{S}$, it is often of interest to provide a minimal or maximal distance between the surfaces. For instance, it is often necessary to meet practical fabrication considerations, e.g., 3d-printer resolution. However, since the offset vector field is in general not orthogonal to the surface, these values cannot be measured directly along it. In order to still allow for such constraints, we project the desired minimal thickness values in the normal direction δ_{l_n} onto particular vectors ν to get a minimum offset in direction of the vector ν: $\delta_l = \frac{1}{\nu^T n}\delta_{l_n}$, where n is the unit surface normal. Thus, the minimal distances between $\underline{S}$ and $\overline{S}$ along V in both directions are given by $\overline{\underline{\delta_l}} = \left[\, \underline{\delta_l}^T \; \overline{\delta_l}^T \,\right]^T$.

Geometric Bounds. An additional option we provide are explicit geometric constraints that allow forcing selected regions to desired offsets. Technically, we accomplish this by overriding the selected offsets $\hat{\delta} \subset \delta$ by setting them to desired values. Especially, by setting $\hat{\delta} = 0$, regions obtain no displacement, which is of interest if the surfaces $\underline{S}$ and $\overline{S}$ should be forced to coincide, which creates openings without affecting the volume (cf. Figure 5).

Offset Penalty. In order to preserve the visible shape, outer offsets could be suppressed by adding their squared sum to the objective, weighted by w_p: $w_p \|\overline{\delta}\|_2^2$. Moreover, this type of penalization could be applied to any other subset of the displacements δ, or used as regularization, as discussed in Section 7.1.

5 Order Reduction

In this section we introduce an efficient order reduction for the optimization problem stated in Equation (1) by transforming it into a spectral representation denoted as manifold harmonics.

5.1 Manifold Harmonics

Essentially, manifold harmonics resemble the bases of the Fourier transform on meshes with arbitrary topology [Taubin 1995], and exhibit a set of advantages we can exploit: first of all, if appropriately chosen, they are orthogonal, making the basis transformation a numerically stable operation. Second, they are smooth, allowing for well-defined continuous optimization. Finally, in terms of an arbitrary manifold mesh, they have the advantage that they "encode" the geometry and topology of the original object [Levy 2006], such that their extrema usually lie at geometrically exposed locations and capture the intrinsic symmetry of the shape [Zhang et al. 2010]

Discrete Laplace-Beltrami. The manifold harmonic functions can be computed as the eigenfunctions of the Laplace-Beltrami operator Δ_S of the input surface. This operator is a generalization of the Laplacian operator to 2-manifolds and allows performing differential operations on surfaces. Generally, it is defined as the divergence of the gradient, i.e., the sum of second partial derivatives of a parameterized surface. However, in a discrete setup, the Laplace-Beltrami operator L can be derived as the umbrella operator directly from the mesh, without the usage of a parameterization.

In the literature, there are a number of propositions how to discretize the differential operator. In our implementation, we follow Pinkall and Polthier [1993] due to its symmetry:

$$L_{i,j} = \begin{cases} \omega_{i,j} & \text{if } (i,j) \in \mathcal{E} \\ \sum_{k \in \mathcal{N}(i)} -\omega_{i,k} & \text{if } (i = j) \\ 0 & \text{otherwise}\,, \end{cases} \tag{2}$$

where $\mathcal{E}$ denotes the set of edges, and $\mathcal{N}(i)$ is the set of first-order neighbors of vertex i. The weights $\omega_{i,j}$ are computed using the geometric properties of the mesh, in particular:

$$\omega_{i,j} = \frac{1}{2}\left(\cot \varphi^l_{i,j} + \cot \varphi^r_{i,j}\right)\,,$$

where $\varphi^l_{i,j}$ and $\varphi^r_{i,j}$ are the angles opposite to the edge (i,j) in the left and right incident triangles respectively.

Harmonic Basis. We have chosen this operator because it is symmetric and positive semi-definite, thus, respecting the spectral theorem, all its eigenvalues λ_i are real and non-negative (i.e., $\lambda_i \geqslant 0$), and all eigenvectors γ_i are mutually orthogonal, and can be further orthonormalized such that $\forall\, i\colon \|\gamma_i\|_2 = 1$. They can be computed by the diagonalization $L = \Gamma\Lambda\Gamma^T$, where the diagonal entries of the matrix Λ are the eigenvalues λ_i, such that $\lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_n$, and the columns of the matrix $\Gamma = \left[\, \gamma_1 \; \gamma_2 \ldots \gamma_n \,\right]$ are the respective eigenvectors. From the functional analytic point of view, the eigenvectors become the *eigenfunctions*, and since they satisfy the Laplace equation, they are denoted as *harmonics*.

The basis functions $\left[\, \gamma_1 \; \gamma_2 \ldots \gamma_n \,\right]$ correspond to the lowest to highest frequencies of the input mesh, thus the first few functions capture the global shape appearance and the remaining ones capture the details. For this reason, a low-frequency base surface can be well approximated using only the first few components.

In practice, the computation of the full diagonalization of a large matrix is a significant computational challenge. However, since we are only interested in a reduced set of k bases, there exist a number of efficient algorithms that can be utilized for our purpose very well. We have used the implementation from the ARPACK library [Lehoucq et al. 1998]. An additional advantage is that we only need to compute them once per mesh.

5.2 Reduced Optimization Problem

The main idea of order reduction is to represent the offsets δ in the manifold harmonic basis. This allows us to significantly reduce the order of the problem. Simultaneously, adjusting the low frequencies only enables us to perform desired changes of the overall shape while still preserving the local details.

In order to achieve this, we express the offset surface $\overline{S}$ as a linear combination of a set of k eigenfunctions $\Gamma_k = \left[\, \gamma_1 \; \gamma_2 \ldots \gamma_k \,\right]$:

$$\overline{x}_i = x_i + \sum_{j=1}^{k} \alpha_j \gamma_{ij}\, \overline{\nu}_i\,,$$

where α_j are elements of the unknown coefficient vector $\alpha \in \mathbb{R}^k$, and the problem now reduces to finding that vector. This drastically

lessens the degrees of freedom of the problem, and also serves as an implicit regularization (discussed in more detail in Section 7). We can now write the optimization from Equation (1) in terms of $\boldsymbol{\alpha}$ as

$$E_S = \min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) \quad \text{s.t.} \quad g_i(\boldsymbol{\alpha}) \leqslant 0 \quad \text{and} \quad \delta_l \preceq \Gamma_k \boldsymbol{\alpha} \preceq \delta_u \,, \tag{3}$$

which has the benefit of resulting in $k \ll n$ optimization unknowns. Please note that this number is independent of the mesh resolution—only the number of chosen harmonic bases is relevant. In our experiments, most offset meshes can be approximated adequately with $k \leqslant 36$ basis vectors (cf. supplemental material).

The geometric constraints can be enforced by truncating the corresponding basis functions, such that $\forall j\colon \gamma_{ij} = 0$. The result is that during the optimization, the changes in $\boldsymbol{\alpha}$ have no effect on the selected vertices $\boldsymbol{v}_i$. Hence, we can also relax the respective box constraint to $\delta_{li} = -\infty$ and $\delta_{ui} = +\infty$ such that they would be able to develop freely. While the surface loses the smoothness at those vertices, the final result is not affected, since we override them with the desired values.

Finally, the harmonic basis also allows us to provide an option to penalize the outer displacements in order to preserve the original shape of the object. We do this by minimizing the squared magnitude of the coefficient $\overline{\alpha}_1$, where we weight the strength of the penalty with w_p:

$$E_S = \min_{\boldsymbol{\alpha}} \left(f(\boldsymbol{\alpha}) + w_p \overline{\alpha}_1^2 \right) .$$

Since the first element corresponds to the constant zero-frequency (DC element), it changes the volume, while the other components contain the detail, but have zero mean.

5.3 Analytic Gradient

Given an objective function f that can be differentiated w.r.t. the surface vertices $\mathbf{X}$ analytically, our formulation of shape optimization using offset surfaces allows for a closed-form analytic computation of the gradient of the functional in Eq. (3). Thus, a major advantage of our formulation is that the gradient can be calculated by repeated application of the chain rule as

$$\nabla E_S = \frac{\partial f}{\partial \boldsymbol{\alpha}} = \frac{\partial f}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \delta} \frac{\partial \delta}{\partial \boldsymbol{\alpha}} ,$$

where $\mathbf{X} = \begin{bmatrix} x_1 & y_1 & z_1 & x_2 & y_2 & z_2 & \dots & x_n & y_n & z_n \end{bmatrix}^T$ are the n vertices of the original surface concatenated into a $\langle 3n \times 1 \rangle$ vector, and δ is the $\langle n \times 1 \rangle$ vector of corresponding scalar displacements.

The derivatives $\partial f / \partial \mathbf{X}$ with respect to surface vertices $\mathbf{X}$ result in a $\langle 1 \times 3n \rangle$ vector. The derivatives of $\partial \mathbf{X} / \partial \delta$ result in a sparse matrix of the size $\langle 3n \times n \rangle$, which contains in each column the respective displacement vector $\boldsymbol{v}$. Eventually, the derivatives $\partial \delta / \partial \boldsymbol{\alpha}$ are the harmonic basis vectors Γ_k, which form a $\langle n \times k \rangle$ matrix. The final partial derivatives of the objective $\partial f / \partial \boldsymbol{\alpha}$ with respect to $\boldsymbol{\alpha}$ thus reduce to a $\langle 1 \times k \rangle$ sized vector. The derivatives for the constraint functions g_i can be computed accordingly.

Please note that the given matrix sizes are with respect to only one-sided displacement with k coefficients. For two-sided offsets, the matrices need to be extended accordingly, which is explained in more detail in the supplemental material.

6 Applications and Results

6.1 Control of Mass Properties

We demonstrate the application of our method by optimizing mass properties of a solid. In particular, the *physical mass moments* of a rigid body are attributes that determine how an object behaves under the influence of mechanical forces, e.g., gravity, torque, etc. The moments of zeroth, first, and second order of a rigid object defined by its surface S describe the total mass $m(S)$, the center of mass $\mathbf{c}(S)$, and the symmetric $\langle 3 \times 3 \rangle$ tensor of inertia $\mathbf{I}(S)$, respectively. We summarize these 10 properties as

$$\mathbf{P}(S) = \begin{bmatrix} m & c_x & c_y & c_z & I_{xx} & I_{yy} & I_{zz} & I_{xy} & I_{yz} & I_{zx} \end{bmatrix}^T . \tag{4}$$

The tensor of inertia $\mathbf{I}(S)$ is symmetric and contains the quadratic terms, denoted as moments of inertia, on the diagonal, and the mixed terms, denoted as products of inertia, on the off-diagonals. Please refer to the supplementary material for a derivation and further details. While the physical mass moments are integrals over the volume of the solid, uniform mass density and Gauss's Divergence Theorem allow evaluating them as functions of the boundary surface of the solid $S = \{\mathcal{X}, \mathcal{T}\}$, which we also describe in detail in the supplementary material.

The object's mass m relates to its volume V by $m = \rho V$, where ρ is the material density coefficient. Without loss of generality, for solid objects with uniform mass density distribution, we can assume $\rho = 1$. Note that in the following, all mass densities are given relative to the object mass density, i.e., all are divided by the mass density of the object, and the density of air is set to 0.

6.2 Objectives

We provide a number of specific sets of objectives and constraints in order to fulfill certain tasks that require the control of mass properties. Often, we want to control the static and rotational stability of shapes. These goals can be specified by placing the object in the origin of the world coordinates in the desired pose.

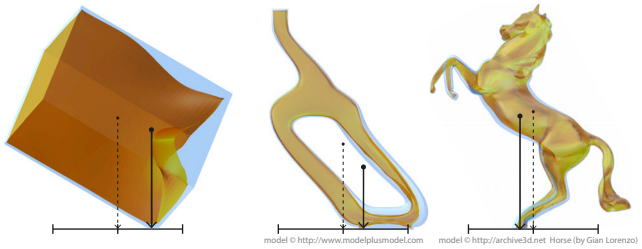

Figure 6: *Examples of static stability optimization. The dashed lines indicate the center of mass of the entire solid, the thick lines the ones of the shell after optimization.*

Static Stability. An object stands stably if the orthogonal projection of its center of mass $\mathbf{c}$ on the ground plane lies within the convex hull defined by the object's ground contact points. We place the object in the xy-plane such that the centroid of the convex hull is in the origin, and we optimize the following objective:

$$\begin{aligned} f(\boldsymbol{\alpha})_{\text{static}} &:= c_x^2 + c_y^2 + c_z \,, \\ g_i(\boldsymbol{\alpha})_{\text{static}} &:= (c_x + c_y)^2 - (r - \varepsilon)^2 \leqslant 0 \,, c_z > 0 \,, \end{aligned} \tag{5}$$

where r is the radius of the largest inscribing circle of the convex base polygon, and ε is a safeguard (cf. Figure 6).

Static Stability under Storage. A storage container alters its mass properties when the initial void is filled with the stored medium. To ensure static stability in both states, we optimize both the empty and the filled container. Assuming a uniform mass density ρ of the stored medium, the mass of the empty and filled container is given by $m_{\text{empty}} = m(\overline{S})$ and $m_{\text{full}} = m_{\text{empty}} + \rho V(S)$ respectively. The center of mass of the empty and filled container is given by

$$\mathbf{c}_{\text{empty}} = \mathbf{c}(\overline{S}) \quad \text{and} \quad \mathbf{c}_{\text{full}} = \frac{m_{\text{empty}} \mathbf{c}_{\text{empty}} + \rho V(S) \mathbf{c}(S)}{m_{\text{full}}} . \tag{6}$$

We now optimize these centers of mass simultaneously by applying objective (5) to both. Figure 1, right, depicts a leaning bottle that has been optimized to stand stable if filled with water.

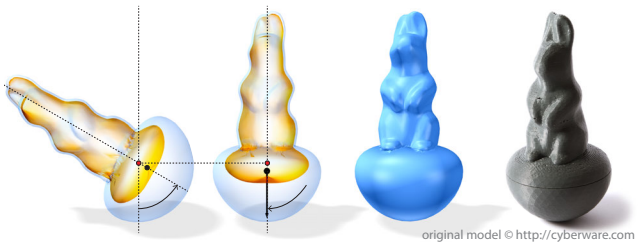

Figure 7: *A roly-poly toy designed with our method. The center of mass (black dot) is placed below the center of the hemisphere at the bottom of the object (red dot). When pushed, a righting moment corrects the pose (left) until the equilibrium position is reached (center). A rendering and a fabricated replica is shown on the right.*

Monostatic Stability. An object is called monostatic if it has only a single stable resting position. If perturbed, its shape and inner mass distribution produces a righting moment that returns it to this equilibrium position. A common application of this principle are roly-poly toys. As shown in Figure 7, they commonly consist of a hemispherical element at the bottom and a figurine on top. The correct righting behavior is obtained when the center of gravity $\mathbf{c}$ lies below the hemisphere center, so we optimize a variant of (5):

$$\begin{aligned} f(\boldsymbol{\alpha})_{\text{static}} &:= c_z\,, \\ g_i(\boldsymbol{\alpha})_{\text{static}} &:= \{\,c_x\,,\,c_y\,\}=0\,, c_z - r\, + \varepsilon < 0\,, \end{aligned} \tag{7}$$

where r denotes the radius of the hemisphere and ε acts as a safeguard against tolerances in the fabrication process, which could raise a center of gravity that is placed just below the hemisphere center (cf. Fig. 7).

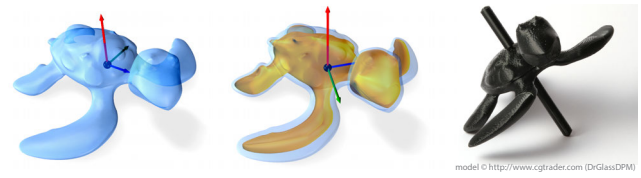

Figure 8: *A spinning turtle model. Left, the original model, right, an object optimized for spinning.*

Rotational Stability. An object rotates stably about an axis if the axis is its smallest or largest principal axis of inertia [Goldstein et al. 2002]. Hence, we place the object in a coordinate frame in a pose we want it to spin, with $\vec{z}$ as the up and rotation axis (see Fig. 13). Then we optimize the body such that its principal axis equates the rotation axis with moment $I_c = I_{zz}$, and such that the cross terms I_{xz} and I_{yz} vanish. Here we adapt the objective as recently proposed by Bächer et al. [2014]:

$$\begin{aligned} f(\boldsymbol{\alpha})_{\text{inertia}} &:= m c_z + \left(\frac{I_a}{I_{zz}}\right)^2 + \left(\frac{I_b}{I_{zz}}\right)^2, \\ g_i(\boldsymbol{\alpha})_{\text{inertia}} &:= \{\,c_x\,,\,c_y\,,\,I_{xz}\,,\,I_{yz}\,\}=0\,, c_z\, > 0\,. \end{aligned} \tag{8}$$

In general, the remaining moments of inertia I_{xx} and I_{yy} do not coincide with the coordinate frame axes. We obtain the principal axis moments I_a and I_b as the eigenvalues of the $\langle 2\times 2\rangle$ upper-left part of the inertia tensor $\mathbf{I}$:

$$\{\,I_a\,,\,I_b\,\} = \frac{1}{2}\left(I_{xx}+I_{yy} \mp \sqrt{I_{xx}^2 + 4I_{xy}^2 - 2I_{xx}I_{yy} + I_{yy}^2}\right).$$

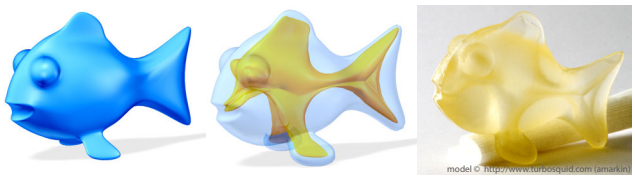

Figure 9: *An object optimized for buoyancy in a liquid of a specific density ρ. By adding a void with appropriate shape and volume inside the object, buoyancy can be achieved. Left: input model, center: optimized shape, right: 3d-printed example.*

Specific Volume and Buoyancy. Our method allows the creation of objects whose inner void should exhibit a specific volume. Given a target volume V, its difference to the volume of the void is given by $(V - V(\underline{S}))$ and can be penalized either in the objective function or enforced by a constraint.

A more complex application of this capability is to control the buoyancy of an object. To ensure static stability of an object that is immersed in a fluid, its gravitational and buoyant forces should be at equilibrium in the desired orientation of the object. We assume for simplicity that our object is incompressible and completely submerged in a fluid with a uniform mass density $\rho < 1$ with downward gravity $\mathbf{g} = -g\vec{z}$. A buoyant force $V(\overline{S})g\vec{z}$ is exerted on the center of buoyancy $\mathbf{c}_{\text{buoy}} = \mathbf{c}(\overline{S})$, while the gravitational force $-m(\overline{S})g\vec{z}$ acts at the center of gravity $\mathbf{c} = \mathbf{c}(\overline{S})$. Equilibrium is established if the magnitudes of the forces cancel, i.e., $0 = V(\overline{S})\rho g - m(\overline{S})g = (\rho-1)V(\overline{S}) + m(\underline{S})$, and if no torque is produced on the object, i.e., $c_{\text{buoy,x}} = c_x$ and $c_{\text{buoy,y}} = c_y$. Furthermore, a stable equilibrium is only reached if $c_{\text{buoy,z}} > c_z$. Otherwise, the object would flip upside down. For a floating object, we consequently optimize

$$\begin{aligned} f(\boldsymbol{\alpha})_{\text{buoyancy}} &:= \left(\,(\rho-1)V(\overline{S}) + m(\underline{S})\,\right)^2, \\ g_i(\boldsymbol{\alpha})_{\text{buoyancy}} &:= \{\,c_x - c_{\text{buoy,x}}\,,\,c_y - c_{\text{buoy,y}}\,\}=0\,, \\ & \qquad c_z < c_{\text{buoy,z}}\,. \end{aligned} \tag{9}$$

A result of an object optimized for buoyancy is provided in Fig. 9.

6.3 Implementation

Implementation. We implemented the optimization framework using MATLAB 2014a and C++, where we utilized the LIBIGL library [Jacobson et al. 2014] for some computations. For the computation of skeletons we used the CGAL implementation of [Tagliasacchi et al. 2012]. For the computation of the eigendecomposition, we utilized the sparse function `eigs`, which implements the ARPACK routines [Lehoucq et al. 1998]. For the constrained NLP-problem, we used the MATLAB Optimization Toolkit using `fmincon`, in particular the constrained medium-scale active-set solver, however, it also works well with the large-scale interior-point solver, or could be easily plugged into another numerical routine. Our experimental MATLAB/C++ code will be available on the paper web page.

Timings. The preprocessing timings lie generally in the range of several seconds. In particular, the convergence time of the skeleton depends on the chosen parameters, but it usually ranges between 5−30s. The longest computation we observed was the Horse model (cf. Figure 6, 86k vertices) with 84s. The computation of the sparse Laplacian and its eigenvectors depends on the chosen k, but even for large meshes (e.g., the Horse with $k = 36$), it takes less than 4s to compute. The optimization time for our examples is usually between a few seconds in simple cases (e.g., Bottle in Figure 1), up to few minutes for more complex examples (e.g., Horse 3.8min). Timings were taken on an Intel(R) Core(TM) i7-3770K CPU@3.50

GHz with 32 GB RAM running Windows 8; please refer to the supplemental material for detailed timings.

Fabrication. We have 3d-printed several models using different 3d-printers: a MakerBot Replicator 2 for early testing, a Dimension uPrint Plus for prototyping and most of the results, and finally also an Objet Eden 260 for poly-jet prints for high-quality results. Since the FDM prints are in general not entirely watertight, we impregnated them with a clear-coating material. Our models worked well with these printers, however, usually they needed to be cut into pieces in order to be printed with support material, and glued together after a base-bath. We performed the cutting manually using a standard modeling software, but also automatic methods for the partitioning of objects [Luo et al. 2012] are available.

7 Discussion

7.1 Harmonic Basis Functions

Basis Dimensionality. The number k of basis-functions allows a trade-off between reduced runtime and increased robustness on the one hand and a solution closer to a reference optimum on the other. For an evaluation, we compare the reduced-order results to a straightforward shape optimization of each individual vertex.

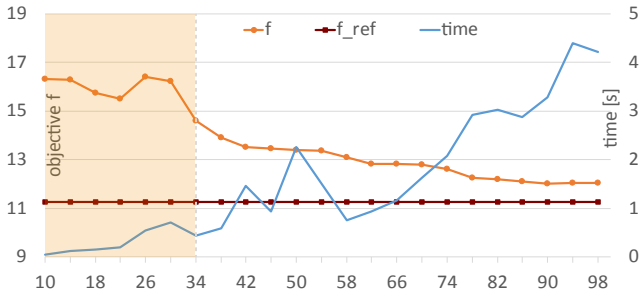

Figure 10: *Number of used basis functions* k *compared to the objective value (orange), processing time (blue), and a reference objective (brown). The reference is* $f_{ref} = 11.28$ *and* $t_{ref} = 87s$*. The object has* $n = 750$*. The results with a low number of basis functions (shaded region) did not converge within the posed constraints.*

In Figure 10, we plot the value of the objective f versus increasing values of k measured on the object shown in Figure 11 with $n = 750$ and the objective (8). If too few basis functions are chosen, the constraints cannot be fulfilled, such that a solution does not exist (shaded area). However, already a small number of modes ($k = 34$) permits a solution that is remarkably good and fast to obtain. Thus, a further increase in k only yields minute improvements while increasing the computation time.

The speedup of our algorithm is in any case significant. If we optimize the shape with 34 basis functions (the dashed vertical line in Fig. 10), we obtain a very good approximation within 0.44 seconds, compared to 87 seconds for the full solution, which is a speedup of 2 orders of magnitude. Indeed, in practice, the speedup for large models (i.e., $n >> 1000$) is even higher, since the full per-vertex computation of such models takes hours or even days.

We have also computed the optimal shape using a full set of basis functions ($k = n$). We obtained the same optimality value ($f = 11.27$) at a runtime of 93s, and a surface very similar to the one delivered by the full solution in the Euclidean space (cf. Figure 11, right). This is to be expected, since the problem is in theory the same, the differences of the final geometry are due to numerical approximation errors. Thus, the general conclusion is to choose the number k low; only sufficiently high to express the shape with the necessary number of degrees of freedom needed to fulfill the constraints.

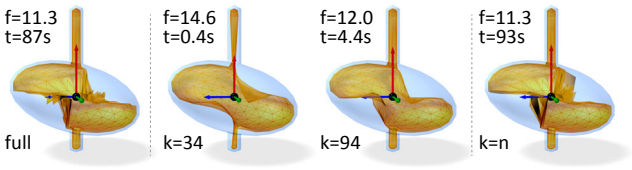

Figure 11: *Results of the evaluation in Figure 10 with* $n = 750$*. Left to right: result of the full method, results with increasing* k*.*

Regularization. An important aspect of our solution is an implicit regularization. Since the full problem is highly underdetermined, there exist many solutions, and the finally obtained one is not necessarily very smooth, as depicted in Figure 11, left. This problem could be approached with Tikhonov regularization by adding an additional term $w_r \|\delta\|_2^2$ to the objective, which would favor small displacements and make the problem fully determined, however, at the cost of its size and computation time. Additionally, we found the results still not smooth. In contrast, using a small number of harmonics allows us to express the displacement with low-frequency basis functions, such that each approximated result provides a smooth surface in the Laplacian sense.

Choice of the Basis. We have chosen the Laplacian as in (2) since it reflects the object's geometry, and it is symmetric positive semi-definite. However, there are other possible approximations of manifold harmonics, as for instance discussed by Vallet and Levy [2008]. We have experimented with them and found that a more accurate discretization of the operator results in a better shape approximation and faster convergence. Thus, a detailed investigation of this issue would be a possible direction for future work.

Surface Detail. A final point to note is that the outer offset surfaces themselves retain the detail of original surfaces. This is because only the offsets are projected to a low-frequency space, not the surface itself.

7.2 Comparisons

In Figure 12, we show a comparison to the state-of-the-art in the form of Make-It-Stand [Prévost et al. 2013], which we use without outer-surface deformation by removing the scaling and deformation terms and performing the plane-carving algorithm only. We also set the wall thickness in our case to a similar distance as the lowest possible 1 voxel in the other method. The figure shows that in this case, our method indeed approximates the interior void better and provides slightly more stable mass properties:

$$\text{our: } f = 2.52\,, \quad m = 21.7\,, \quad \mathbf{c} = [\,0.63 \;\; 0.0 \;\; 2.13\,]\,,$$
$$\text{ref: } f = 2.96\,, \quad m = 26.1\,, \quad \mathbf{c} = [\,0.78 \;\; 0.0 \;\; 2.35\,]\,.$$

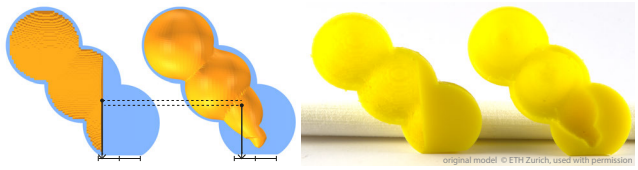

Figure 12: *Comparison with the method of [Prévost et al. 2013] without outer deformation of the shape. Both results achieve static stability as evidenced by their fabricated replicas (right), however, our method provides a slightly better result.*

In Figure 13 we show a comparison of our method with the one of Bächer et al. [2014]. We set $k = 24$, and also in this case only the optimization of the interior has been used in both cases. Our method converged in 6.8s. Below, we provide the mass moments

of both results, where we can see that we achieve nearly identical values as the reference, while providing a smooth inner surface:

$$\text{our: } f = 11.4\,, \quad \mathbf{P} = [\,1.07 \mid 0 \;\; 0 \;\; 1.13 \mid 0.35 \;\; 0.41 \;\; 0.52 \mid 0 \;\; 0 \;\; 0\,],$$
$$\text{ref: } f = 8.16\,, \quad \mathbf{P} = [\,1.20 \mid 0 \;\; 0 \;\; 1.13 \mid 0.34 \;\; 0.36 \;\; 0.57 \mid 0 \;\; 0 \;\; 0\,].$$

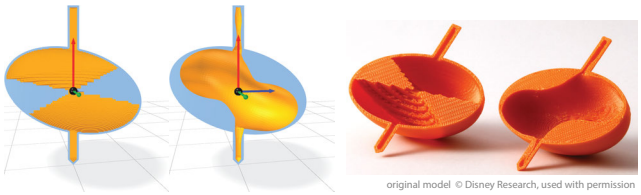

Figure 13: *Comparison with Spin-It [Bächer et al. 2014]. The input unstable spinning top is optimized using their (far left) and our method with* $\mathrm{k} = 24$ *(center left). Right: fabricated results.*

7.3 Limitations

Deformation Limitations. Other approaches [Prévost et al. 2013; Bächer et al. 2014] utilize linear blend skinning (LBS) with bounded biharmonic weights [Jacobson et al. 2011] for the adjustment of the objects' shapes. Using this type of deformation with well-defined similarity transformations (scale, translation, rotation) enables their methods to deform the model in a meaningful manner with more degrees of freedom than given by our outer displacement. Additionally, careful manual placement of control handles enables the user to influence the deformation semantically, i.e., by adding handles to the extremities of articulated objects, the system can explicitly account for their pose [Prévost et al. 2013]. In contrast, even if the basis functions have global support, our displacement can deform the shape only along a predefined vector field, which is a much more restricted shape-editing operation, and it cannot change the objects' pose (cf. Figure 14).

Boundary Constraints Limitations. The presented optimization pipeline is in general very robust. However, the algorithm also depends on the quality of the provided skeleton. We have experimented with several approaches, and found the solution of Tagliasacchi et al. [2012] to be the most reliable, since it provides an approximation of the medial surface, denoted as meso-skeleton, that is a trade-off between the medial axis and a smooth 1d skeleton. Nonetheless, their approach still requires tuning of parameters. This issue is crucial, since the offset surface is shifted along vectors toward the skeleton, hence there must exist a surjective (in the continuous sense) mapping between $\mathcal{S}$ and $\check{\mathcal{S}}$. If the vectors $\check{\mathbf{v}}$ intersect or cross the boundary in any way, our method can produce artifacts. This can happen at fine details, where the skeleton approximates the surface too roughly, as in the case of the fingers of a hand.

Design Space Limitations. Our solution space is limited by the degrees of freedom provided by the k manifold harmonics, thus any better optimum that lies outside of this space cannot be reached. Moreover, the usage of a constant shape skeleton as an upper bound for inner displacements additionally limits the design space, such that potentially valid solutions that require crossing of the medial axis cannot be reached. Finally, problems where multiple voids are necessary to achieve a good optimum are not well served with our algorithm. Space carving methods that perform global topology optimization are potentially more flexible for problems where multiple voids are needed, since their solution space is generally bigger and not limited to one surface.

7.4 Fabrication Considerations

Print Time and Support Material. We observed that smooth models need less support material and have lower print times than the voxelized models for FDM. This is basically due to the fact that voxel-bottoms that are parallel to the ground plane need full support, while smooth surfaces are self-bearing up to a certain degree. Moreover, the total distance traveled by the printer head is shorter on smooth curves than Manhattan-distance voxels. In a direct comparison with the work of Prévost et al. [2013] using the results shown in Figure 12, our approach cuts both the support material and the print time roughly by half. Further details can be found in the supplementary material.

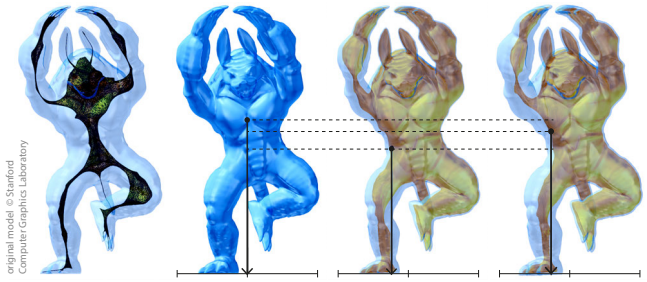

Figure 14: *Example of static stability optimization of a complex model. From left to right: meso-skeleton, outer shape with center of mass, inner surface only, inner and outer surface optimization. The inner surface result has a too thin wall to be printed. The outer surface offset solves this problem.*

Material Density Issues. In practice, we found that the material density values as presented by manufactures (e.g., $\rho_{\mathrm{ABSplus}} = 1.04$) do not necessarily apply to the printed models. This is due to the fact that FDM-fabricated solids have a large number of micro-holes integrated in the massive. These holes can be filled with air or water (especially after base-bath for support dissolution), changing the actual density of the mass. This problem has also been addressed by Christiansen et al. [2015]. We resolved it by coating wet models, which stabilized their density.

8 Conclusions

We have presented a novel method for shape optimization of an input object represented by a 2-manifold in order to attain given global specifications. The key idea was the utilization of offset surfaces, whose parameters are determined by continuous constrained non-linear optimization, such that the enclosed rigid body realizes a given objective. We ensured the computational feasibility of our method by reducing the order of the problem by solving it in a suitable lower-dimensional subspace, given by the manifold harmonic basis. Apart from increased numerical robustness due to an inherent regularization, we achieved a reduction of the computation times of at least of 2-3 orders of magnitude using common off-the-shelf numerical solvers in our implementation. We documented the versatility of our approach by optimizing a range of physical properties of rigid objects, and we provided a comparison with previous methods for shape optimization. In the future, we intend to integrate our technique in popular modeling software tools to assist users in the creation of fabrication-ready customized models.

Acknowledgments

We would like to acknowledge Christian Hafner for help with the renderer, Paul Guerrero for help with MATLAB, Moritz Bächer for sharing the models and providing the permission to use them, Sebastien Loriot for sharing his code for the computation of the skeletons, and Andrea Tagliasacchi for pointing us there.

This research was funded by the Austrian Science Fund (grants: FWF P24600-N23, FWF P27972-N31, FWF P23700-N23), the European Research Council (ERC Advanced Grant "ACROSS", grant

agreement 340884), and the German Research Foundation (DFG, Gottfried-Wilhelm-Leibniz Programm).

References

BÄCHER, M., BICKEL, B., JAMES, D. L., AND PFISTER, H. 2012. Fabricating articulated characters from skinned meshes. *ACM Transactions on Graphics 31*, 4 (July), 1–9.

BÄCHER, M., WHITING, E., BICKEL, B., AND SORKINE-HORNUNG, O. 2014. Spin-It: Optimizing Moment of Inertia for Spinnable Objects. *ACM Transactions on Graphics 33*, 4 (July), 1–10.

BICKEL, B., BÄCHER, M., OTADUY, M. A., LEE, H. R., PFISTER, H., GROSS, M., AND MATUSIK, W. 2010. Design and fabrication of materials with desired deformation behavior. *ACM Transactions on Graphics 29*, 4 (July), 1.

BICKEL, B., GROSS, M., KAUFMANN, P., SKOURAS, M., THOMASZEWSKI, B., BRADLEY, D., BEELER, T., JACKSON, P., MARSCHNER, S., AND MATUSIK, W. 2012. Physical face cloning. *ACM Transactions on Graphics 31*, 4 (July), 1–10.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics 23*, 3 (Aug.), 630.

CHRISTIANSEN, A. N., SCHMIDT, R., AND BÆ RENTZEN, J. A. 2015. Automatic balancing of 3D models. *Computer-Aided Design 58* (Jan.), 236–241.

COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. 2013. Computational design of mechanical characters. *ACM Transactions on Graphics 32*, 4 (July), 1.

GOLDSTEIN, H., POOLE, C. P., AND SAFKO, J. L. 2002. *Classical Mechanics*. Addison Wesley.

HAFTKA, R. T., AND GÜRDAL, Z. 1992. *Elements of Structural Optimization*, 3rd rev. a ed. Springer.

JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics 30*, 4 (July), 1.

JACOBSON, A., PANOZZO, D., AND OTHERS, 2014. {libigl}: A simple {C++} geometry processing library.

KIM, T., AND JAMES, D. L. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Transactions on Graphics 28*, 5 (Dec.), 1.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings ACM SIGGRAPH '98* (July), 105–114.

LEHOUCQ, R. B., SORENSEN, D. C., AND YANG, C. 1998. *ARPACK Users' Guide*, band 6 of ed. Society for Industrial and Applied Mathematics (SIAM), Jan.

LEVY, B. 2006. Laplace-Beltrami Eigenfunctions: Towards an Algorithm That "Understands" Geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, IEEE, 13–13.

LU, L., CHEN, B., SHARF, A., ZHAO, H., WEI, Y., FAN, Q., CHEN, X., SAVOYE, Y., TU, C., AND COHEN-OR, D. 2014. Build-to-Last: Strength to Weight 3D Printed Objects. *ACM Transactions on Graphics 33*, 4 (July), 1–10.

LUO, L., BARAN, I., RUSINKIEWICZ, S., AND MATUSIK, W. 2012. Chopper: partitioning models into 3D-printable parts. *ACM Transactions on Graphics 31*, 6 (Nov.), 1.

MESSNER, A. M., AND TAYLOR, G. Q. 1980. Algorithm 550: Solid Polyhedron Measures [Z]. *ACM Transactions on Mathematical Software 6*, 1 (Mar.), 121–130.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: model dynamics for graphics and animation. *ACM SIGGRAPH Computer Graphics 23*, 3 (July), 207–214.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1, 15–36.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make It Stand: Balancing Shapes for 3D Fabrication. *ACM Transactions on Graphics 32*, 4 (July), 1.

SKOURAS, M., THOMASZEWSKI, B., BICKEL, B., AND GROSS, M. 2012. Computational Design of Rubber Balloons. *Computer Graphics Forum 31*, 2pt4 (May), 835–844.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: improving structural strength of 3D printable objects. *ACM Transactions on Graphics 31*, 4 (July), 1–11.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Transactions on Graphics 26*, 3 (July), 80.

TAGLIASACCHI, A., ALHASHIM, I., OLSON, M., AND ZHANG, H. 2012. Mean Curvature Skeletons. *Computer Graphics Forum 31*, 5 (Aug.), 1735–1744.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95*, ACM Press, New York, New York, USA, 351–358.

THOMASZEWSKI, B., COROS, S., GAUGE, D., MEGARO, V., GRINSPUN, E., AND GROSS, M. 2014. Computational design of linkage-based characters. *ACM Transactions on Graphics 33*, 4 (July), 1–9.

TIMMER, H., AND STERN, J. 1980. Computation of global geometric properties of solid objects. *Computer-Aided Design 12*, 6 (Nov.), 301–304.

UMETANI, N., KAUFMAN, D. M., IGARASHI, T., AND GRINSPUN, E. 2011. Sensitive couture for interactive garment modeling and editing. *ACM Transactions on Graphics 30*, 4 (July), 1.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics 31*, 4 (July), 1–11.

VALLET, B., AND LÉVY, B. 2008. Spectral Geometry Processing with Manifold Harmonics. *Computer Graphics Forum 27*, 2 (Apr.), 251–260.

VON TYCOWICZ, C., SCHULZ, C., SEIDEL, H.-P., AND HILDEBRANDT, K. 2013. An efficient construction of reduced deformable objects. *ACM Transactions on Graphics 32*, 6 (Nov.), 1–10.

WANG, W., WANG, T. Y., YANG, Z., LIU, L., TONG, X., TONG, W., DENG, J., CHEN, F., AND LIU, X. 2013. Cost-effective printing of 3D objects with skin-frame structures. *ACM Transactions on Graphics 32*, 6 (Nov.), 1–10.

ZHANG, H., VAN KAICK, O., AND DYER, R. 2010. Spectral Mesh Processing. *Computer Graphics Forum 29*, 6 (Sept.), 1865–1894.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case structural analysis. *ACM Transactions on Graphics 32*, 4 (July), 1.

ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. 2012. Motion-guided mechanical toy modeling. *ACM Transactions on Graphics 31*, 6 (Nov.), 1.