A System for High-Resolution Topology Optimization

Jun Wu, Christian Dick, and Rüdiger Westermann

Abstract—A key requirement in 3D fabrication is to generate objects with individual exterior shapes and their interior being optimized to application-specific force constraints and low material consumption. Accomplishing this task is challenging on desktop computers, due to the extreme model resolutions that are required to accurately predict the physical shape properties, requiring memory and computational capacities going beyond what is currently available. Moreover, fabrication-specific constraints need to be considered to enable printability. To address these challenges, we present a scalable system for generating 3D objects using topology optimization, which allows to efficiently evolve the topology of high-resolution solids towards printable and light-weight-high-resistance structures. To achieve this, the system is equipped with a high-performance GPU solver which can efficiently handle models comprising several millions of elements. A minimum thickness constraint is built into the optimization process to automatically enforce printability of the resulting shapes. We further shed light on the question how to incorporate geometric shape constraints, such as symmetry and pattern repetition, in the optimization process. We analyze the performance of the system and demonstrate its potential by a variety of different shapes such as interior structures within closed surfaces, exposed support structures, and surface models.

Index Terms—Topology optimization, 3D printing, finite element analysis, multigrid.

1 Introduction

With the advent of the 3D printing era, optimizing shapes so that their printed replicas can resist specific external forces in their practical use becomes increasingly relevant. Shape optimization originates from the requirement in engineering to manufacture shapes which meet the structural needs of a part while reducing manufacturing or operating costs associated with volume or weight [1]. With the availability of high-resolution low-cost 3D printing devices, even consumer products like candlesticks, cloth hangers, racks, multi-leg pots or amulets will be manufactured using individual exterior shapes with optimized interior, respecting arbitrary and often non-local force distributions due to touch, load, or contact. Some items with a design supporting specific force conditions are shown in Fig. 1.

The most general approach to perform shape optimization is topology optimization [2]. It iteratively simulates the removal and redistribution of material from a part such that a given volume reduction is achieved and mechanical interior loads due to prescribed external forces are minimized. In topology optimization the interior loads are commonly measured by the object's *global compliance*, yielding a topology that minimizes the deformation of the structure when the external forces are applied. Topology optimization doesn't impose constraints on the resulting shapes—apart from optimizing the total internal strain energy—yet it requires to recompute a strain response in every iteration of the optimization. Because high-resolution models must be used to guarantee stable solutions this causes extremely long optimization cycles of several hours, strongly limiting the use of topology optimization for 3D model generation and printing [3].

In computer graphics shape optimization has also been performed by assembling the shape from predefined structures such as trusses [4], deformable solid blocks [5], skin-frames [6], or honeycomb-like Voronoi cells [7]. It has been demonstrated by Wang et al. [6] and Lu et al. [7] that the layout and geometry of such composites can be optimized with respect to the occurring *local stresses*, so that under applied forces a prescribed threshold at which the object would break is not exceeded. These methods can achieve expressive interior shapes, yet since they limit the space of possible shapes they have to release volume constraints for arbitrary external force settings. Notably, shape optimization under the constraint of maximum stress yields an optimization problem for which a general solution using volumetric element-wise parameterizations is still considered a challenge [8], [9].

1.1 Contribution

We present the design and realization of a scalable computer system for shape optimization. In response to the mentioned properties of existing approaches, our first design decision was to use topology optimization as the underlying methodology and, thus, to keep the constraints on the resulting shapes as low as possible. Yet to be of practical use, this decision required a new computational approach to enable the efficient numerical simulation of the interior strain distribution of extremely high-resolution models, such as $621 \times 488 \times 1000$ voxel models processed in this work.

Since numerical strain solvers for large problems can converge slowly and are memory bound, our second de-

• *Jun Wu, Christian Dick, and Rüdiger Westermann are with the Computer Graphics and Visualization Group, Technische Universität München, Munich, Germany.*
E-mail: jun.wu@tum.de, dick@tum.de, westermann@tum.de.

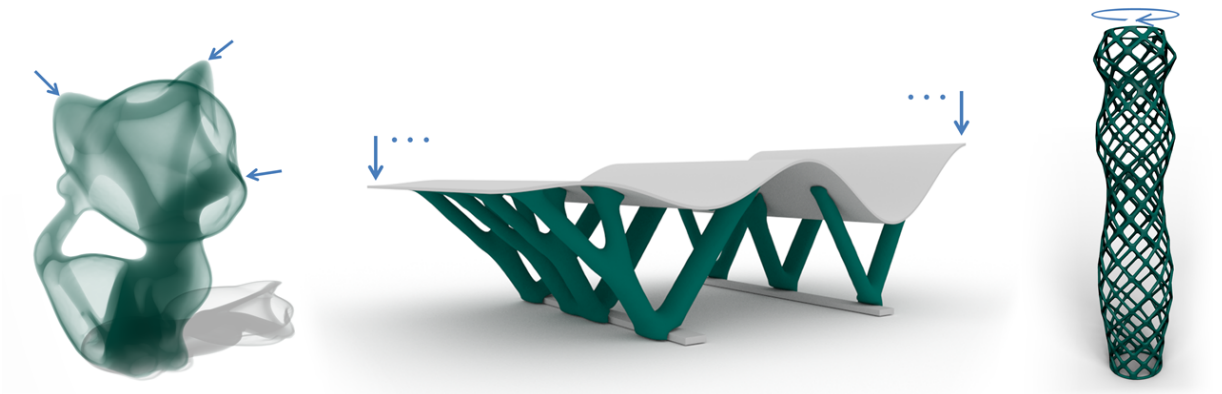


Fig. 1: Shapes generated by optimizing mechanical properties with respect to prescribed loads (blue arrows). The optimization process is applied to interior structures within closed surfaces (left), exposed support structures (middle), and surface models (right). The examples are obtained from an initial solid design domain, targeting a prescribed volume reduction and highest possible stiffness.

sign decision was to develop a high-performance multigrid solver with deep integration of GPU computing to achieve good convergence, and to restructure this solver towards low memory consumption and reduced bandwidth requirements. Due to these modifications, the solver can compute the compliance of models comprising several millions of elements in a few minutes on a desktop computer. Until today this performance could only be achieved on high performance computer systems [10].

Our third design decision was to ensure the printability of the shapes. To account for this (the printed versions of some of the shapes we discuss in this work are shown in Fig. 2), we have integrated an additional fabrication-specific constraint into the optimization process, i.e., the minimum thickness of the generated structures. This constraint is respected simultaneously to the minimization of compliance in the material update procedure in every iteration of the optimization process.

In addition to the fabrication-specific and structural optimization constraints required to generate printable and light-weight-high-resistance structures, our fourth design decision was to integrate additional geometric constraints enforcing pattern repetition and symmetry into the optimization process. Underlying our decision is the interest to explore the feasibility of control over the emerging shapes. Geometric constraints, in general, counteract the shape evolution towards mechanically optimized structures and tend to result in less resistant shapes, yet we believe it is nevertheless interesting to explore which compromises can be achieved.

By using computational efficiency in combination with additional constraints on the optimized shapes, our system allows for the generation of printable and mechanically optimized shapes at high resolution in a short period of time. We demonstrate this for a variety of shapes such as interior structures within closed surfaces, exposed support structures, and surface models.

2 Related Work

Modeling 3D shapes that can be printed and whose printed replicas satisfy certain requirements arising from their real-world use is now a central research area in computer graphics. For a survey of some of the techniques falling into this area let us refer to the work by Bickel and Alexa [11], the Siggraph Asia course on 3D printing oriented design by Liu et al. [12], the Siggraph course on computational tools for 3D printing by Umetani et al. [13], and the Dagstuhl report on computational fabrication by Alexa and co-workers [14].

Stava et al. [3] proposed a system to automatically detect and correct possible structural deficiencies of 3D printable objects. Their work is related to ours, because they make use of a finite-element-based elasticity analysis to compute a stress field for a given shape under predicted loads. Following the stress analysis is a set of correction operations like hollowing, thickening, and strut insertions to prepare the object for fabrication. Our research is partly motivated by their insight that post-analysis and -correction could be alleviated by embedding automatic shape optimization techniques into the construction process. We also draw inspiration from the observation that extremely high model resolutions are required to accurately represent the detail that can be produced by modern printing devices, raising the computational requirements so high that the practical use of shape optimization becomes very cumbersome.

One possible approach to generate optimized shapes is topology optimization [2]. It iteratively removes and redistributes material from a part while obtaining minimum compliance and keeping it lightweight. For an introduction to the basic principles underlying topology optimization and the shapes it generates let us refer to the TopOpt App by Aage et al. [15]. Previous and current developments in the field are reviewed in [9], [8]. In particular the density-based solid isotropic material with penalization approach (SIMP) [16] and the (bidirectional) evolutionary structural optimization approach (ESO) [17], [18] have widespread use in engineering. Both methods employ an element-wise parametrization over a regular voxel domain, enabling efficient material simulation and update routines. We have integrated both approaches into our system to exploit their individual characteristics, i.e., superior convergence of SIMP and stable handling of distributed external forces of BESO. A different approach to topology optimization uses level-set parameterizations to simulate the evolution of the structure

Fig. 2: The printed replicas of some objects that were generated by our system.

boundary by transport of a level set function [19], [20]. This approach is flexible in handling arbitrary domains and smooth material boundaries, yet it requires to connect the evolution of the material function with the boundary of the topology contours.

To tackle the computational complexity of topology optimization, a multi-scale embedding of high-resolution design domains into coarse finite element domains was suggested in [21]. Furthermore, a number of parallelization strategies using high-performance GPU and CPU architectures been proposed [22], [10], [23]. Despite these efforts, however, it was recently observed that the performance and user friendliness of topology optimization need to be improved considerably for graphics applications [24]. This also reflects in the most recent simulation results, where high-resolution models ($1008 \times 336 \times 336$) were demonstrated on a 1800-core computer, taking 4.5 hours [10]. Our approach, in comparison, requires 12 minutes on a single GPU for a $1000 \times 621 \times 488$ model with 14.2 million finite elements, based on non-trivial optimizations of the numerical simulation process.

Another approach to shape optimization is to use an objective function so that local material stresses do not exceed a given threshold. Currently, however, it is still considered a challenge to directly optimize the maximum stress in the general volumetric element-wise parameterization. To obtain approximate solutions, a number of techniques resort to predefined support structures and try to assemble such structures in a way that the occurring local stresses do not exceed the given limit. Such techniques, in general, limit the space of possible shapes and make it difficult to generate a layout that is optimal with respect to given physical constraints. In early work by Smith et al. [4], the mass and layout of truss structures is optimized for designing bridges and towers. Whiting et al. [5] analyzed the optimal arrangement and shape deformation of solid soft blocks for designing special families of buildings. Wang et al. [6] used skin-frame structures to reduce the interior volume of a given shape while maintaining its visual appearance. Given an input model, an initial frame representation consisting of nodes connected by struts is generated, and this configuration is successively optimized to reduce the radii and number of struts. Lu et al. [7] introduced honeycomb-like Voronoi structures to hollow the interior volume of solid objects while maintaining a high strength-to-weight ratio. From an initial distribution of Voronoi cells that is guided by a stress analysis, an adaptive centroidal Voronoi diagram is computed to partition the interior volume.

In the context of 3D printing, Zhou et al. [25] have used structural analysis and optimization to determine the ”worst” possible load distribution that will cause high local stresses in the object. Chen et al [26] addressed the design of shapes that deform to a desired pose under a given force distribution. Skouras et al. [27] introduced a method for designing actuated deformable characters by optimizing their internal material composition. Umetani and Schmidt [28] optimized the orientation of the 3D printed shapes to further increase stability. Panetta et al. [29] and Schumacher et al. [30] optimized microstructures for desired elastic behaviors. Pérez et al. [31] optimized rod meshes to match prescribed poses under specific boundary conditions. Dumas et al. [32] optimized shapes resulting from texture synthesis by using stress analysis and including additional support structures to release high stresses potentially causing fracture. Hu et al. [33] proposed to optimize surface meshes by deformation to create self-supporting shapes with improved stability. Internal material layout has been investigated for the design of self-balancing shapes [34] and spinnable objects [35]. Musialski et al. [36] proposed to control mass properties by optimizing the offset surface. Physics-based analysis and optimization has also been studied in computer graphics for shape design in the context of other types of computational fabrication [37], [38], [39].

3 System Overview

The basic operations performed by our system during topology optimization, as well as the intermediate shapes it generates, are illustrated in Fig. 3.

Our system takes as input any closed polygonal surface. Via an interactive interface, the user can specify the concrete boundary conditions to be considered in the optimization process, i.e., the location, direction, and strength of the external forces acting on the object. Therefore, the user picks a surface point, orients a vector glyph according to the direction from which the force should be acting on this point, and specifies a fall-off radius indicating how the point force is spread (via a Gaussian kernel) across the surface. Then, the user specifies the force strength in

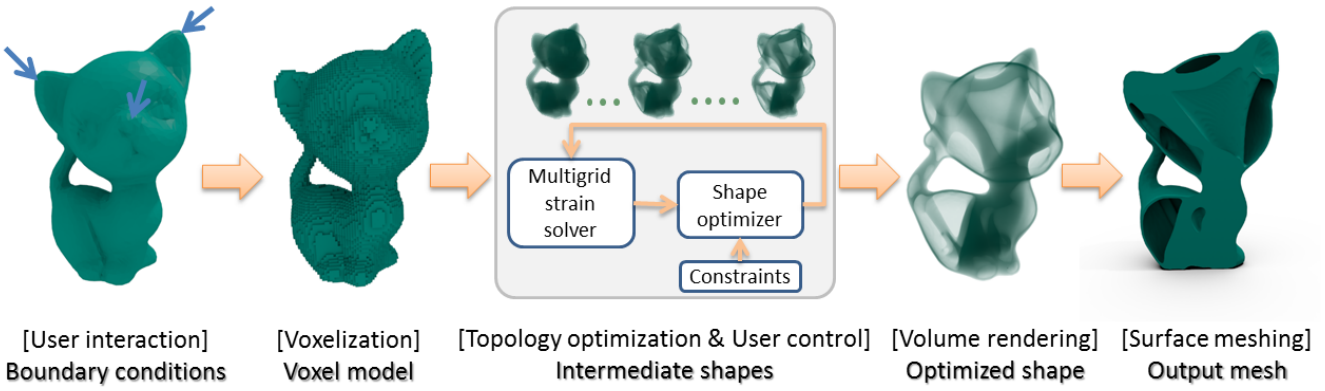


Fig. 3: Overview of the basic system functions and the intermediate shapes the system generates.

units of Newton, and a global force field is reconstructed by a linear interpolation of all local force contributions. The user also specifies the locations where the object is fixed, by selecting the respective surface vertices. Some typical boundary conditions are automatically generated, for instance, a uniformly distributed force field acting on each surface patch along its inward-pointing normal, and the fixation of the bottom zone of an object.

Then, the system computes a conservative surface voxelization—with a user-selected resolution—to determine all voxels lying in the interior of the surface. Voxelization is performed on the GPU via the CUDA parallel programming API using the method proposed by Schwarz et al [40], [41]. The voxelized model is employed as a discretization of the elasticity problem. Following previous works [3], [7], we consider a linear elastic model and assume the material is isotropic and homogeneous. To this end, all voxels are initialized with the same Young's modulus and Poisson's ratio.

The voxel model is then optimized using topology optimization as described in Sec. 4. The objective is to find an optimized shape such that the mechanical stiffness is as high as possible, under the constraint of a given volume budget specified by the user. In each iteration of the optimization procedure, a linear elasticity solver (see Sec. 5) is used to compute the internal stress distribution. Using this distribution, material update, i.e., changing voxels from solid to void and vice versa, is performed so that the total strain is reduced. A sequence of shapes generated during topology optimization is shown in Fig. 3. Additional control rules can be used to balance between mechanical soundness and printability on the one hand, and expressiveness on the other hand.

Once the prescribed volume loss is achieved and the optimization has been converged, the final shape—represented by a voxel model—is output. The voxel model can either be viewed via direct volume rendering to inspect the interior structures, or the boundary surface between solid and void voxels is extracted via the Marching Cube algorithm [42] and further smoothed via Taubin smoothing [43] to reduce staircase artifacts resulting from the uniform voxel structure.

In the following, we describe and discuss each system operation in detail, and we show in a number of examples the results they produce.

4 Topology Optimization

4.1 Problem Formulation

Topology optimization is based on a discrete formulation of an elastic model to minimize the compliance of a shape under the constraint of an exerting force and a prescribed material consumption [16]. The incorporation of additional fabrication- and geometry-specific constraints will be discussed in Sec. 6.

Given the domain Ω that is covered by an initial part, the material distribution is represented by a real-valued design variable $\rho(x) \in [0,1]$, $x \in \Omega$, representing material points continuously varying between solid $(\rho(x) = 1)$ and void $(\rho(x) = 0)$. The object's compliance $c(\rho)$ is measured in terms of the internal strain energy, by summing the energy over all material points.

To formulate the compliance minimization problem in terms of a discrete set of design variables ρ_e, we use a hexahedral finite element discretization of a linear elastic material on a uniform Cartesian grid:

$$\underset{\rho}{\text{minimize}} \quad c(\rho) = \frac{1}{2} u^T K(\rho) u, \tag{1}$$

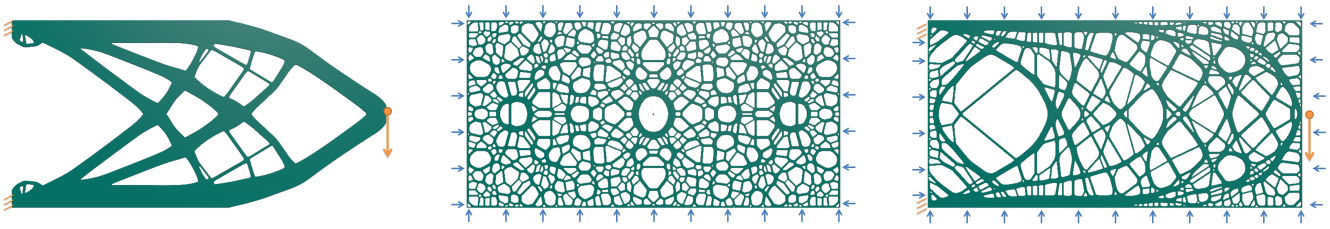
$$\text{subject to} \quad K(\rho)u = f, \tag{2}$$

$$V(\rho) = \sum_e \rho_e \leq V^*, \tag{3}$$

$$\rho_e \in [0,1], \forall e. \tag{4}$$

We assemble the stiffness matrix K from element stiffness matrices $K^e = \int_{\Omega_e} B^T D B \; dx$, where Ω_e is the domain of the finite element, B is the element strain matrix, and D is the linear material law. The displacement vector u is calculated from the equilibrium equation Eq. 2, which describes the static state of the object under the given external force f. The volume constraint Eq. 3 restricts the material consumption to a desired threshold V^*.

To avoid singularities of the global stiffness matrix K, Eq. 4 is relaxed by prescribing a non-zero constant ρ_{min} (e.g., $\rho_{min} = 0.001$) to represent void material. Material properties (i.e., Young's modulus) are expressed using the power-law relationship, i.e., $E_e = (\rho_e)^p E_0$, where p is a penalization parameter (typically $p = 3$), and E_0 is the Young's modulus of the solid $(\rho_e = 1)$ material. This formulation is known as the Solid Isotropic Material with Penalization (SIMP) model [16], and the design variable ρ is analogously viewed as material density.



(a) Concentrated force (b) Distributed forces (c) Concentrated force + distributed forces

Fig. 4: (a) Optimized structure obtained via topology optimization, i.e., minimizing the compliance of the structure under the prescribed boundary forces while reducing the volume of the initial solid by 60%. (b) Shape optimization considering touch/grasp interaction at arbitrary locations via uniformly distributed forces. (c) Optimization with respect to concentrated work-condition forces and considering arbitrary touch/grasp interaction. In all images, orange and blue vector icons indicate concentrated and uniformly distributed forces, respectively, while the orange lines indicate the fixation regions.

4.2 Optimization for Printed Objects

Depending on the use case, external forces from different sources and with different distributions will be exerting on the printed object. As illustrated in Fig. 4, our system considers two different types of forces: *Concentrated* and *distributed* forces. Concentrated forces are typically considered in engineering applications to design shapes which are resistant to permanent or recurring concentrated loads. Such forces are applied to a local area on the object's surface in its work condition, e.g., forces applied to cloth hangers and brackets. Distributed forces are applied during object fabrication and transportation, as well as user interaction with the object in its real-world use. For instance, the user may touch or grasp the object at an arbitrary location. Distributed forces are assumed to act uniformly on the surface along the inward-pointing normal.

Given the external forces, the solution of the topology optimization problem is obtained iteratively. In each iteration the following computational steps are performed:

1) Solving the elasticity equation Eq. 2,
2) objective and sensitivity analysis, and
3) design update.

In the following we will briefly review steps (2) and (3), before we present the new computational approach for solving the linear system of equations in step (1).

4.2.1 Objective and sensitivity analysis

Once the displacement vector u for the current shape is computed via a finite element analysis in step (1), the system evaluates the objective function Eq. 1 under the constraint Eq. 3. Therefore, the derivatives of the total strain energy c and the total volume V with respect to the design variable ρ_e are computed as

$$\frac{\partial c}{\partial \rho_e} = -\frac{p}{2}(\rho_e)^{p-1} u_e^T K^0 u_e \tag{5}$$

and

$$\frac{\partial V}{\partial \rho_e} = 1. \tag{6}$$

To suggest the density updates in the following design update procedure, the impact of the allocated per-element density on the change of the strain energy needs to be quantified. It is common to measure this impact via the sensitivity of the strain energy to the change of an element's volume as

$$G_e = \frac{-\partial c/\partial \rho_e}{\partial V/\partial \rho_e}. \tag{7}$$

A large per-element value indicates that the same amount of volume (represented by the density) allocated at this element has a high impact on the change of the strain energy.

4.2.2 Design update

To determine the density updates in every iteration, the effect of these updates is estimated from the elements' sensitivities. In particular, if an element's sensitivity is relatively small, its density can safely be reduced because this does not increase the total strain energy significantly. On the other hand, if the sensitivity is relatively large, a density increase should be favored because this decreases the total strain energy remarkably. This leads to the SIMP updating scheme for the intermediate values:

$$\rho_e^{new} = \mathsf{Clamp}(\mathsf{Clamp}((G_e/\bar{G})^{\eta}\rho, \rho-\Delta\rho, \rho+\Delta\rho), \rho_{min}, 1.0), \tag{8}$$

where $\Delta\rho$ is a positive move-limit (e.g., $\Delta\rho = 0.2$), η is a numerical damping coefficient (typically $\eta = \frac{1}{2}$), and $\bar{G}$ is the sensitivity threshold such that after updating all densities the volume constraint at the current iteration is satisfied, i.e., $V(\rho) < V^i$, where V^i denotes the desired volume at the i-th iteration (see below). $\bar{G}$ is found via recursive bi-section of the sensitivity interval until the selected value leads to the prescribed volume decrease.

For optimizing shapes under concentrated forces, the continuous SIMP updating scheme has good convergence and, via the penalization parameter p, the densities can be steered towards a binary design, as shown in Fig. 5 (left). However, when distributed forces are simulated (Fig. 5 (right)), our experiments show that the continuous updating scheme does not lead to a stable binary solid-void design, since the material sensitivities are very similar throughout the design domain. Thus, we constrain the design update to enforce a binary design via the BESO updating scheme:

$$\rho_e^{new} = \begin{cases} \rho_{\min} & \text{if } G_e < \bar{G}, \\ 1.0 & \text{otherwise.} \end{cases} \tag{9}$$

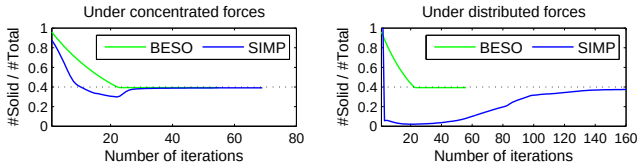


Fig. 5: The ratio of the number of solid elements over the number of total elements in the optimization process. Under concentrated forces (left) SIMP converges to a binary design, while under distributed forces (right) SIMP requires a significantly larger number of iterations. The prescribed ratio is indicated by the dashed line.

To improve the convergence of the discrete evolution process, a stabilization is performed by averaging the sensitivity values with the values in the previous iteration [18].

In practice, starting from a solid object (i.e., $\rho_e = 1, \forall e$), the volume is gradually decreased by a small amount in every iteration, i.e.,

$$V^i = \max((1.0 - \alpha)V^{i-1}, V^*), \tag{10}$$

where α is the volume reduction rate. The effect of the volume reduction rate on the rate of change of the volume and the compliance is shown in Fig. 6. It can be observed that the optimization process has two phases: First, the volume (top) is gradually reduced to a prescribed volume, with the speed depending on α. In this phase, as the volume is reducing, the compliance (bottom) increases. Once the target volume is reached, it doesn't change any further, but material is redistributed to find the lowest possible compliance. This second phase is possible because both updating schemes Eq. 8 and Eq. 9 are bi-directional and allow existing structures to be replaced. With a smaller α, in the first phase the volume decreases slower and the overshoot above the final compliance is smaller. In our examples we choose α between $1\% \sim 5\%$, and the optimization process was terminated if a) the targeted volume was reached and b) the change in compliance compared to the last iteration was below 0.5%, as indicated by a '+' sign in the curves.

Fig. 4 shows some 2D results of topology optimization using the described process with concentrated and distributed force constraints, and using both simultaneously. The structures were generated by prescribing different force fields on the outer object boundary, and fixing the material at certain locations. 3D examples are shown in Fig. 15.

5 A Memory-Efficient Multigrid Solver

The performance bottleneck in topology optimization is solving the static equilibrium equation (Eq. 2) to obtain the strain energy. It involves assembling and solving a large, sparse linear system of equations $Au = b$, which takes up to 85% of the processing time. Current topology optimization approaches mainly resort to conjugate gradient (CG) schemes to solve this system, yet it is well known that especially for high model resolutions the convergence of CG solvers can be very slow.

To achieve improved convergence rates when simulating linear elastic material, in a number of previous works the computational efficiency of geometric multigrid solvers was

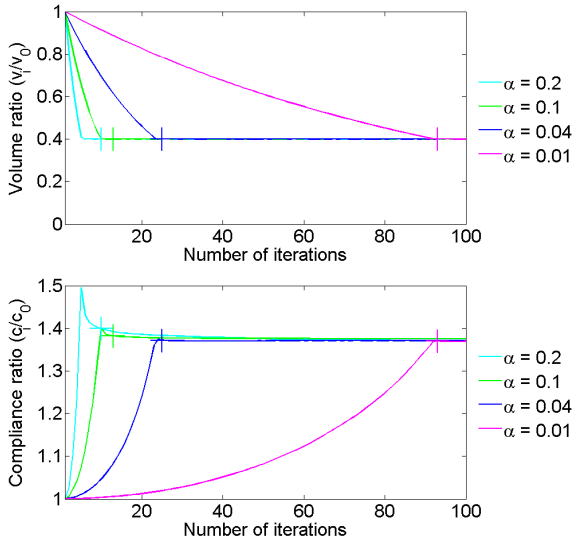


Fig. 6: The volume (top) and the compliance (bottom) behavior in the optimization process.

exploited [44], [45], [46]. For a thorough introduction to the basic principles underlying multigrid methods in elasticity simulations let us refer to the tutorial by Briggs [47]. In our work we use the geometric multigrid solver as preconditioner for a conjugate gradient solver. Geometric multigrid methods combine a hierarchy of coarser grids and relaxation schemes (smoothers) to restrict a fine-grid residual (which remains after smoothing) to the next coarser grid and computing a correction term on this grid (which is then interpolated to the fine grid) by applying this principle recursively. This leads to a basic grid traversal scheme referred to as V-cycle, which is outlined in Algorithm 1. Here, $A^\ell x^\ell = b^\ell$ denotes the respective linear system on level ℓ, with level number $\ell = 0$ corresponding to the finest level. r^ℓ denotes the residual on level ℓ, and $R_\ell^{\ell+1}$ and $I_{\ell+1}^\ell$ denote the restriction and interpolation operator between levels ℓ and $\ell + 1$.

Algorithm 1 V-Cycle

1: **for** $\ell = 0, \ldots, L-1$ **do** ▷ Go down in the V-cycle
2: **if** $\ell > 0$ **then** $x^\ell \leftarrow 0$ **end if**
3: Relax $A^\ell x^\ell \approx b^\ell$ ▷ Apply smoother
4: $r^\ell \leftarrow b^\ell - A^\ell x^\ell$ ▷ Compute residual
5: $b^{\ell+1} \leftarrow R_\ell^{\ell+1} r^\ell$ ▷ Restrict residual
6: **end for**
7: Solve $A^L x^L = b^L$ directly ▷ Solve on coarsest level
8: **for** $\ell = L-1, \ldots, 0$ **do** ▷ Go up in the V-cycle
9: $x^\ell \leftarrow x^\ell + I_{\ell+1}^\ell x^{\ell+1}$ ▷ Interpolate error & correct
10: Relax $A^\ell x^\ell \approx b^\ell$ ▷ Apply smoother
11: **end for**

Linear elasticity multigrid solvers for large problems are memory bound, meaning that they operate close to the theoretical memory bandwidth and further performance increases are difficult to achieve. To address this limitation, Dick et al. [48] proposed a GPU multigrid implementation which exploits the fast memory interface on such architectures. In combination with a dedicated parallelization

scheme and matrix-free data structures, significant performance improvements compared to a CPU implementation were reported.

However, regardless the simulation performance that can be achieved on the GPU, large model resolutions corresponding to tens of millions of finite elements cannot be handled on such systems due to the limited GPU memory capacity. To address this limitation, we build on the GPU multigrid implementation by Dick et al., yet we exploit the hexahedral finite element discretization in order to assemble the numerical stencils on-the-fly and to aggressively coarsen the geometric domain hierarchy that is used by the multigrid solver. This enables us to target model resolutions as large as $621 \times 488 \times 1000$ on a single GPU.

Starting from a uniform simulation grid, we construct a coarse grid hierarchy by doubling the cell size with each coarser level, and creating a cell on the next level iff it covers at least one cell on the previous level. We employ trilinear interpolation operators, the restriction operators are the transpose of the interpolation operators, and the coarse grid operators are obtained by Galerkin-based coarsening, i.e., the coarse grid operators are assembled automatically from the stencils on the finest grid using variational principles. We use V-cycles with one pre- and one post-smoothing 8-color Gauss-Seidel step. We further adopt the index-based FEM and multigrid formulation proposed by Dick et al., which enumerates and accesses finite elements and vertices via indices rather than storing them in a matrix structure. This formulation allows for an efficient GPU parallelization by spawning one CUDA thread for each finite element or grid vertex, as well as 3×3 threads for simultaneously computing per element matrix coefficients. For solving the linear equation system on the coarsest level, we found that utilizing a CPU solver—specifically, we employ the Cholesky solver from the TAUCS library [49]—is most efficient, despite of the introduced communication overhead between CPU and GPU. The reason is that the number of vertices on the coarsest level is too small to fully occupy the GPU.

The numerical stencil A^v at each vertex v requires 27 3×3 matrices, as well as 3-component vectors for the displacement u^v, the right-hand side b^v, and the residual r^v. Storing vertex-adjacency information explicitly (rather than implicitly using a rectangular-shaped grid with a potentially large number of void vertices) further requires storing for each vertex the indices of the 27 adjacent vertices of the stencil, the 27 fine-level vertices that restrict to the considered vertex, as well as the up to 8 coarse-level vertices the considered vertex interpolates from. Using double floating point precision and 32-bit integer indices, this leads in total to average storage requirements of 2.4 kB per finite element (2.3 GB per one million finite elements).

5.0.3 On-the-fly Assembly

Our first contribution to reduce the high memory requirements is the assembling of the numerical stencils on the finest level $\ell = 0$ during Gauss-Seidel relaxation and residual computation on-the-fly, from the 8 incident elements' matrices. Since the element matrices scale linearly with the Young's modulus value, only a single element matrix K^0 is required, which can be stored in fast on-chip memory. Since the number of GPU registers per thread is very limited, it is important to perform the computation in such a way that the use of temporary variables is minimized. We achieve this by performing the relaxation step on level 0 at each vertex v according to

$$S \leftarrow \sum_{k=1}^{8} \left(E_{e_k} \left(\sum_{\substack{m=1 \\ m \neq 8-k}}^{8} K^0_{[8-k,m]} u^{v_{k,m}} \right) \right), \tag{11}$$

$$M \leftarrow \sum_{k=1}^{8} E_{e_k} K^0_{[8-k,8-k]}, \tag{12}$$

and successively for $i \in \{1, 2, 3\}$:

$$u_i^v \leftarrow \frac{1}{M_{ii}} \left(b_i^v - S_i - \sum_{\substack{j=1 \\ j \neq i}}^{3} M_{ij} u_j^v \right). \tag{13}$$

Here, $K^0_{[ij]}$ denotes the (i, j)-th 3×3-block of the generic element matrix K^0, e_k ($k = 1, \ldots, 8$) are the eight incident elements of the considered vertex v, and $v_{k,m}$ ($m = 1, \ldots, 8$), are the 8 vertices of element e_k (each enumeration in Z-order, x first, z last). S and M are a temporary 3-component vector and a 3×3-matrix, respectively. Compared to fetching the assembled stencils from memory, assembling the stencils on-the-fly roughly doubles the number of FLOPs per vertex during relaxation, but is about a factor of two faster because it reduces the memory bandwidth requirements on the GPU.

For the coarser levels $\ell = 1, 2, \ldots$, however, assembling each vertex's stencil on-the-fly from $8^{\ell+1}$ finest-level matrices would be too costly and it is thus necessary to store these stencils in GPU memory. Unfortunately, since the stencils on level 1 still consume about $(27 \times 9 \times 8)/8 = 240$ byte with respect to each finite element on level 0, for very large model resolutions the memory requirements are still huge.

5.0.4 Non-dyadic Multigrid

To further reduce the memory requirements, we propose a slightly different coarsening strategy in the realization of a multigrid V-cycle. To understand the rationale underlying this approach, let us mention that the physical simulation is always performed using the elements on the finest multigrid level, yet the upper part of the multigrid hierarchy serves to improve the convergence of the numerical solution process.

A fact that has not been considered so far is that the particular Galerkin-based coarsening scheme we use is not restricted to a dyadic decimation of the grid resolution. Instead, the coarsening can be performed using other decimation factors as well. In particular, since this doesn't affect the finest level, still the same solution will be obtained, yet it might reduce the speed of convergence.

In our realization we thus adapt the multigrid operators so that the transfer is directly performed between the finest level (level 0), which is assembled on-the-fly, and level 2, where the stencils are stored. Thus the stencils on level 1 are not required. Conceptually, this means that we coarsen the finest level by 4:1, and the remaining levels by 2:1. As a consequence, 7^3 level-0 vertices restrict to each level-2 vertex, and each level-0 vertex interpolates from up to 8

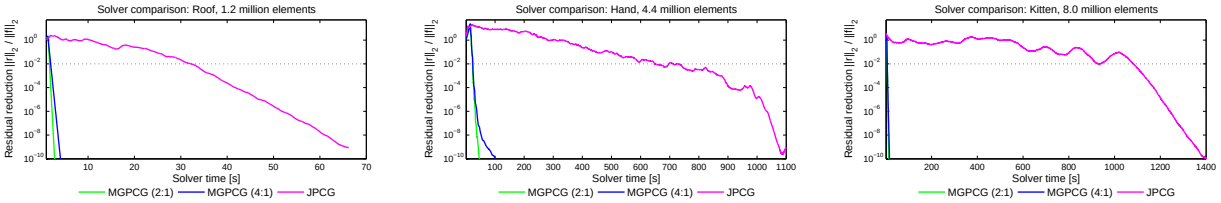


Fig. 7: Comparison of the convergence behavior (computing time vs. norm of residual relative to the norm of the right-hand side) of a multigrid preconditioned conjugate gradient solver (MGPCG) using 4:1 and 2:1 coarsening on the finest level (in both cases 2:1 coarsening on the remaining levels), and a Jacobi preconditioned CG solver (JPCG). The diagrams correspond to the first topology optimization design cycle for the roof, hand, and kitten model consisting of 1.2, 4.4, and 8.0 million finite elements, respectively. In the topological optimization process, the solver is stopped when $\|r\|_2 \leq 10^{-2}\|f\|_2$, as indicated by the horizontal dashed line.

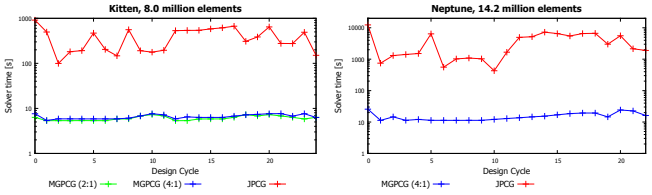


Fig. 8: Solver times required in each optimization cycle to reach a residual norm of $\|r\|_2 \leq 10^{-2}\|f\|_2$ for the kitten and Neptune model consisting of 8.0 and 14.2 million finite elements, respectively. Note that the scale of the vertical axis is logarithmic. For the Neptune model, a standard 2:1 coarsened multigrid hierarchy exceeds the capacity of the GPU memory.

level-2 vertices, using trilinear interpolation weights, and the same weights for restriction. In particular, the restriction from level ℓ to $\ell + \Delta\ell$ is given by

$$b^{v_c} \leftarrow \sum_{\substack{v_f \in \mathcal{V}^\ell \\ \|x^{v_f} - 2^{\Delta\ell} x^{v_c}\|_\infty < 2^{\Delta\ell}}} w^{v_f \leftrightarrow v_c} r^{v_f} \tag{14}$$

for each coarse-grid vertex $v_c \in \mathcal{V}^{\ell+\Delta\ell}$, where $\mathcal{V}^{\ell+\Delta\ell}$ denotes the set of vertices on level $\ell+\Delta\ell$, and the interpolation and coarse-grid correction from level $\ell + \Delta\ell$ to ℓ by

$$u^{v_f} \leftarrow u^{v_f} + \sum_{\substack{v^c \in \mathcal{V}^{\ell+\Delta\ell} \\ \|x^{v_f} - 2^{\Delta\ell} x^{v_c}\|_\infty < 2^{\Delta\ell}}} w^{v_f \leftrightarrow v_c} u^{v_c}, \tag{15}$$

for each fine-grid vertex $v_f \in \mathcal{V}^\ell$, using the trilinear interpolation weights

$$w^{v_f \leftrightarrow v_c} = \prod_{i=1}^{3} \frac{2^{\Delta\ell} - |x_i^{v_f} - 2^{\Delta\ell} x_i^{v_c}|}{2^{\Delta\ell}}. \tag{16}$$

It is $\Delta\ell = 2$ for the restriction and interpolation between levels 0 and 2, and $\Delta\ell = 1$ for the restriction and interpolation between levels ℓ and $\ell + 1$, $\ell \geq 2$. x^v denotes the 3D integer position of vertex v on the underlying lattice of the respective level. Note that $x^{v_f} = 2^{\Delta\ell} x^{v_c}$ for a vertex v_f on level ℓ and a vertex v_c on level $\ell + \Delta\ell$, when these vertices are located at the same spatial position.

As our experiments demonstrate (see Fig. 7, Fig. 8 and Table 1), the modification of the inter-grid transfer only slightly decreases the convergence rate, but it significantly reduces the memory requirements. In combination with the index-based model representation on the GPU, a drastic memory reduction can be achieved, enabling an accurate representation of high resolution models with fine geometric details.

6 Optimization Constraints

6.1 Minimum Thickness Control

Additive manufacturing requires a minimum thickness of the printed shapes. To incorporate this constraint into the optimization process, Guest [50] suggested to perform a thickness-dependent projection of the density field ρ. The projection smooths out the design variable, and it effectively prevents the occurrence of structures smaller than the prescribed minimum thickness $d_{\min}$. It is performed prior to the design update step, so that the update of each element takes into account the design variables of elements in its neighborhood. This is achieved by using an auxiliary design variable ϕ, from which the density ρ_e is computed as the weighted average of ϕ in close proximity as

$$\rho_e = \frac{\sum_{i \in S_e} \phi_i \omega(x_i, x_e)}{\sum_{i \in S_e} \omega(x_i, x_e)}. \tag{17}$$

Here, x_e is the centroid of the considered element, and S_e is the set of elements with the distance $\|x_i - x_e\|$ smaller than one half of the prescribed allowable thickness $d_{\min}$. As weight function we choose the compactly supported radial spline function [51]

$$\omega(x_i, x_e) = 1 - 6r^2 + 8r^3 - 3r^4, \tag{18}$$

with $r = \frac{2\|x_i - x_e\|}{d_{\min}}$.

The optimization problem with ϕ as the design variable is solved in a similar way to the original problem. In contrast, however, the derivative of the strain energy and the volume with respect to the new design variable need to be considered to obtain a solution to the objective function Eq. 1. The derivatives of both quantities with respect to ϕ are obtained by applying the chain rule, i.e.,

$$\frac{\partial c}{\partial \phi_i} = \sum_{e \in S_i} \frac{\partial c}{\partial \rho_e} \frac{\partial \rho_e}{\partial \phi_i}, \tag{19}$$

and

$$\frac{\partial V}{\partial \phi_i} = \sum_{e \in S_i} \frac{\partial V}{\partial \rho_e} \frac{\partial \rho_e}{\partial \phi_i}. \tag{20}$$

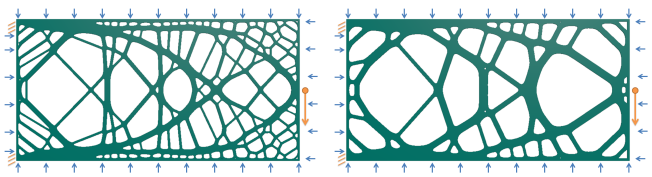


Fig. 9: Incorporating the minimum thickness constraint into the optimization process for the setting in Fig. 4 (c). A larger minimum thickness is enforced on the right.

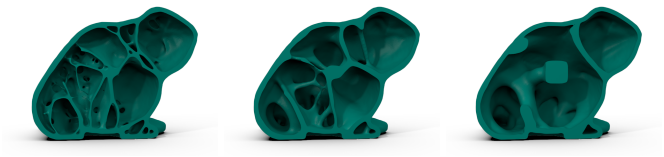


Fig. 10: From left to right: Interior shape variants with increasing minimum thickness. The same external forces are applied, with prescribed volume reduction of 60%.

Note that since the density ρ_e only depends on the values of the design variable ϕ in close proximity to a considered element, computing the derivatives requires only a small number of operations and can be performed very efficiently.

Fig. 9 shows the different shapes that occur when different thickness constraints are considered during topology optimization, i.e., the optimization process always tries to optimize the structures under consideration of the currently allowed thickness. As the allowed thickness increases, the resulting structures become thicker to better accommodate mechanical loads, and at the same time the topology is simplified towards a smaller number of cavities. The same effects are shown in Fig. 10 for a 3D shape.

6.2 Shape Regulation

Letting the topology of a part evolve solely with respect to equations Eq. 1 to 4 yields maximum resistance under the given constraints. On the other hand, there are other situations where one would like to trade the mechanical optimality of the generated shape for an improved appearance of this shape. In what follows we introduce additional geometric constraints to achieve this, yielding a further increase of flexibility in shape design. While such focus can reduce the mechanical optimality, it opens up new possibilities for shape design beyond classical shape optimization.

Symmetries and pattern repetition The beauty of many designs results from the regularity of the used patterns, symmetries, and repetitions. To enforce symmetries and repetitions, we make use of a domain partitioning strategy and a mapping from an imaginary design domain to the partitions. The key idea underlying our approach is to compute the elasticity problem on the initial domain, i.e., for all repeated patterns simultaneously, yet to perform the same sensitivity update for all corresponding elements in the sub-domains.

To create the same shape in different domain parts, the design domain Ω is first partitioned into a number of sub-domains Ω^i. The sub-domains need not to be of equal size, but it is only required that a parametrization exists to ensure a one-to-one mapping from an imaginary domain Ω^* to each

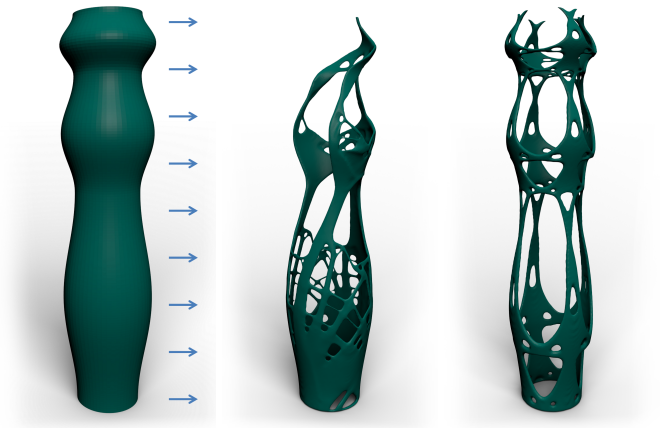


Fig. 11: Topology control with pattern repetition and gradation. Left: A curved cylinder shell serves as design domain. Middle: The optimized shape when geometric constraints are not considered. Right: The optimized shape when a constraint on vertical pattern gradation (3 repetitions of the same pattern are shown) and on rotational symmetry is enforced. Both shapes retain a volume ratio of 0.4.

Ω^i. The elasticity problem required in the shape optimization process is solved on the real design domain Ω, while the design update is performed in the imaginary domain. The derivative of the objective function with respect to the density values in the imaginary design domain is calculated for each element as

$$\frac{\partial c}{\partial \rho_e^*} = \frac{1}{N} \sum_i \frac{\partial c}{\partial \rho_e^i}, \tag{21}$$

where N is the number of sub-domains, and ρ_e^i is the density of the corresponding element in the i-th sub-domain. In this way, averaged densities from the sub-domains are used in the sensitivity update. After updating the densities in Ω^*, the densities in the sub-domains are updated accordingly, i.e., $\rho_e^i = \rho_e^*$. If the sub-domains have different sizes, the corresponding values are trilinearly interpolated, so that the designed shapes are scaled according to the size of the respective sub-domain. This way, the same binary material distribution is generated in all sub-domains, and the elasticity simulation works on a global field comprising multiple identical copies of the shape.

An example showing the effect of this process is shown in Fig. 11, where a highly asymmetric force is applied to the design domain. The domain is divided along the horizontal into three sub-domains of different sizes, and a rotational symmetric shape is enforced via the accumulation of sensitivities as described.

7 Results

In the following, we evaluate the performance of our system and provide timings for the most time-consuming operations. We further demonstrate the shapes that can be generated by our system, and we analyze the mechanical properties of some of them. Our experiments were run on a standard desktop PC equipped with an Intel Xeon

Fig. 12: The Neptune model was discretized on a $621 \times 488 \times 1000$ grid, with 14.19 million finite elements. Shape optimization took 12 minutes. The hand model was discretized on a $401 \times 250 \times 178$ grid, with 4.39 million finite elements. Shape optimization took 7 minutes.

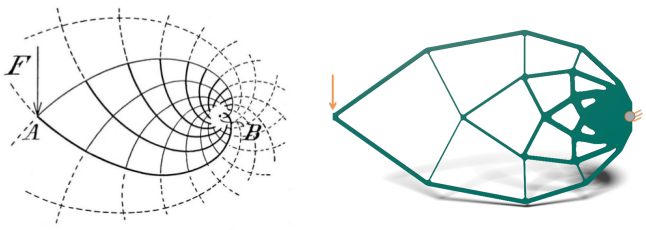


Fig. 13: Left: The principle of lightweight structures as illustrated by Michell [52]. Right: Shape optimization at a resolution of $800 \times 8 \times 500$.

X5560 processor running at 2.80 GHz, 8 GB of RAM, and an NVIDIA Tesla C2070 graphics card with 6 GB memory.

Examples We have used our system to optimize a variety of shapes including interior structures within closed surfaces, exposed support structures, and surface models. A minimum thickness of 6 voxels was used in all of our experiments. Fig. 2 shows some printed objects which were optimized by our method. The objects are printed with a consumer-grade printer (Ultimaker 2, which uses fused-deposition modeling).

Fig. 12 shows practical hangers with prescribed exterior shape, and an interior design that was optimized to resist the load of the keys. While the interior structures do not seem intuitive at first, from a structural mechanics point of view they are reasonable. For instance, it can be observed that the head of the Neptune statue and the upper part of the trident are completely empty, since the forces are not transmitted to these areas.

Fig. 13 (left) shows the Michell truss [52], a famous optimal shape design. It is derived analytically in the simple setting of a supporting point load with a distant base in the planar domain. At a high-resolution of $800 \times 8 \times 500$, shape optimization using the same setting generates a lightweight structure which matches extremely well with the analytical solution. It is worth noting here, that in our simulation we did not constrain the shape to be composed of trusses, yet the thin elongated structures along the principal stress directions evolved automatically from the optimization process.

We further used our system to optimize the shape of a complex vase-shaped model shown in Fig. 1. The optimization process was started with a curved cylinder shell with a thickness of 6 simulation elements, and a target volume reduction of 90%. The model was fixed at the bottom, and an external force exerting on the top surrounding was prescribed so that the shape started to twist. Due to the high volume reduction, a very sparse, truss-like pattern is obtained.

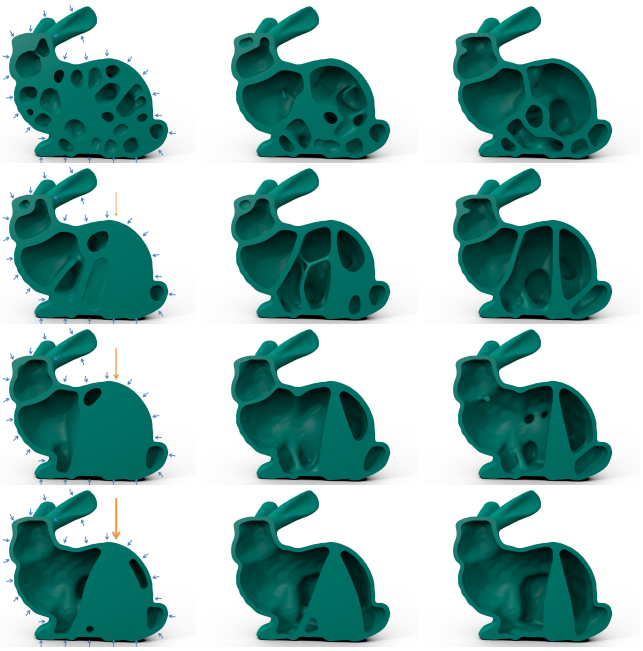


Fig. 14: Variants of optimized internal structures for the given exterior bunny shape under various external forces. From left to right, the volume is reduced by a factor of 40%, 50%, 60%, respectively. The forces applied on the back increase from top to bottom.

The first row of Fig. 14 shows the bunny model under distributed forces exerting on its boundary. From left to right, with increasing volume reduction, the number of cavities becomes smaller. The cavities automatically merge to accommodate different volume constraints. From top to down, we show the automatically optimized structures regarding an increasing concentrated force applied to the

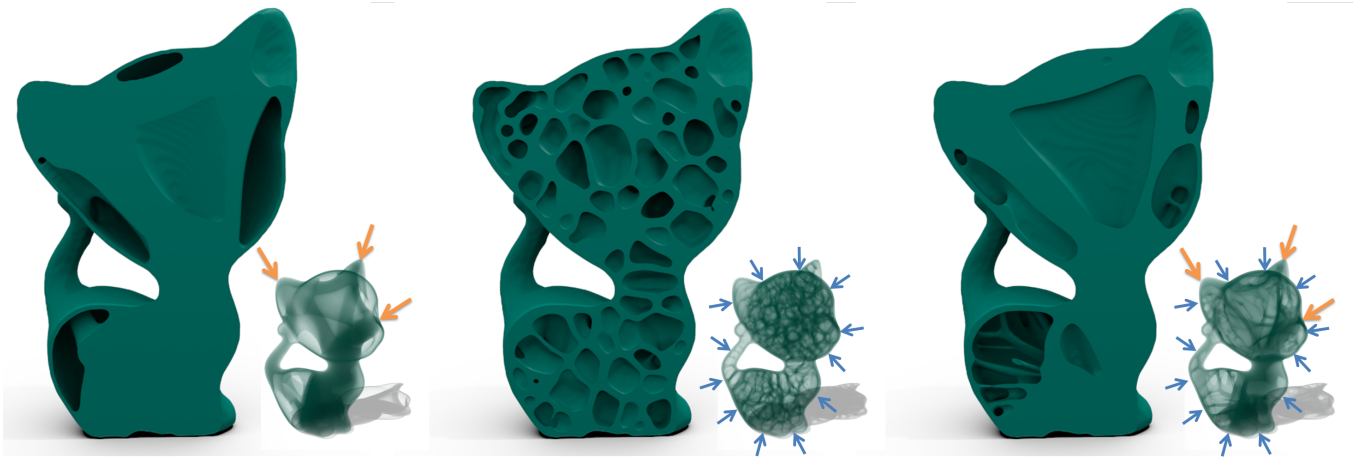


Fig. 15: The kitten model's interior under various external forces. In the first example, trusses connect the three loading locations (supporting each other), and connect to the bottom which is fixed on the ground. On its left, a short truss connects the tail, and through another long truss to the bottom. The retained boundary layer is considered in the elasticity analysis and sustains forces as well. Isotropic (middle) or more elongated (right) shapes can be generated by using distributed or concentrated external forces. In all examples a volume reduction of 50% was prescribed. Let us refer to the accompanying video for a demonstration of the iterative optimization process.

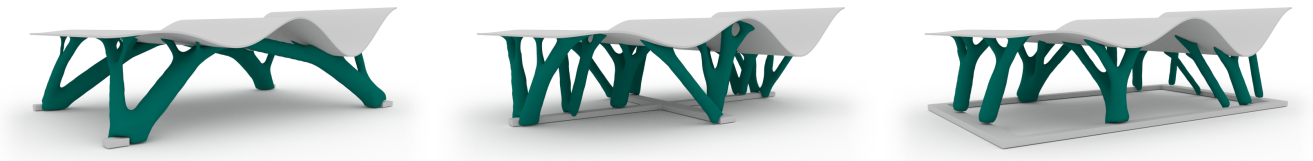


Fig. 16: Variants of optimized roof support structures with different prescribed fixations.

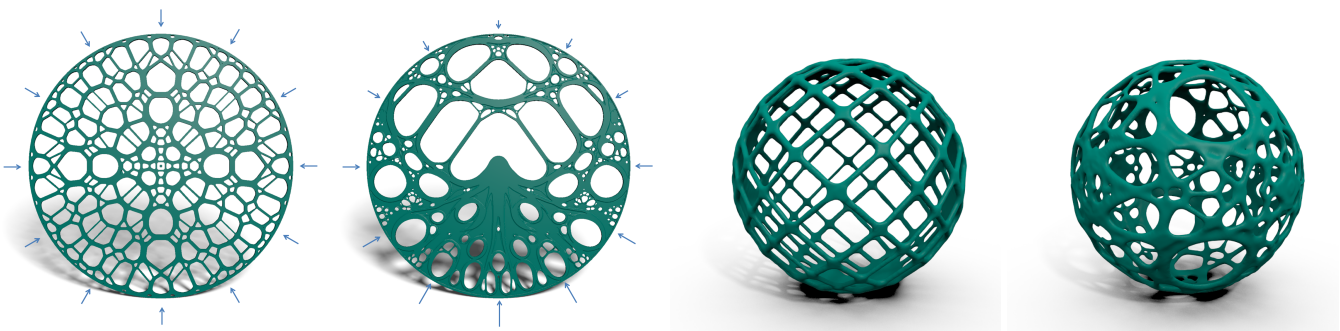


Fig. 17: Variants of disks (left) and spheres (right) under different boundary forces.

bunny's back. The visible change in shape is attributed to the fact that the material tends to allocate to support the increasing force.

Fig. 15 shows the effect of concentrated and distributed forces on the interior structures of the same 3D exterior shape. Interestingly, the distributed forces lead to interior shapes that are very similar to the honeycomb-like structures generated by Lu et al. [7]. In contrast, however, while their structures were generated by the particular construction principle using a Voronoi-guided hollowing strategy, the structures in our examples arise automatically in a mechanically optimized way under the influence of the given force distributions. Moreover, by changing the force constraints, we can flexibly steer the optimization process towards the generation of more elongated structures, very similar to the skin-frame structures used in the work of Wang et al. [6].

Fig. 16 shows different variants of optimized roof support structures. In these examples, only the locations of the trusses on the ground were prescribed, yet the trusses themselves evolved from a solid initial part under the load applied on the roof. Artistic support structures like this are well-recognized in civil engineering, and some of them have actually being built. For example, in the Qatar National Convention Center [53].

Finally, Fig. 17 shows surface-like structures that were optimized to certain external forces. To design the amulet on the left, uniformly distributed radial forces $f_i = k(p_i - c)$ were prescribed on the outer boundary, where k is a scaling factor, p_i is the position of elements on the circumference, and c is the amulet center. The element at the center is fixed. The amulet on the right was designed by moving c

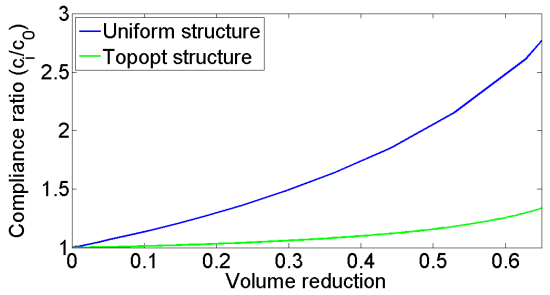


Fig. 18: Comparison of the compliance between a uniform support structure and optimized structures computed via shape optimization (see Fig. 15 (right)). The vertical axis denotes the compliance of a structure over the compliance of the initial solid object. The smaller the compliance, the more resistant the shape is. For different amounts of volume reduction, the optimized structure show stronger resistance.

upwards by the length of one element, slightly enlarging the downward forces. The sphere design on the left is obtained by fixing two poles, while tangential forces are applied at two opposite points on the equator. On the right, the sphere domain is divided by mutually perpendicular cuts into eight pieces of the same size. Forces are applied on the boundary of one piece, and then symmetry constraints are employed to guarantee the shape repetition in each piece. Symmetry is explicitly enforced using the method presented in Sec. 6.2, since otherwise the symmetry breaks due to numerical errors.

Resistance analysis To evaluate the mechanical properties of the structures generated by topology optimization, we compare their compliance to the compliance of a uniform structure. The uniform structure was obtained from an initial solid part, by gradually removing material around a set of uniformly distributed points. The graphs in Fig. 18 compare the compliance of the optimized shape in Fig. 15 (right) to the uniform support structure for the same kitten model, and increasing volume reduction. The boundary surface of a fixed size was maintained in both tests. As can be seen, the mechanically optimized shape shows a much higher resistance to the external forces than the uniform support structure.

In Fig. 19, we compare the compliance-optimized shapes generated by topology optimization with the stress-constrained honeycomb structures proposed in the work by Lu et al. [7]. We use the same models and set the same material properties and boundary conditions. In the first row, it can be seen from the stress distribution in the honeycomb structures (left) that forces are transmitted from the top, where external forces are applied, to the bottom, which is fixed, mostly through the neck and also through the tail. We prescribed the same amount of material volume and let topology optimization generate the compliance-optimized shape. The honeycomb structures exhibit a maximum stress that is about a factor of 2 larger than the maximum stress occurring at the same load in the compliance-optimized structures (middle). Topology optimization distributes the material to connect the top and the bottom via a straight structure through the neck, and also to create a truss con-

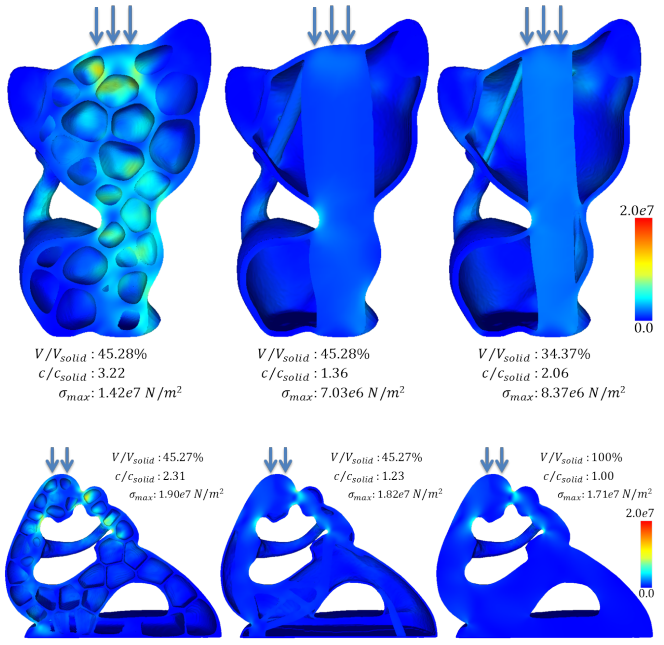


Fig. 19: Comparison between honeycomb structures (left) and the structures generated by topology optimization (middle and upper right), using the same exterior shapes. The solid shape (bottom right) serves as a reference in the experiment using the fertility model.

necting the tail. Even with a lower prescribed material volume (right), the resulting shape exhibits a smaller maximum stress than the honeycomb structures. In the second row, the comparison is performed using a different model, again showing lower maximum stresses in the compliance-optimized structures. A solid shape serves as a reference with maximum resistance.

Performance In Table 1 we analyze the performance of the system for various optimization tasks. The second group of columns shows the complexity of the used simulation models. The number of elements ranges from 1 to 15 millions, and even the largest model fits well into the memory of a single GPU.

The third group lists the target volume reduction, the number of iterations, and the ratio of the final shape's compliance over the initial solid shape's compliance. The convergence criterion of the optimization process is that a) the targeted volume was reached, and b) the change in compliance between the last iterations ($\frac{\|c_i+c_{i-1}-c_{i-2}-c_{i-3}\|}{c_i+c_{i-1}}$) is smaller than 0.5%. The convergence criterion in each iteration is the residual reduction to 10^{-2}. The compliance ratio demonstrates that even after a significant volume reduction, the compliance does not change too much compared to a completely solid configuration, especially the Michell optimal truss (Fig. 13). The compliance ratios, except those used in the comparison (Fig. 19), are measured with $\rho_{min} = 0.001$ representing void elements.

Finally, the last column of Table 1 summarizes timings for optimizing the topology of all models. Most models with millions of finite elements can be simulated within few minutes, indicating the possibility to even position the system in the consumer marker for designing customized

Model	Resolution	# Elements	Mem. [MB]	Vol. Red.	# Iter.	c/c_0	FEM [s]	OPT [s]	Total [s]
Roof	200×121×67	1.22×10^6	123.3	0.9	56	6.14	233.35	32.61	265.96
Michell	800×6×500	2.00×10^6	198.4	0.84	100	1.0009	507.15	67.70	574.85
Cantilever	200×100×100	2.00×10^6	198.4	0.8	50	3.06	113.38	32.25	145.63
Hand	401×250×178	4.39×10^6	579.6	0.6	31	1.36	333.43	87.67	421.10
Kittern	262×238×400	8.00×10^6	962.0	0.6	25	1.27	199.38	63.95	263.33
Bunny	390×302×386	11.79×10^6	1 531.0	0.6	28	1.29	308.26	92.10	400.36
Neptune	621×488×1000	14.19×10^6	4 050.5	0.6	23	1.17	569.83	139.33	709.16

TABLE 1: Performance statistics for different models.

shapes. The total computing time depends largely on the efficiency of the finite element analysis of elasticity.

Limitations The proposed system for topology optimization is stable and fast, yet it doesn't come without limitations. First, to 3D print the models, additional support structures are necessary during the printing process. The support structures might be located in the interior of a closed cavity and cannot be removed. Secondly, the mechanical analysis is based on the linear theory of elasticity, which is an idealization of printed objects. Thirdly, due to the non-convex nature of the structural optimization problem, the solution is likely to converge to a local minimum, and a mechanically more sound solution might exist. To alleviate this problem, sophisticated optimization algorithms such as the method of moving asymptotes [54] or the continuation method [2] can be employed. For instance, it has been reported that these methods can achieve a compliance that is up to 10% smaller than the compliance achieved via the standard optimization solver, yet they require a significantly larger number of optimization cycles [18].

8 Conclusion

In this work we have presented the design and performance of a high-performance system for topology optimization on desktop computer architectures. The system provides options to optimize shapes at high computational efficiency, and to give users more control over the shapes of the optimization results by including mechanical, fabrication- and user-specific constraints into the simulation process. We have shown practical results, including interiors that were optimized for maximum resistance to external loads, as well as compelling shapes which demonstrate the potential of topology optimization for shape design. Our approach can decrease the typical processing times of topology optimization from some hours to few minutes.

In the future we plan to extend on the current approach in the following ways: First, it is interesting to directly incorporate an additional constraint to avoid overhangs and support structures at all. Second, the current topology optimization minimizes the global compliance with a constraint on the volume. Possibilities to directly impose a constraint on the stress and to minimize the volume would be an interesting alternative we will further consider in an extension.

Acknowledgements

We thank the reviewers for their constructive suggestions, Lin Lu and Yuan Wei for providing the comparison data, Niels Aage and Ole Sigmund for helpful discussions, and Florian Reichl and Matthäus G. Chajdas for image and video rendering. This work was partially supported by the European Union under the ERC Advanced Grant 291372 SaferVis – Uncertainty Visualization for Reliable Data Discovery.

References

[1] J. Haslinger and R. Mäkinen, *Introduction to Shape Optimization*. Society for Industrial and Applied Mathematics, 2003.

[2] M. P. Bendsøe and O. Sigmund, *Topology optimization: theory, methods and applications*. Springer, 2003.

[3] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch, "Stress relief: Improving structural strength of 3d printable objects," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 48:1–48:11, Jul. 2012.

[4] J. Smith, J. Hodgins, I. Oppenheim, and A. Witkin, "Creating models of truss structures with optimization," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 295–301, 2002.

[5] E. Whiting, H. Shin, R. Wang, J. Ochsendorf, and F. Durand, "Structural optimization of 3d masonry buildings," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 159:1–159:11, 2012.

[6] W. Wang, T. Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, and X. Liu, "Cost-effective printing of 3d objects with skin-frame structures," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 177:1–177:10, Nov. 2013.

[7] L. Lu, A. Sharf, H. Zhao, Y. Wei, Q. Fan, X. Chen, Y. Savoye, C. Tu, D. Cohen-Or, and B. Chen, "Build-to-last: Strength to weight 3D printed objects," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 97:1–97:10, Jul. 2014.

[8] J. D. Deaton and R. V. Grandhi, "A survey of structural and multidisciplinary continuum topology optimization: post 2000," *Struct. Multidiscip. Optim.*, vol. 49, no. 1, pp. 1–38, 2014.

[9] O. Sigmund and K. Maute, "Topology optimization approaches," *Struct. Multidiscip. Optim.*, vol. 48, no. 6, pp. 1031–1055, 2013.

[10] N. Aage, E. Andreassen, and B. Lazarov, "Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework," *Struct. Multidiscip. Optim.*, pp. 1–8, 2014.

[11] B. Bickel and M. Alexa, "Computational aspects of fabrication: Modeling, design, and 3d printing," *IEEE Computer Graphics and Applications*, vol. 33, no. 6, pp. 24–25, Nov 2013.

[12] L. Liu, A. Shamir, C. Wang, and E. Whiting, "3d printing oriented design: Geometry and optimization," in *SIGGRAPH Asia 2014 Courses*, 2014.

[13] N. Umetani, B. Bickel, and W. Matusik, "Computational tools for 3d printing," in *ACM SIGGRAPH 2015 Courses*. New York, NY, USA: ACM, 2015.

[14] M. Alexa, B. Bickel, S. McMains, H. E. Rushmeier, M. Alexa, B. Bickel, S. McMains, and H. E. Rushmeier, "Computational aspects of fabrication," *Dagstuhl Reports*, vol. 4, pp. 126–150, 2015.

[15] N. Aage, M. Nobel-Jrgensen, C. Andreasen, and O. Sigmund, "Interactive topology optimization on hand-held devices," *Struct. Multidiscip. Optim.*, vol. 47, no. 1, pp. 1–6, 2013.

[16] O. Sigmund, "A 99 line topology optimization code written in matlab," *Struct. Multidiscip. Optim.*, vol. 21, no. 2, pp. 120–127, Apr. 2001.

[17] Y. Xie and G. Steven, "A simple evolutionary procedure for structural optimization," *Computers & Structures*, vol. 49, no. 5, pp. 885 – 896, 1993.

[18] X. Huang and Y.-M. Xie, "A further review of ESO type methods for topology optimization," *Struct. Multidiscip. Optim.*, vol. 41, no. 5, pp. 671–683, 2010.

[19] M. Y. Wang, X. Wang, and D. Guo, "A level set method for structural topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 12, pp. 227 – 246, 2003.

[20] G. Allaire, F. Jouve, and A.-M. Toader, "Structural optimization using sensitivity analysis and a level-set method," *Journal of Computational Physics*, vol. 194, no. 1, pp. 363 – 393, 2004.
[21] T. Nguyen, G. Paulino, J. Song, and C. Le, "A computational paradigm for multiresolution topology optimization (MTOP)," *Struct. Multidiscip. Optim.*, vol. 41, no. 4, pp. 525–539, 2010.
[22] S. Schmidt and V. Schulz, "A 2589 line topology optimization code written for the graphics card," *Computing and Visualization in Science*, vol. 14, no. 6, pp. 249–256, 2011.
[23] V. Challis, A. Roberts, and J. Grotowski, "High resolution topology optimization using graphics processing units (GPUs)," *Struct. Multidiscip. Optim.*, vol. 49, no. 2, pp. 315–325, 2014.
[24] A. N. Christiansen, J. A. Bærentzen, M. Nobel-Jørgensen, N. Aage, and O. Sigmund, "Combined shape and topology optimization of 3d structures," *Computers & Graphics*, vol. 46, no. 0, pp. 25 – 35, 2015.
[25] Q. Zhou, J. Panetta, and D. Zorin, "Worst-case structural analysis," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 137:1–137:12, Jul. 2013.
[26] X. Chen, C. Zheng, W. Xu, and K. Zhou, "An asymptotic numerical method for inverse elastic shape design," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 95:1–95:11, Jul. 2014.
[27] M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, and M. Gross, "Computational design of actuated deformable characters," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 82:1–82:10, Jul. 2013.
[28] N. Umetani and R. Schmidt, "Cross-sectional structural analysis for 3d printing optimization," in *SIGGRAPH Asia 2013 Technical Briefs*, ser. SA '13. New York, NY, USA: ACM, 2013, pp. 5:1–5:4.
[29] J. Panetta, Q. Zhou, L. Malomo, N. Pietroni, P. Cignoni, and D. Zorin, "Elastic textures for additive fabrication," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 135:1–135:12, Jul. 2015.
[30] C. Schumacher, B. Bickel, J. Rys, S. Marschner, C. Daraio, and M. Gross, "Microstructures to control elasticity in 3d printing," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 136:1–136:13, Jul. 2015.
[31] J. Pérez, B. Thomaszewski, S. Coros, B. Bickel, J. A. Canabal, R. Sumner, and M. A. Otaduy, "Design and fabrication of flexible rod meshes," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 138:1–138:12, Jul. 2015.
[32] J. Dumas, A. Lu, S. Lefebvre, J. Wu, and C. Dick, "By-Example Synthesis of Structurally Sound Patterns," *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015.
[33] K. Hu, S. Jin, and C. C. Wang, "Support slimming for single material based additive manufacturing," *Computer-Aided Design*, vol. 65, pp. 1 – 10, 2015.
[34] R. Prévost, E. Whiting, S. Lefebvre, and O. Sorkine-Hornung, "Make it stand: Balancing shapes for 3d fabrication," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 81:1–81:10, Jul. 2013.
[35] M. Bächer, E. Whiting, B. Bickel, and O. Sorkine-Hornung, "Spin-it: Optimizing moment of inertia for spinnable objects," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 96:1–96:10, Jul. 2014.
[36] P. Musialski, T. Auzinger, M. Birsak, M. Wimmer, and L. Kobbelt, "Reduced-order shape optimization using offset surfaces," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 102:1–102:9, Jul. 2015.
[37] B. Bickel, M. Bächer, M. A. Otaduy, H. R. Lee, H. Pfister, M. Gross, and W. Matusik, "Design and fabrication of materials with desired deformation behavior," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 63:1–63:10, Jul. 2010.
[38] M. Skouras, B. Thomaszewski, B. Bickel, and M. Gross, "Computational design of rubber balloons," *Comp. Graph. Forum*, vol. 31, no. 2pt4, pp. 835–844, May 2012.
[39] M. Skouras, B. Thomaszewski, P. Kaufmann, A. Garg, B. Bickel, E. Grinspun, and M. Gross, "Designing inflatable structures," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 63:1–63:10, Jul. 2014.
[40] M. Schwarz and H.-P. Seidel, "Fast parallel surface and solid voxelization on GPUs," *ACM Transactions on Graphics*, vol. 29, no. 6 (Proceedings of SIGGRAPH Asia 2010), pp. 179:1–179:9, Dec. 2010.
[41] J. Pantaleoni, "Voxelpipe: A programmable pipeline for 3d voxelization," in *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, ser. HPG '11. New York, NY, USA: ACM, 2011, pp. 99–106.
[42] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87. New York, NY, USA: ACM, 1987, pp. 163–169.
[43] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 351–358.
[44] Y. Zhu, E. Sifakis, J. Teran, and A. Brandt, "An efficient multigrid method for the simulation of high-resolution elastic solids," *ACM Trans. Graph.*, vol. 29, no. 2, pp. 16:1–16:18, Apr. 2010.
[45] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis, "Efficient elasticity for character skinning with contact and collisions," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 37:1–37:12, Jul. 2011.
[46] J. Wu, R. Westermann, and C. Dick, "Real-time haptic cutting of high resolution soft tissues," *Studies in Health Technology and Informatics*, vol. 196, pp. 469–475, 2014.
[47] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. SIAM, 2000.
[48] C. Dick, J. Georgii, and R. Westermann, "A real-time multigrid finite hexahedra method for elasticity simulation using CUDA," *Simulation Modelling Practice and Theory*, vol. 19, no. 2, pp. 801–816, 2011.
[49] S. Toledo, D. Chen, V. Rotkin, and O. Meshar, "TAUCS: A library of sparse linear solvers," 2003, http://www.tau.ac.il/∼stoledo/taucs.
[50] J. K. Guest, J. Prévost, and T. Belytschko, "Achieving minimum length scale in topology optimization using nodal design variables and projection functions," *International Journal for Numerical Methods in Engineering*, vol. 61, no. 2, pp. 238–254, 2004.
[51] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas, "Meshless animation of fracturing solids," in *ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005, pp. 957–964.
[52] A. Michell, "LVIII. The limits of economy of material in frame-structures," *Philosophical Magazine Series 6*, vol. 8, no. 47, pp. 589–597, 1904.
[53] M. Sasaki, T. Itō, and A. Isozaki, *Morphogenesis of flux structure*. AA Publications, 2007.
[54] K. Svanberg, "The method of moving asymptotes—a new method for structural optimization," *International Journal for Numerical Methods in Engineering*, vol. 24, no. 2, pp. 359–373, 1987.

Jun Wu is a PostDoc in the Computer Graphics and Visualization Group at the Technische Universität München (TUM), Germany. He received a PhD in Computer Science in 2015 from TUM, and a PhD in Mechanical Engineering in 2012 from Beihang University, Beijing, China, where he also received a B.Eng in Astronautics Engineering in 2006. His research is focused on physically-based modeling and interactive simulation, including deformable body simulation, virtual cutting, collision detection, haptic rendering, and shape optimization.

Christian Dick is a PostDoc in the Computer Graphics and Visualization Group at the Technische Universität München, Germany. He received a diploma in computer science in July 2007 and a PhD in January 2012, both from Technische Universität München. His research is focussed on interactive simulation and visualization methods, including physics-based simulation of deformable objects and fluids, simulation and visualization in the context of medical applications, as well as the visualization of very large scientific data sets.

Rüdiger Westermann studied computer science at the Technical University Darmstadt, Germany. He pursued his Doctoral thesis on multiresolution techniques in volume rendering, and he received a PhD in computer science from the University of Dortmund, Germany. In 2002, he was appointed the chair of computer graphics and visualization at the Technical University Munich. His research interests include scalable simulation and visualization algorithms, GPU computing, real-time rendering of large data, and uncertainty visualization.