Pacific Graphics 2016
E. Grinspun, B. Bickel, and Y. Dobashi
(Guest Editors)

Volume 35 (2016), Number 7

Direct shape optimization for strengthening 3D printable objects

Yahan Zhou Evangelos Kalogerakis Rui Wang Ian R. Grosse

University of Massachusetts Amherst

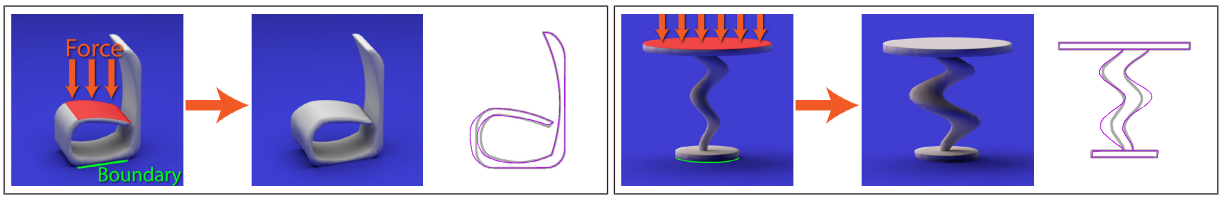


Figure 1: Two examples of optimized shapes using our algorithm. In each example, the user provides an input shape, the direction/region of the external force, and geometry constraints (such as fixed boundary). Our algorithm generates an output that remains similar to the input but can withstand three times as much force (as the maximum that the input shape can withstand). The silhouette images illustrate, from a cross section, how the shapes have changed (gray corresponds to the input shape and purple corresponds to the optimized).

Abstract

Recently there has been an increasing demand for software that can help designers create functional 3D objects with required physical strength. We introduce a generic and extensible method that directly optimizes a shape subject to physical and geometric constraints. Given an input shape, our method optimizes directly its input mesh representation until it can withstand specified external forces, while remaining similar to the original shape. Our method performs physics simulation and shape optimization together in a unified framework, where the physics simulator is an integral part of the optimizer. We employ geometric constraints to preserve surface details and shape symmetry, and adapt a second-order method with analytic gradients to improve convergence and computation time. Our method provides several advantages over previous work, including the ability to handle general shape deformations, preservation of surface details, and incorporation of user-defined constraints. We demonstrate the effectiveness of our method on a variety of printable 3D objects through detailed simulations as well as physical validations.

1 Introduction

The growing interest in 3D printing has led to increasing demands of algorithms that can optimize shapes to ensure functionality when they are printed. A fundamental aspect of object functionality is structural reliability. Ensuring that an object will not yield, or collapse, during typical use scenarios is a central goal for designers. It is particularly important when the object needs to withstand significant external force, such as a table, chair, or bench. While there has been a number of structural analysis techniques that identify potentially vulnerable areas in objects [US13, ZPZ13], it is often hard for artists and novice users to select the best possible corrections in their designs that guarantee structural reliability, and minimally affect surface appearance as well as the use of printing material. Thus, an automatic method for computing optimal shape corrections would be of great help to designers. Existing approaches are limited to perform heuristics-based local thickening of parts [SVB*12], in conjunction with other alternatives allowed by the user and the particular printing method, such as introducing hollowing or supporting structures [SVB*12, WWY*13, LSZ*14, VGB*14] and changing printing directions [US13]. Local thickening is the most straightforward solution to improve the object's structural strength, however, it can be suboptimal in terms of the increased use of material and the impact to surface appearance (Figure 2). In addition, it is not always as effective as alternative shape deformations to increase the object's structural strength, such as straightening, bending, stretching or other shape changes.

One approach to strengthen 3D shapes is to perform optimization directly in the input shape representation to reduce mechanical stress. This type of approach has so far been considered infeasible in computer graphics for a variety of reasons [SVB*12], including the perceived need to run a physics simulation step whenever the shape is changed, use of numerical gradients causing instabilities and errors, and overall computational complexity. Similar observations have been made in the mechanical engineering literature. Shape optimization techniques were mainly successful in the case of parametric surface representations of mechanical components, such as Bezier or B-spline patches [HM03, WMC08, BCC*10] with limited number of control points (e.g., tens or a few hundreds), procedural models based on high-level design variables [BR88, RG92, Tor93], or alternatively, level set representations [AJT02, AJ08, DMLK13] in relatively low resolution grids (e.g., with a few thousands cells in total). Directly optimizing an input mesh, which is by far the most common representation of shapes in computer graphics pipelines, has been discouraged due to the perceived inability to maintain surface smoothness, detail, and satisfactory resolutions [BF84, WMC08].

In contrast to the common belief, our paper demonstrates that an optimization approach operating directly on an input mesh to improve structural reliability is totally feasible. It in fact offers a number of distinct advantages, including the ability to execute a multitude of generic shape deformations, preservation of surface de-

tails, and easy incorporation of user-defined design constraints. We found a number of key ingredients necessary to enable such an optimization approach. First, we observe that performing the physics simulation to solve for mechanical stresses is itself an optimization problem, which can be solved as part of the whole shape optimization process in a unified framework. Second, we employ geometric constraints to maintain desired surface properties, including surface details and symmetry. Third, instead of using first-order optimization methods, such as steepest descent, we adapt a second-order (Newton-based) method along with analytic gradients to improve convergence and computational performance. In contrast to previous work in computer graphics that mainly focuses on improving the structural strength of objects under gravity, we apply our method in various practical scenarios of significant external forces. For example, our method optimizes chairs and tables with significant weight acting on them, where alternative optimization techniques, such as hollowing, would be inadequate. We demonstrate the effectiveness of our method through both detailed simulations and physical validity tests (using a professional force gauge) in a variety of 3D printable objects.

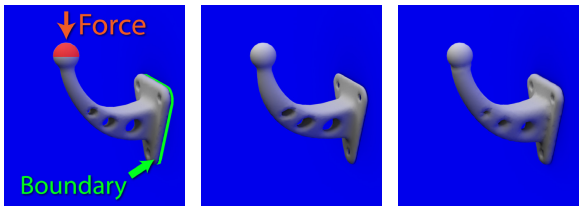
Contributions. We present a direct shape optimization method to strengthen 3D printable objects. Our method operates directly on the input mesh, and we introduce an optimizer integrated with physics simulator to iteratively reduce stress under significant external force. In contrast to previous shape deformation techniques that focus on local part thickening and rely on heuristics under particular input surface scenarios, our method is more general, allowing shape deformations that are not constrained to particular types. Our method is applicable to a wide variety of objects and external force scenarios, and we verify results with physical validations.

2 Related Work

We now discuss prior approaches in the context of mechanical stress reduction and shape optimization.

Strengthening objects. Our work is related to research on improving the structural strength of 3D printable objects, which aims to diminish the chances that objects will break during common use scenarios. A number of approaches have been proposed towards achieving this goal, including modifications to the interior structure of objects, such as hollowing [SVB*12, LSZ*14, VGB*14] or internal skin framing [WWY*13], addition of supporting struts [SVB*12], change of printing directions [HBA13, US13], and shape deformations [SVB*12, US13].

The above approaches are complementary to each other. In other words, to reduce stress in objects, a common practice is to use a combination of these approaches. However, there are cases where adding supporting structures or modifying the internal object structure are undesirable. For example, adding struts can be inappropriate for artistic designs, since they alter the appearance and aesthetics of the designed objects. Hollowing may increase mechanical stress in the presence of large external forces, and sometimes requires users to perform additional post-processing on the printed object e.g., remove internal powder [WWY*13]. Changing the printing direction may also not always yield a decrease in mechanical stress [HBA13]. Thus, applying shape deformations remains one of the main options to reduce mechanical stress, and in some cases, the most desirable one. Existing shape deformations for reducing mechanical stress have been so far limited to uniform part thickening. Stava et al. [SVB*12] identifies thin, vulnerable, parts



(a) Original shape (b) Our method (c) Local Thickening

Figure 2: Given an input object of a utility hook (a), we apply our method (b) to optimize the shape such that it can withstand a specified external force load. We compare against local thickening (c) that can alternatively be used to make the object withstand the same force load. The volume of our result is 20% less than local thickening, making our method more cost-efficient for 3D printing. Also, our result is better at preserving shape features.

in an object and applies a heuristic to increase their thickness assuming that these parts are thin tubes and acting forces cause bending. Vulnerable parts can be alternatively identified with worst-case load analysis methods [ZPZ13]. Umetani and Schmidt [US13] highlight vulnerable areas to users and allow them to interactively thicken these areas. A downside of part thickening is that it increases material use and can affect the appearance of the model in an undesirable manner. In addition, Stava et al.'s heuristic applies only to tubular surfaces, while interactive thickening can become laborious for complex shapes.

In contrast to the above heuristics-based and interactive part thickening approaches, our method automatically computes optimal shape deformations to reduce mechanical stress. Our optimization does not make any particular assumptions about the geometry of vulnerable parts. It does not restrict deformations to a particular type. It lets the optimization to perform shape corrections under the constraint that the user's design should be as intact as possible. While it is possible for users to interactively discover the optimal shape that would meet certain physical constraints, manual shape edits are often tedious, especially for casual modelers. Our method is largely automatic and does not require human supervision.

Shape optimization. Shape optimization approaches have been demonstrated for improving the balance [PWLSH13], spinnability [BWBSH14], and aggregate mass properties of objects [MAB*15], as well as for particular fabrication scenarios, such as inverse elastic shape design [CZXZ14], shape printing with microstructures [PZM*15, SBR*15], and support material reduction [HJW15]. To achieve their goal, some of these methods use cage-based deformations [PWLSH13, BWBSH14] or displacements with offset surfaces [MAB*15]. Our approach instead aims to reduce mechanical stress of an input shape under a given force load. Reducing mechanical stress often requires localized changes, sparsely applied throughout the input object, which cannot be captured well by the above deformation techniques.

Thera is a large body of research in shape optimization approaches for stress reduction in the mechanical engineering literature. These approaches are often restricted to the case of parametric surfaces [HM03, WMC08, BCC*10], such as B-splines or Bezier patches, subdivision surfaces [BRC16], level set representations [AJT02, AJ08, DMLK13], specialized topology modifications [All12, BS13], or procedural models based on high-level design variables [BR88, RG92, Tor93]. Parametric surface representations impose limitations on the surface topology of input shapes, while level set and subdivision methods tend to lose surface detail,

since they are applied to low-resolution grids and coarse meshes. Topology modifications may alter the appearance or aesthetics of the users' original designs in an undesirable manner. Procedural models handle only particular families or types of shapes i.e., those that can be created through given design parameters. Our method instead directly optimizes mesh representations, which are the most common representations in shape design. Mesh optimization approaches have also been investigated in mechanical engineering [MSS05, LBT11, HSB14], but require filtering (e.g., Laplacian smoothing), or other forms of post-processing at each optimization step to combat instabilities and surface noise. As a result, in these methods, surface detail and features are lost, or convergence to a plausible solution is not guaranteed. In contrast to the above mechanical engineering techniques, our method combines physics simulation and feature-preserving shape deformation into one optimization problem, which is efficiently solved through Netwon-based techniques. Our method is also designed to preserve surface features, which is highly desirable in shape design.

Perhaps most similar to our approach is the recent method by Martinez et al. [MDLW15]. Their method optimizes the geometry of an object given an exemplar shape capturing its target appearance, and desired structural properties including mechanical stress. Similarly to level-set based mechanical engineering approaches, their method represents shapes with binary 2D grids of elements (i..e, elements can be "void" or "solid") to execute the optimization. Their method can only produce 3D shapes made of planar sections, each represented by its own grid. Our method instead performs optimization directly on an input 3D mesh without posing particular constraints on the shape of an object.

3 Overview

Our goal is to optimize the shape of a 3D object so that it can withstand forces in common use scenarios. For example, given an input chair or table (Figure 1) along with its material properties (e.g., thermoplastic), boundary conditions (e.g., standing on the floor), and external forces exerting on specified regions of the surface, we optimize the shape to ensure that it will not yield under that load. An object yields when the mechanical stress induced by the loading exceeds its capacity to resist the load. Thus, we aim to reduce the mechanical stress of the object so that it is lower than the yield strength of its material. Given the force load, the mechanical stress is a function of the material properties of the object and its shape. We assume the material properties are pre-specified and determined by the printing material. In addition, we assume the input shape is represented by a volumetric mesh. If the input shape is a surface mesh, we tetrahedralize it to obtain its volumetric version. We tetrahedralize the shape as uniformly as possible, and the resulting surface is naturally resampled as part of the tetrahedralization.

A naive solution to this problem would be to exhaustively try different perturbations of the input mesh, and for each perturbed shape, use physics simulation to compute the shape deformation and stresses caused by the load. Then gradient descent with numerical gradients over the mesh points could be used to reduce stress at points exceeding the material's yield strength. Such approach would be prohibitively slow, extremely prone to numerical instabilities, and lead to rather unpredictable results.

Instead, we integrate the physics simulation into the shape optimization, which enables the computation of analytic gradients. To begin, given an input *volumetric mesh* $\mathbf{X}^0$ with vertex coordinates $\{\mathbf{x}_1^0, \mathbf{x}_2^0, ..., \mathbf{x}_M^0\}$, our optimization method solves for: (i) vertex coordinates of the optimized shape $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M\}$ (called material coordinates or rest state in the context of physics simulation), and (ii) vertex coordinates of the deformed shape $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_M\}$ caused by the load (called world coordinates or deformed state). To ensure that the object will not yield, we introduce *inequality constraints* in formulation, expressing that the stress at each tetrahedron t of the volumetric mesh should be lower than the yield strength C of the material. Specifically, we use the von Mises stress $\hat{\sigma}_t$ per element, which is a widely employed measure of stress in computer graphics and mechanical engineering [Mis13] for checking whether an object yields under a given load. The stress depends on the deformation of the object under the load, thus, as part of our optimization problem, we need to solve for the deformed state (world coordinates) of the object when it reaches a force equilibrium. We use *equality constraints* to express that the deformed state must reach the force equilibrium under boundary conditions. Boundary conditions involve boundary vertices $v \in B$ that remain fixed during deformation (e.g., vertices at the bottom of table legs touching the floor). For these boundary vertices, we use equality constraints to express that they should remain fixed. For all other vertices $v \notin B$, we use equality constraints to express that the sum of both internal and external forces $\mathbf{f}_v$ should be zero.

In addition to the above simulation constraints, we include additional equality constraints to preserve important geometric properties related to the aesthetics and functionality of the input shape, as well as to ensure numerical stability during the optimization. These are called *geometric constraints*, which include: (i) *symmetry constraints* to ensure that the optimized shape will preserve local and global symmetries found in the original shape (e.g. reflective and rotational symmetries); (ii) *interior uniformity constraints* to ensure that the finite elements (tetrahedra) remain well-formed and all interior mesh vertices remain uniformly spaced during the optimization, a necessary condition for numerical stability and accuracy; (iii) *user-defined constraints* that practically ensure that the object meet the design constraints (e.g. the height of the table should be fixed) after optimization. These user-defined geometric constraints are optionally given as part of the input to our algorithm, and depend on the user's preferences on how the object will be used. Finally, we use an objective function to penalize geometric deviation of the optimized shape from the original shape. Ideally, the surface of the optimized shape should remain as similar as possible to the original shape. Thus, our objective function is designed to minimize changes to the original surface, preserve its original details and smoothness. Our optimization problem is defined as follows:

$$\arg\min_{\mathbf{X}} \; D(\mathbf{X}, \mathbf{X}^0) \qquad \text{(Objective function)}$$

subject to:

$$\forall v \in B\text{: } \mathbf{x}_v = \mathbf{p}_v, \;\; \forall v \notin B\text{: } \mathbf{f}_v(\mathbf{X}, \mathbf{P}) = \mathbf{0} \qquad \text{(Simulation constraints)}$$

$$\forall t\text{: } \hat{\sigma}_t^2(\mathbf{X}, \mathbf{P}) < C \qquad \text{(Stress constraints)}$$

$$g(\mathbf{X}, \mathbf{X}^0) = 0 \qquad \text{(Geometric constraints)}$$

where v is a mesh vertex, t is a tetrahedron, $D(\mathbf{X}, \mathbf{X}^0)$ is our objective function expressing the geometric dissimilarity between the optimized shape $\mathbf{X}$ and original shape $\mathbf{X}^0$, $\hat{\sigma}_t$ is the von Mises stress at a tetrahedron t (expressed as a function of the optimized shape and its deformed state $\mathbf{P}$), $\mathbf{f}_v$ is the sum of internal and external forces on a vertex v, and $g(\mathbf{X}, \mathbf{X}^0)$ is a function encoding the geometric relationships of points in the original and optimized shape.

4 Shape Optimization

We now discuss how the functions in the above optimization problem are computed, and how our algorithm solves the optimization.

4.1 Elasticity Model

The forces per vertex consist of external and internal forces. The external forces are part of the input – they include gravity and force exerted on surface areas according to a typical use scenario for the object. The external forces are always applied on a fixed set of vertices defined by the user and are constant (during optimization, the force area may change due to shape deformations, but the total external force remains the same). The internal forces are elastic forces resulting from the deformation of the object – they are a function of the vertex coordinates of the undeformed shape $\mathbf{X}$ (material coordinates) and its deformed version $\mathbf{P}$ (world coordinates). Since the input shape is discretized into finite elements (tetrahedra), the deformation is measured per tetrahedron t. Consequently, the total force per vertex v can be expressed as:

$$\mathbf{f}_v(\mathbf{X},\mathbf{P}) = \sum_{\forall t,\, v\in t} \mathbf{f}_{t,v}(\mathbf{X},\mathbf{P}) + \mathbf{f}_v^{ext} \tag{1}$$

where $\mathbf{f}_v^{ext}$ are the external forces per vertex, and $\mathbf{f}_{t,v}(\mathbf{X},\mathbf{P})$ expresses the internal elastic forces per vertex of each tetrahedron t as a function of the optimized and deformed shape. As commonly done in structural analysis methods, we assume linear elasticity to model the elastic forces in our object. The elastic forces are modeled as the partial derivatives of the strain energy per tetrahedron with respect to its deformed vertex coordinates [RH08]:

$$\mathbf{f}_{t,v}(\mathbf{X},\mathbf{P}) = \frac{\partial U_t(\mathbf{X},\mathbf{P})}{\partial \mathbf{p}_v} \tag{2}$$

The strain energy per tetrahedron t is in turn a function of its volume $\mathcal{V}_t$, its deformation expressed through its strain tensor $\boldsymbol{\varepsilon}_t$ (symmetric rank-two tensor i.e., 3x3 matrix), and the material properties $\mathbf{E}$ of the object (rank four tensor), which are given as part of the input. In all our experiments, we assume isotropic material. In this case, the material properties tensor depends only on the Young's modulus, which measures the object's stiffness, and Poisson's ratio, which measures the object's contraction when it is stretched. The strain energy is expressed as follows [RH08]:

$$U_t(\mathbf{X},\mathbf{P}) = \frac{\mathcal{V}_t}{2}\boldsymbol{\varepsilon}_t : \mathbf{E}\boldsymbol{\varepsilon}_t \tag{3}$$

where : denotes double tensor contraction, which corresponds here to the double inner product between tensors. The strain tensor is a function of the deformed and undeformed shape, while the volume of the tetrahedron is a function of the vertex coordinates of the optimized shape. Specifically, if v_0, v_1, v_2, v_3 are the four vertices of a tetrahedron and $\bar{\mathbf{X}}_t$ represents its edge vectors stored in a 3x3 matrix $\bar{\mathbf{X}}_t = [\mathbf{x}_{v_1} - \mathbf{x}_{v_0}, \mathbf{x}_{v_2} - \mathbf{x}_{v_0}, \mathbf{x}_{v_3} - \mathbf{x}_{v_0}]$, the volume is simply determined as: $V_t(\mathbf{X}) = \frac{1}{6}|\bar{\mathbf{X}}_t|$, where $|\cdot|$ refers to the determinant of the matrix. The strain can be computed from the per-tetrahedron deformation gradient, which describes its scaling and rotation due to the compression or elongation of the object's material. By representing the edge vectors for each deformed tetrahedron in a 3x3 matrix $\bar{\mathbf{P}}_t = [\mathbf{p}_{v_1} - \mathbf{p}_{v_0}, \mathbf{p}_{v_2} - \mathbf{p}_{v_0}, \mathbf{p}_{v_3} - \mathbf{p}_{v_0}]$, the deformation gradient is computed as:

$$\mathbf{F}_t(\mathbf{X},\mathbf{P}) = \bar{\mathbf{P}}_t \bar{\mathbf{X}}_t^{-1} \tag{4}$$

We follow the invertible finite element methods [ITF04] to extract the scaling from the deformation gradient using SVD: $\mathbf{F}_t = \mathbf{U}_t\boldsymbol{\Sigma}_t\mathbf{V}_t^T$. During each iteration of the optimization, we assume that each tetrahedron undergoes an infinitely small rotation. This assumption works well in practice. Since the rotation matrices $\mathbf{U}$ and $\mathbf{V}$ are constant during each optimization step, only the scaling matrix is a function of the optimized and deformed shape. We use the Cauchy strain tensor to express the strain based on the scaling per tetrahedron: $\boldsymbol{\varepsilon}_t = \boldsymbol{\Sigma}_t - \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Based on the SVD of the deformation gradient and Equation 4, we can express the strain tensor as a function of the optimized and deformed shape:

$$\boldsymbol{\varepsilon}_t(\mathbf{X},\mathbf{P}) = \mathbf{U}_t^T \bar{\mathbf{P}}_t \bar{\mathbf{X}}_t^{-1} \mathbf{V}_t - \mathbf{I} \tag{5}$$

Replacing the strain tensor function in Equation 2, the force function can be rewritten as:

$$\mathbf{f}_{t,v}(\mathbf{X},\mathbf{P}) = \frac{\partial U_t(\mathbf{X},\mathbf{P})}{\partial \mathbf{p}_v} = \mathcal{V}_t \left(\mathbf{U}_t^T \bar{\mathbf{P}}_t \bar{\mathbf{X}}_t^{-1} \mathbf{V}_t - \mathbf{I}\right) : \mathbf{E}\frac{\partial \boldsymbol{\varepsilon}_t}{\partial \mathbf{p}_v} \tag{6}$$

where the derivative of the strain tensor $\boldsymbol{\varepsilon}_t$ for a tetrahedron with respect to each of the three coordinates of a deformed vertex $\mathbf{p}_v$ at that tetrahedron are the following:

$$\frac{\partial \boldsymbol{\varepsilon}_t}{\partial p_{v,k}} = \mathbf{U}_t^T \frac{\partial \bar{\mathbf{P}}_t}{\partial p_{v,k}} \bar{\mathbf{X}}_t^{-1} \mathbf{V}_t \tag{7}$$

Here k is an index for the x, y or z coordinate in the vertex $\mathbf{p}_v$. The partial derivative of each element (i, j) in the matrix $\bar{\mathbf{P}}_t$ storing the deformed edges are the following:

$$\frac{\partial \bar{\mathbf{P}}_t(i,j)}{\partial p_{v,k}} = \begin{cases} 1 & \text{if } v = v_j \text{ and } k = i \\ -1 & \text{if } v = v_0 \text{ and } k = i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

To solve our optimization problem, we also need analytic gradients of the force function. We refer the reader to the supplementary material that includes the analytic gradients.

4.2 Stress Function

As described in Section 3, during the optimization, we set inequality constraints to ensure that the von Mises stress in the object will not exceed the material's yield strength. The von Mises stress is extracted from the stress tensor that characterizes the state of stress at each tetrahedron. The stress tensor is a symmetric 3x3 matrix:

$$\boldsymbol{\sigma}_t = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \tag{9}$$

Given the strain tensor $\boldsymbol{\varepsilon}_t$ of a tetrahedron t, the stress tensor is expressed as $\boldsymbol{\sigma}_t = \mathbf{E}\boldsymbol{\varepsilon}_t$. Since the strain tensor is a function of the undeformed shape and its deformed version (Equation 5), the stress tensor elements are also functions of their vertex coordinates. The von Mises stress is computed from the stress tensor as follows [Mis13]:

$$\begin{aligned}\hat{\sigma}_t^2(\mathbf{X},\mathbf{P}) =& 0.5[(\sigma_{11}-\sigma_{22})^2 + (\sigma_{22}-\sigma_{33})^2 + (\sigma_{33}-\sigma_{11})^2 \\ &+ 6(\sigma_{23}^2 + \sigma_{31}^2 + \sigma_{12}^2)]\end{aligned} \tag{10}$$

Our optimization procedure requires the analytic gradient and Hessian of the above function, which we also provide in the supplementary material.

4.3 Geometric Constraints

We now describe the geometric constraints that we use to preserve important geometric properties of the input shape related to its aesthetics, functionality and numerical stability during optimization.

Symmetry constraints. Symmetry is an important factor in the aesthetics of man-made objects. Ignoring symmetry can cause a shape to become asymmetric during optimization, which is undesirable (Figure 3). To incorporate symmetry, we first detect dominant planar reflective and rotational symmetries in the original shape [MGP06]. We aim to preserve these symmetries in the optimized shape. A planar reflective symmetry constraint can be defined through a reflection matrix $\mathbf{S}_m$. For each pair of symmetric points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the optimized shape, we set $\mathbf{S}_m\mathbf{x}_i = \mathbf{x}_j$. A rotational symmetry is similarly defined through a rotation matrix $\mathbf{S}_r$. For a set of k ordered points $\{\mathbf{x}_1, \mathbf{x}_2 ... \mathbf{x}_k\}$ related through a rotational symmetry, we set $\mathbf{S}_r\mathbf{x}_i = \mathbf{x}_{i+1} (1 \leq i < k)$ and $\mathbf{S}_r\mathbf{x}_k = \mathbf{x}_1$. Other types of symmetry can also be supported.

(a) original shape (b) w/o symmetry (c) with symmetry

Figure 3: Without symmetry constraints, the resulting shape (b) becomes asymmetric, which is undesirable.

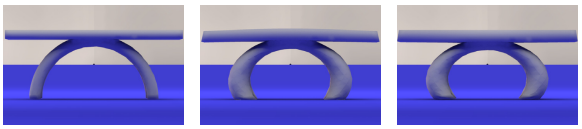
Interior uniformity constraints. During optimization, a key factor for numerical stability and convergence is to promote well-shaped elements, free of self-intersections, and all interior (non-surface) vertices are uniformly spaced. Leaving the interior vertices float freely during optimization would also cause inaccuracy in the stress calculations and analytic gradients. We account for this by forcing all interior vertices of the optimized volumetric mesh to be positioned exactly at the centroid of their neighbors. Mathematically, for each interior vertex, we set its position to be:

$$\mathbf{x}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \tag{11}$$

where $\mathcal{N}(i)$ is the set of neighboring vertices. The above expression can be also thought of as forcing the graph laplacian of the interior vertices to be zero. This constraint is also guaranteed during initialization where we tetrahedralize the input shape.

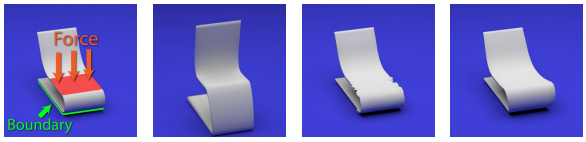
User-defined constraints. Our method allows the user to specify additional geometric constraints for aesthetics or functionality purposes. The most common constraint is to force certain edges to be straight, or certain vertices to remain fixed during optimization. An example is shown in Figure 4: without user-defined constraints, the top of the shape is deformed. To keep it flat and the edges straight, the top surface is explicitly marked as fixed. The result of optimization is shown in Figure 4c. These user-defined constraints can be achieved by setting $\mathbf{x}_i = \mathbf{x}_i^0$ for all vertices that need to remain fixed, where $\mathbf{x}_i^0$ is the position of the vertex on the original shape.

Summary. In general, we represent all the above geometric constraints in our formulation as equality constraints, in the form of $g(\mathbf{X}, \mathbf{X}^0) = 0$. Each g is a linear function of the vertices on the input and optimized shapes.



(a) original shape (b) w/o user constraints (c) with user constraints

Figure 4: Without user-defined constraints, the top surface of the resulting shape (b) is deformed / curved, which is undesirable.



(a) original shape (b) intrinsic only (c) extrinsic only (d) both

Figure 5: Without extrinsic distance, the result (b) deviates significantly from the original; without intrinsic distance, the result (c) contains bumpy surface/jaggy edges and lose surface details. Including both measures is important to ensure the result is geometrically similar to the original.

4.4 Objective function

Our objective function aims to penalize surface dissimilarity between the original and the optimized shape. The goal is to keep the optimized shape as similar as possible to the original. We measure surface distance using an *extrinsic* distance as well as an *intrinsic* distance. The extrinsic distance directly computes the distance between surface vertices of the two shapes – it prevents the optimized surface to deviate significantly away from the original surface in Euclidean space. The intrinsic distance computes the distance of differential surface properties – it preserves surface details and smoothness, and prevents the resulting surface from becoming bumpy/jaggy. An example is shown in Figure 5. In practice we found that both measures are important for producing an optimized shape that is geometrically similar to the original shape.

Extrinsic distance. Our extrinsic distance considers the squared Euclidean distance between the vertices of the original surface $\mathbf{X}^0$ and the optimized surface $\mathbf{X}$:

$$D_{extrinsic}(\mathbf{X}, \mathbf{X}^0) = \sum_{i \in \mathcal{S}} \| \mathbf{x}_i^0 - \mathbf{x}_i \|^2 \tag{12}$$

where $\mathcal{S}$ is the set of surface vertices. We do not consider the interior vertices as they are not related to shape appearance. Note that during optimization the number of surface vertices does not change, and we have an explicit correspondence between the original and optimized surface vertices, thus this distance is easy to compute.

Intrinsic distance. Our intrinsic distance compares the differential surface properties in terms of surface Laplacians. Surface Laplacians have been extensively used in detail-preserving shape deformations due to their ability to capture information about the local shape of the surface, and in particular the size and orientation of local surface details [SCOL*04, Sor05]. The Laplacian of a surface vertex can be computed as follows:

$$\mathcal{L}(\mathbf{x}_i) = \sum_{j \in \mathcal{N}(i)} \omega_{i,j} (\mathbf{x}_i - \mathbf{x}_j) \tag{13}$$

where $\mathcal{N}(i)$ here represents the neighboring vertices on the surface (we exclude any interior vertices from the calculation of surface Laplacians), and $\omega_{i,j}$ are weights controlling the desired properties for the Laplacian [WMKG07]. We experimented with both uniform

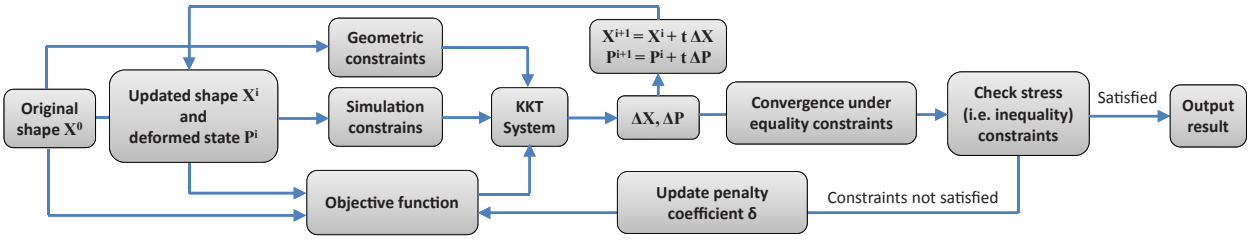


Figure 6: Block diagram of our optimization approach.

weights ($\omega_{i,j} = 1/|\mathcal{N}(i)|$) and the cotangent scheme [MDSB03] and in practice did not observe any significant differences of the two, due to the relatively uniform tesselations of our input meshes. Thus for simplicity, we use uniform coefficients.

Following [SCOL*04], our intrinsic distance computes differences between transformed Laplacians of the original surface vertices and Laplacians of the corresponding surface vertices in the optimized shape:

$$D_{intrinsic}(\mathbf{X},\mathbf{X}^0) = \sum_{i\in\mathcal{S}} \| \mathbf{T}_i\mathcal{L}(\mathbf{x}_i^0) - \mathcal{L}(\mathbf{x}_i) \|^2 \qquad (14)$$

where $\mathbf{T}_i$ is a transformation matrix that optimally translates, rotates, and scales the original surface Laplacian of each vertex i before comparing it to its surface Laplacian on the optimized surface. The reason for transforming the surface Laplacian is that we want to allow local surface rotations and scaling since they can both contribute to reducing the mechanical stress of the object without significantly affecting its surface apperanace. Following [SCOL*04], the transformations can be computed by aligning each surface neighborhood in the original shape with the one on the optimized shape in least-squares sense:

$$\mathbf{T}_i = \arg\min_{\mathbf{T}} \left(\| \mathbf{T}\mathbf{x}_i^0 - \mathbf{x}_i \|^2 + \sum_{j\in\mathcal{N}(i)} \| \mathbf{T}\mathbf{x}_j^0 - \mathbf{x}_j \|^2 \right) \qquad (15)$$

The transformation matrices are internal variables in our algorithm and are solved simultaneously with the optimized shape $\mathbf{X}$. We experimented with transformations restricted to rotation, translation and uniform scaling (as in [SCOL*04]), but also with affine transformations. In the case of affine transformations, we compute them using the 2-ring surface vertex neighborhoods in the above expression to ensure enough number of constraints for the least-squares solution. In the case of planar neighborhoods on the original surface, the least-squares solution can still be ill-defined. For these neighborhoods we compute transformations that preserve their planarity (see also supplementary material).

One consequence of allowing affine transformations is that under significant deformation it can lead to shearing distortions [FAT07]. However, in our case, due to the extrinsic distance and overall optimization, surface changes are generally moderate. In addition, non-uniform local stretching can be particularly effective for reducing mechanical stress in man-made objects without significant impact on the surface appearance. Therefore we adopt affine transformations by default and use those in our experiments. If needed, users can easily override this option and execute the shape optimization by restricting the transformations to rotations and uniform scaling only, as in [SCOL*04].

Objective function. Our objective function is defined as a weighted sum of the intrinsic and extrinsic distances:

$$D(\mathbf{X},\mathbf{X}^0) = w_1 D_{intrinsic}(\mathbf{X},\mathbf{X}^0) + w_2 D_{extrinsic}(\mathbf{X},\mathbf{X}^0) \qquad (16)$$

where w_1 and w_2 are the weights. By default, we set $w_1 = 10^5$ and $w_2 = 1$, which worked well in our tests. Reducing w_1 causes loss of surface details and smoothness, while reducing w_2 causes the optimized surface to deviate significantly from the original. Both intrinsic and extrinsic terms are quadratic forms in the surface vertices, thus their gradients and Hessians are straightforward to compute. We note that our objective function is scale-invariant, and the intrinsic and extrinsic distances have the same unit.

4.5 Optimization approach

We now describe the procedure to solve our optimization problem. Although our objective function has the desirable property to be a quadratic (i.e., convex) function on the shape $\mathbf{X}$ we wish to optimize, the force and von Mises stress functions involved in the constraints are non-linear and non-convex over both the shape $\mathbf{X}$ and its deformed version $\mathbf{P}$. In particular, any small change in $\mathbf{X}$ or $\mathbf{P}$ can easily cause a large violation of the force equilibrium constraints. An additional challenge is that for shapes of reasonably high resolution, the number of unknown variables is on the order of tens of thousands at least. The number of unknown variables is equal to the number of vertices of the optimized shape, multiplied by a factor of 6, since their deformed state is included in the unknowns, and we deal with 3D coordinates. Thus, to solve our problem, we need a non-linear, large-scale optimization method. An important fact that we can leverage is that all functions are twice differentiable, and we are able to derive both analytic gradients and Hessians. We experimented with various solvers including Sequential Quadratic Programming, interior-point, and penalty methods [NW06]. None of the techniques worked out of the box. We implemented our own variant of the penalty method [McC71] which we describe below.

The original penalty method reformulates a constrained optimization problem as a sequence of unconstrained problems. It has the advantage of fast local convergence guarantees under relatively weak assumptions [BM08]. The unconstrained problems are formed by adding a term for each equality and inequality constraint, called penalty function, to the objective function. Each penalty function consists of a penalty coefficient multiplied by a measure of violation of each constraint. The violation measure is non-zero in regions where the constraint is violated and is zero where the constraint is met. By iteratively increasing the penalty coefficients, the solver converges to a solution where hard constraints are eventually satisfied i.e., the maximum stress constraints will not be violated.

Our approach is inspired by the penalty method. Our optimization procedure is demonstrated with the schematic diagram of Figure 6. Given an input mesh, our algorithm iteratively updates the shape along with its deformation due to the external forces and boundary conditions. Specifically, a linear system is used to update the shape to satisfy all the geometric and simulation equality constraints under our objective function. The system has the form of a

KKT system - in the optimization literature, KKT denotes the first-order necessary Karush-Kuhn-Tucker conditions that are necessary to satisfy for the solution to be optimal. For each shape update, its deformation is also computed from the KKT system such that the deformed state reaches the force equilibrium. After each solution of the KKT system, we check if the inequality constraints are satisfied i.e., if the stress at each vertex is below the material's yield strength. If the inequality constraints are not satisfied, we increase a penalty term used in our objective function to handle the inequality constraints for the stress (inspired by the penalty methods used in optimization [McC71]). The KKT system is re-solved under the updated objective function. We repeat the procedure until our method converges to an output shape where the stress at each point is below the required threshold, thus all constraints are satisfied.

Mathematically, our method reformulates the equality- and inequality- constrained optimization problem, presented in Section 3, as a sequence of equality-constrained problems as follows:

$$\arg\min_{\mathbf{X}} D(\mathbf{X},\mathbf{X}^0) + \delta \cdot \sum_t h(C - \hat{\sigma}_t(\mathbf{X},\mathbf{P})) \quad \text{(New objective function)}$$

subject to:

$$\forall v \in B\text{: } \mathbf{x}_v = \mathbf{p}_v, \quad \forall v \notin B\text{: } \mathbf{f}_v(\mathbf{X},\mathbf{P}) = \mathbf{0} \quad \text{(Simulation constraints)}$$

$$g(\mathbf{X},\mathbf{X}^0) = 0 \quad \text{(Geometric constraints)}$$

where $h(\cdot) = \min(0,\cdot)^2$ is the penalty function that measures the degree of violation in the inequality constraints, and δ is a penalty coefficient. Starting from a small value ($\delta = 0.01$), we solve the above optimization problem and check whether our inequality constraints are satisfied. If they are not satisfied, the penalty coefficient is increased by a factor of 2, and we solve the problem again starting with the solution found in the previous iteration. Solutions of the successive optimization problems will eventually converge to the solution of the original problem. Practically, in our experiments, this happens when δ reaches the value of 100 at most.

In each iteration, we still need to still solve the equality-constrained optimization problem. Although we could use a penalty function (e.g. a quadratic function) again for those, thus convert the problem to a completely unconstrained one (as done in the original penalty method), this approach did not work. The reason was that the force equilibrium constraints could not be satisfied without causing the optimization to become ill-conditioned due to extremely large penalty coefficients. An augmented Lagrangian approach [BM08] could deal with this problem, but at a higher computational cost. We instead resort to a simpler approach, by solving the equality-constrained optimization problem using the Newton's approach [Goo85]. Mathematically, it is represented as (this is also known as the KKT system):

$$\begin{bmatrix} \mathbf{H} & \mathbf{J}^T \\ \mathbf{J} & 0 \end{bmatrix} \begin{bmatrix} \{\Delta\mathbf{X},\Delta\mathbf{P}\} \\ \mathbf{w} \end{bmatrix} = - \begin{bmatrix} \mathbf{g} \\ \mathbf{b} \end{bmatrix} \tag{17}$$

where $\mathbf{H}$ and $\mathbf{g}$ are the Hessian and gradient of the new objective function respectively, $\mathbf{J}$ is the Jacobian of the equality constraints, and $\mathbf{b}$ represents the function values of the constraints. We solve the above system for multiple iterations until convergence: at each iteration, the shape $\mathbf{X}$ is updated by $t\Delta\mathbf{X}$, and similarly its deformed version is also updated by $t\Delta\mathbf{P}$, where t is a step size determined by a bisection line search. In practice, to ensure that the deformed state of an optimized shape reaches the force equilibrium (i.e., satisfy the simulation constraints), for each update in the shape $\mathbf{X}$, we perform several more updates to its deformed state $\mathbf{P}$ until the force equilibrium is reached (sum of internal and external forces per vertex is practically zero). Note that the system is sparse since each vertex is only affected by its local neighbors. To this end, we apply the sparse LU decomposition algorithm available in the Eigen library [Eig10]. We also refer the reader to our source code for our implementation.

5 Results

To understand how our algorithm works, we begin by using a simple rectangular board as the testing shape, as shown in Fig. 7. Here the two ends of the board are fixed at the bottom (indicated by blue color), and hence they are set as fixed boundaries to the algorithm. We note that setting boundaries as hard constraints in this example will cause a singularity, since in theory the stress on fixed points becomes unbounded. To avoid this singularity, we relax the boundary constraints to soft constraints. Whenever the boundary vertices deviate from their rest states, a spring force will pull them back. The external force is applied in the middle region (indicated by red color). Under these settings, we ran our algorithm and the results are shown in Fig. 8. Fig. 8a shows the shape before optimization, and Fig. 8b shows the optimized shape. Fig. 8c and 8d visualize the von Mises stress distribution before and after optimization. Red color corresponds to high stress and blue corresponds to low stress. Our algorithm automatically strengthens the input shape in areas with high von Mises stress, and the result has significantly reduced stress. Quantitatively, the optimized shape can withstand twice as much (i.e. 100% more) force/weight as the original shape.

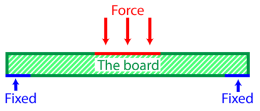
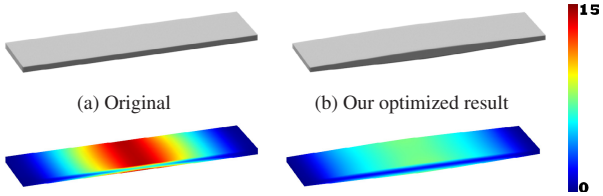


Figure 7: A simple test shape



(a) Original (b) Our optimized result

(c) von Mises Stress of (a) (d) von Mises Stress of (b) (e)

Figure 8: Results of the test shape. Right: color coding of von Mises stress.

Fig. 9 shows a gallery of experimental results including furniture and utility tools (see also accompanying video). For each input shape we show the region and direction of the external force, and the surface/boundary that's fixed. We find the maximum force each shape can withstand (i.e. when the von Mises stress reaches the material limit). We then increase that force by 50%, 100%, and 200% respectively, and apply our algorithm to optimize the shape until it can withstand that much additional force. For each optimized shape, we show a rendering of the shape, a silhouette image that compares the difference between the optimized shape with the original shape, and two color-coded images indicating the displacement of vertices after optimization (i.e. how far each vertex has moved before and after optimization), and the von Mises stress distribution. Observe that as we increase the external force, the optimized shape changes more significantly, although remaining similar to the original shape.

We note that alternative methods such as adding struts or rib-like structures, smoothing creases, thickening corners, changing topology, can potentially also remedy the weak spots for some of the

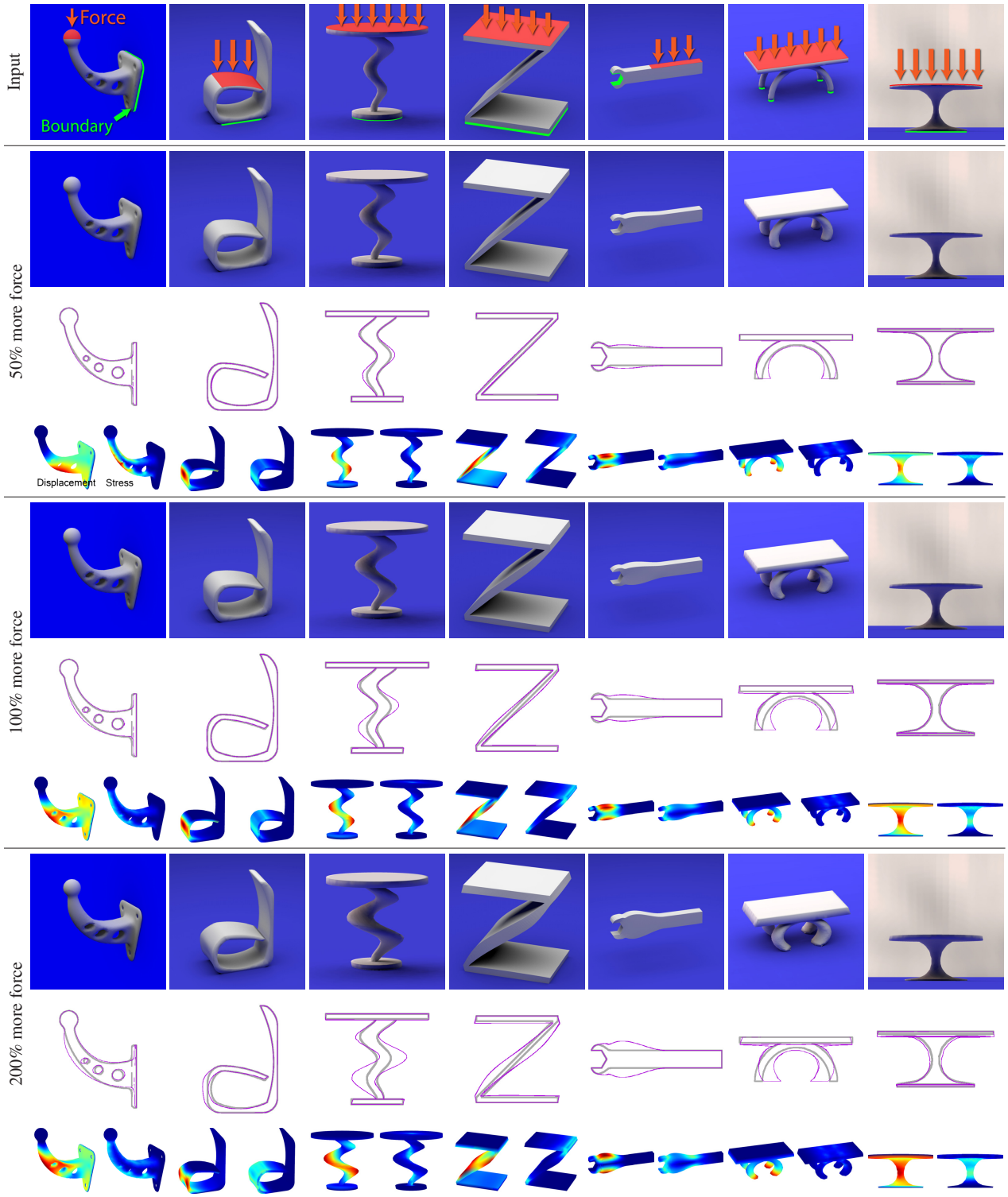


Figure 9: A gallery of experimental results. The first row shows the original shapes and region/direction that external forces are applied. We find the maximum force each original shape can withstand, and apply our algorithm to optimize the shape until it can withstand 50%, 100%, and 200% additional force respectively. The results are shown in the next three rows. Each result contains a rendered image of the optimized shape, a silhouette image overlaying the optimized shape on the input to show the deformation, and the color encoding of the vertex displacement (i.e. how far the vertex has moved before and after optimization) as well as von Mises stress distribution over the surface.

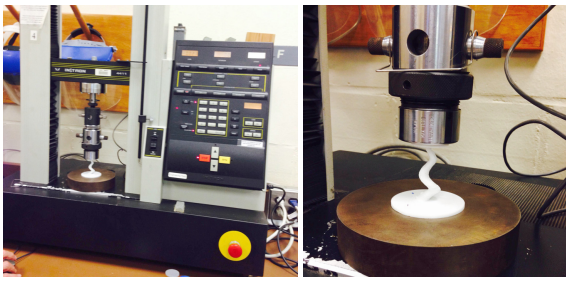


Figure 10: Photographs showing the Instron Universal Testing machine, which applies increasing axial force on the shape until it breaks.

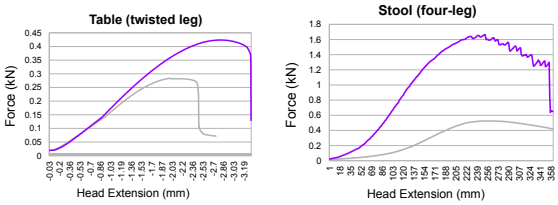


Figure 11: Physics validation results for the twist-leg table and the four-leg stool models. In each graph, the x-axis is the head extension in millimeter (i.e. the distance that the load cell has moved); the y-axis is the measured force in kilonewton. The original shape is plotted in gray, and optimized shape plotted in purple. The point where the curve reaches the bottom is where it breaks under the corresponding force.

presented shapes. However, such solutions would tend to alter the aesthetics of the original designs. In general, preserving geometric features, symmetry, sharp edges and local surface details are all major requirements in shape modeling. Our method preserves such geometric features as well as meet the physical requirements by incorporating terms and constraints that satisfy both. For example, in the case of the z-table and the wrench, our solution successfully preserves their sharp edges due to Laplacian similarity constraints. In other examples, such as the utility hook and the d-shaped chair, our method discovers more global deformations, which cannot be achieved through local thickening.

Physical validation. We perform physical validations by using a professional force gauge (Instron Universal Testing machine, Figure 10), which applies increasing axial force on the shape until it breaks. We have printed each original shape and the optimized result using a 3D printer (CraftBot) with PLA material, 100% infill, and 100 micron resolution.

Figure 11 shows our test results for two sets of models: the twist-leg table, and the four-leg stool. We use our algorithm to compute an optimized shape that can withstand twice (100% more) force than the maximum force allowed on the original shape. For each example we plot the force measured on the original shape (gray curve) and that on the optimized shape (purple curve). The point where each curve reaches the bottom is where the model breaks.

From the measured data, for the four-leg stool model, the original shape breaks at $523N$ and the optimized shape breaks at $1605N$. Thus, the optimized shape can withstand 207% more force. This is considerably better than simulation, which predicts 100% more force. For the twist-leg table, the original shape breaks at $282N$, while the optimized shape breaks at $425N$. Thus, the optimized shape can withstand 50.7% more force than the original shape, which is worse than simulation. The mismatch between actual validation and simulation exists due to a variety of reasons, such as normalization of volume (which increases the maximum force that the original shape can withstand), printing quality, theoretical vs. practical material parameters. For example, our theory assumes isotropic material, but in practice we used extrusion-based 3D printer which leads to anisotropic prints. We believe this way the results are more representative for consumer prints since extrusion-based printers are widely available. In addition, plastic deformations are not considered by our formulation. Note that the curves in the plot indicate some amount of plastic deformations, yet they have predominant linear regions and the breaking points are well under 2-3mm head extension while the models are generally 100mm in height. Thus, we believe the deformations are dominated by the linear elastic model. Despite the mismatches, the physical validation shows that our optimization is still effective, since it manages to significantly strengthen the input shapes in both cases.

Timings. The running time for shape optimization in typical force scenarios ranges from 2 hours for our model with fewest number of vertices ($1K$ vertices) to 6 hours for our largest model ($3K$ vertices). The complexity scales linearly w.r.t. the number of non-zeros, which in turn depend linearly on the number of tetrahedra vertices. Our implementation is currently far from optimal, and we believe that the running times can be significantly improved with a GPU-based implementation to solve the KKT system.

Implementation and source code. Our method is implemented in C++. For uniform tetrahedralization, we used the Trelis software [tre]. Material parameters are included in the configuration files provided with our source code in the supplementary material.

6 Limitations and Future Work

In this paper we introduced an algorithm to directly optimize a 3D shape with the purpose of making it withstand larger external forces. The algorithm incorporates physics simulation and geometric constraints within a unified optimization framework. Experiments demonstrate that our algorithm can produce optimized shapes able to withstand larger forces, while they remain perceptually similar to the original shapes.

Our method based on its current implementation is slow, but we believe that there is significant room for improving its performance. The main bottleneck of our system is solving the KKT system through a sparse linear system solver. A GPU-based sparse solver and a multi-resolution approach would significantly improve the running times of our method. Investigating other numerical optimization strategies would also be useful to improve convergence and stability. Another limitation of our method is that tetrahedra might become inverted in non-convex and highly thin shape parts during the optimization. In our experiments we rarely encountered this problem because our objective function tends to prevent aggressive shape changes in these areas. During optimization, some finite elements can become highly stretched, which will in turn cause the computation of stresses to become inaccurate. In the case of symmetric shapes, our method requires that their input meshing is also symmetric, otherwise the symmetry constraints will not be enforced. We believe that a remeshing strategy along with mesh symmetrization would deal with the above problems. Another future extension of our method could be to use higher-order Laplacians [BS08] in the intrinsic distance term to further promote surface feature preservation. Finally, it would be interesting to extend our algorithm to other domains, which involves other design goals, such as improving the aerodynamics properties of printable shapes.

Acknowledgements. We thank the reviewers for their comments. This work was supported by NSF grants CHS-1422441, CHS-1617333, and IIS-1423082.

References

[AJ08] ALLAIRE G., JOUVE F.: Minimum stress optimal design with the level set method. *Engineering Analysis with Boundary Elements 32*, 11 (2008). 1, 2

[AJT02] ALLAIRE G., JOUVE F., TOADER A.-M.: A level-set method for shape optimization. *Comptes Rendus Mathematique 334*, 12 (2002). 1, 2

[All12] ALLAIRE G.: *Shape optimization by the homogenization method*, vol. 146. Springer Science & Business Media, 2012. 2

[BCC*10] BAZILEVS Y., CALO V., COTTRELL J., EVANS J., HUGHES T., LIPTON S., SCOTT M., SEDERBERG T.: Isogeometric analysis using t-splines. *Comput. Methods Appl. Mech. Eng. 199*, 5-8 (2010). 1, 2

[BF84] BRAIBANT V., FLEURY C.: Shape optimal design using b-splines. *Comput. Methods Appl. Mech. Eng. 44*, 3 (1984). 1

[BM08] BIRGIN E. G., MARTÍNEZ J. M.: Improving ultimate convergence of an augmented lagrangian method. *Optimization Methods Software 23*, 2 (2008). 6, 7

[BR88] BELEGUNDU A. D., RAJAN S. D.: A shape optimization approach based on natural design variables and shape functions. *Comput. Methods Appl. Mech. Eng. 66*, 1 (1988). 1, 2

[BRC16] BANDARA K., RÜBERG T., CIRAK F.: Shape optimisation with multiresolution subdivision surfaces and immersed finite elements. *Comput. Methods Appl. Mech. Eng. 300* (2016). 2

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Trans. Vis. Comp. Graph. 14*, 1 (2008). 9

[BS13] BENDSOE M. P., SIGMUND O.: *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013. 2

[BWBSH14] BÄCHER M., WHITING E., BICKEL B., SORKINE-HORNUNG O.: Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph. 33*, 4 (2014). 2

[CZXZ14] CHEN X., ZHENG C., XU W., ZHOU K.: An asymptotic numerical method for inverse elastic shape design. *ACM Trans. Graph. 33*, 4 (2014). 2

[DMLK13] DIJK N. P., MAUTE K., LANGELAAR M., KEULEN F.: Level-set methods for structural topology optimization: A review. *Struct. Multidiscip. Optim. 48*, 3 (2013). 1, 2

[Eig10] EIGEN:, 2010. URL: http://eigen.tuxfamily.org. 7

[FAT07] FU H., AU O. K.-C., TAI C.-L.: Effective derivation of similarity transformations for implicit Laplacian mesh editing. *Comp. Graph. Forum 26*, 1 (2007). 6

[Goo85] GOODMAN J.: Newton's method for constrained optimization. *Mathematical Programming 33*, 2 (1985). 7

[HBA13] HILDEBRAND K., BICKEL B., ALEXA M.: Orthogonal slicing for additive manufacturing. *Comp. and Graph. 37*, 6 (2013). 2

[HJW15] HU K., JIN S., WANG C. C.: Support slimming for single material based additive manufacturing. *Comp. Aided Design 65* (2015). 2

[HM03] HASLINGER J., MAKINEN R. A. E.: *Introduction to Shape Optimization: Theory, Approximation, and Computation*. Society for Industrial and Applied Mathematics, 2003. 1, 2

[HSB14] HOJJAT M., STAVROPOULOU E., BLETZINGER K.-U.: The vertex morphing method for node-based shape optimization. *Comput. Methods Appl. Mech. Eng. 268* (2014). 3

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc. SCA* (2004). 4

[LBT11] LE C., BRUNS T., TORTORELLI D.: A gradient-based, parameter-free approach to shape optimization. *Comput. Methods Appl. Mech. Eng. 200*, 9-12 (2011). 3

[LSZ*14] LU L., SHARF A., ZHAO H., WEI Y., FAN Q., CHEN X., SAVOYE Y., TU C., COHEN-OR D., CHEN B.: Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph. 33*, 4 (2014). 1, 2

[MAB*15] MUSIALSKI P., AUZINGER T., BIRSAK M., WIMMER M., KOBBELT L.: Reduced-order shape optimization using offset surfaces. *ACM Trans. Graph. 34*, 4 (2015). 2

[McC71] MCCORMICK G.: Penalty function versus non-penalty function methods for constrained nonlinear programming problems. *Mathematical Programming 1*, 1 (1971). 6, 7

[MDLW15] MARTÍNEZ J., DUMAS J., LEFEBVRE S., WEI L.-Y.: Structure and appearance optimization for controllable shape design. *ACM Trans. Graph. 34*, 6 (2015). 3

[MDSB03] MEYER M., DESBRUN M., SCHRÃŰDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III* (2003), Springer-Verlag. 5

[MGP06] MITRA N., GUIBAS L., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph. 25*, 3 (2006). 5

[Mis13] MISES R. V.: Mechanik der festen korper im plastisch deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Gottingen 1913*, 1 (1913). 3, 4

[MSS05] MESKE R., SAUTER J., SCHNACK E.: Nonparametric gradient-less shape optimization for real-world applications. *Structural and Multidisciplinary Optimization 30*, 3 (2005). 3

[NW06] NOCEDAL J., WRIGHT S. J.: *Numerical Optimization*. Springer, 2006. 6

[PWLSH13] PRÉVOST R., WHITING E., LEFEBVRE S., SORKINE-HORNUNG O.: Make It Stand: Balancing shapes for 3D fabrication. *ACM Trans. Graph. 32*, 4 (2013). 2

[PZM*15] PANETTA J., ZHOU Q., MALOMO L., PIETRONI N., CIGNONI P., ZORIN D.: Elastic textures for additive fabrication. *ACM Trans. Graph. 34*, 4 (2015). 2

[RG92] ROSEN D., GROSSE I.: A feature based shape optimization technique for the configuration and parametric design of flat plates. *Engineering with Computers 8*, 2 (1992). 1, 2

[RH08] RONALD HUSTON H. J.: *Practical Stress Analysis in Engineering Design, Third Edition*. CRC Press, 2008. 4

[SBR*15] SCHUMACHER C., BICKEL B., RYS J., MARSCHNER S., DARAIO C., GROSS M.: Microstructures to control elasticity in 3d printing. *ACM Trans. Graph. 34*, 4 (2015). 2

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proc. SGP* (2004). 5, 6

[Sor05] SORKINE O.: Laplacian Mesh Processing. In *Eurographics, STAR Reports* (2005). 5

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: Improving structural strength of 3d printable objects. *ACM Trans. Graph. 31*, 4 (2012). 1, 2

[Tor93] TORTORELLI D. A.: A geometric representation scheme suitable for shape optimization. *Mechanics of Structures and Machines 21*, 1 (1993). 1, 2

[tre] Trelis. URL: http://www.csimsoft.com/. 9

[US13] UMETANI N., SCHMIDT R.: Cross-sectional structural analysis for 3d printing optimization. In *SIGGRAPH Asia Technical Briefs* (2013). 1, 2

[VGB*14] VANEK J., GALICIA J. A. G., BENES B., MÄŻCH R., CARR N., STAVA O., MILLER G. S.: Packmerger: A 3d print volume optimizer. *Comp. Graph. Forum 33*, 6 (2014). 1, 2

[WMC08] WALL W., MORITZ F., CYRON C.: Isogeometric structural shape optimization. *Comput. Methods Appl. Mech. Eng. 197* (2008). 1, 2

[WMKG07] WARDETZKY M., MATHUR S., KÄLBERER F., GRINSPUN E.: Discrete laplace operators: No free lunch. In *Proc. SGP* (2007). 5

[WWY*13] WANG W., WANG T. Y., YANG Z., LIU L., TONG X., TONG W., DENG J., CHEN F., LIU X.: Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. Graph. 32*, 6 (2013). 1, 2

[ZPZ13] ZHOU Q., PANETTA J., ZORIN D.: Worst-case structural analysis. *ACM Trans. Graph. 32*, 4 (2013). 1, 2