Designing Volumetric Truss Structures

RAHUL ARORA, University of Toronto
ALEC JACOBSON, University of Toronto
TIMOTHY R. LANGLOIS, Adobe Research
YIJIANG HUANG, MIT
CAITLIN MUELLER, MIT
WOJCIECH MATUSIK, MIT CSAIL
ARIEL SHAMIR, The Interdisciplinary Center
KARAN SINGH, University of Toronto
DAVID I.W. LEVIN, University of Toronto

Fig. 1. We generate stress-aligned 3D trusses which are structurally-sound and lightweight. The structures produced by our method consist of families of smooth, continuous curves tracing stress lines. Representative examples demonstrating the versatility of our approach (from left to right): the Stanford bunny, the rear pylon of a single engine helicopter, a bridge made of wooden beams and a simulation showing that a miniature (20cm wide) plastic bridge could support a 93 kg (205 lbs) person.

We present the first algorithm for designing volumetric Michell Trusses. Our method uses a parametrization approach to generate trusses made of structural elements aligned with the primary direction of an object's stress field. Such trusses exhibit high strength-to-weight ratios. We demonstrate the structural robustness of our designs via a posteriori physical simulation. We believe our algorithm serves as an important complement to existing structural optimization tools and as a novel standalone design tool itself.

CCS Concepts: •**General and reference** → **Design;** •**Computing methodologies** → **Physical simulation;** *Mesh models;*

Additional Key Words and Phrases: curve networks, design, simulation, topology optimization

1 INTRODUCTION

It is sometimes said that the primary objective of engineering is to develop the stiffest possible structure by using the least amount of material [Doubrovski et al. 2011]. This guiding principle can be seen in many everyday structures such as bridges and stadiums. Strength-to-weight trade-off is naturally expressed as an optimization problem and its solution has become a foundational challenge in mathematics, computer science and engineering.

Almost all structural optimization algorithms discretize the material distribution within the structure and then attempt to sparsify this distribution (see Figure 3). The nature of this discretization, be it voxels, level-sets or trusses, gives birth to the specific optimization and algorithm applied.

Unfortunately, all of these methods have inherent limitations, rooted in the requirement of an overprescribed set of design variables (either voxels or bars) as initialization. The voxel grids of traditional Topology Optimization often require the use of additional regularization terms in the optimization objective in order to avoid "checker-boarding" artifacts, level-sets can require additional foliation terms to generate topology change and truss-based methods require an appropriate set of input trusses be specified.

Even when such difficulties can be overcome, these methods (by construction) lack any global notion of object topology or geometry. This can have implications down stream during the design process wherein an architect or engineer may wish to make small changes to the design, such as constraining certain points, deleting structural members or consistently resizing elements. The importance of these operations, coupled with the difficulty in performing them on standard topology optimized output, has led practitioners to employ frustratingly manual solutions such as tracing over optimized results to produce a final fabricable object (Figure 2).

In this paper, we take a different approach to the generation of light, and strong structures. Instead of starting with an overprescribed solution and sparsifying it, we use one-dimensional cylindrical structural elements to define a truss, and formulate its design as a fitting problem for these elements. Michell [Michell 1904] laid the foundations for creating such trusses by proving that for a given material budget, all elements of the optimal (stiffest) truss must follow

2016. XXXX-XXXX/2016/1-ART1 $15.00
DOI: 0000001.0000001_2

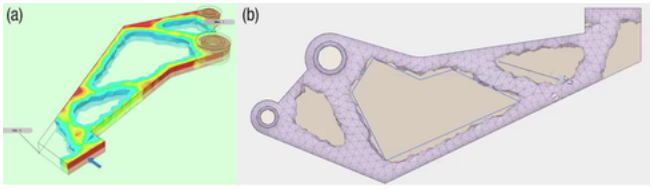


Fig. 2. Topology optimization results can be challenging to fabricate using non-additive manufacturing techniques. Even when using state of the art commercial tools for topology optimization such as Fusion 360 [Autodesk 2018] (a), users have to manually trace over the results (b) to produce a fabricable geometry. © Autodesk Sustainability Workshop https://youtu.be/lyTULzvHhXw.

paths of maximum strain. Structures which fulfill this property are called *Michell Trusses*. Hence, by aligning the individual elements with the principal stress directions of an object's stress tensor field, a structurally sound design can be created without needing to fill the entire shape volume with material (and later sparsifying it).

Michell Trusses can be difficult to design for all but the simplest geometries and loading conditions. While a number of recent works have shown how to computationally design Michell Trusses for 2D domains or on height-fields, an algorithm for generating Michell Trusses for arbitrary loads acting on three dimensional shapes has remained elusive until now.

We present the first algorithm to design truss structures following Michell Truss principles inside arbitrary 3D domains. Rather than optimizing an initial guess, we treat truss optimization as a fitting problem. Our method requires only a single solve of the static equilibrium equations to compute a continuous stress field. We then use a novel parametrization method to produce a graph of a prescribed resolution where each graph edge is as aligned as possible with the underlying stress tensor field. Our method avoids many of the difficulties of previous methods, its initialization is trivial, and we require no additional regularization terms to avoid high-frequency artifacts in our results.

Contributions. To summarize, the main contribution of our work is the first algorithm for generating 3D Michell Truss structures on complex 3D geometry. In realization of this overarching goal, we make the following three technical contributions:

- A method for extracting a volumetric, stress-aligned frame field.
- A method for generating a volumetric texture parametrization with coordinate lines aligned with the frame field.
- A method for extracting the truss structure from a volumetric texture.

We show results on various 2D and 3D examples and demonstrate the high strength-to-weight ratio we achieve compared to naïve truss layouts.

2 RELATED WORK

Structural optimization is a classic problem in computational design, fabrication and digital manufacturing. Methods exist to help designers identify the absolute weakest parts of objects [Zhou et al. 2013] or the weakest parts under real world forces [Langlois et al. 2016]. Other methods attempt to reinforce designs to improve their strength [Stava et al. 2012] or find the most stable orientation for 3D printing a design [Umetani and Schmidt 2013].

In this paper, we focus on the problem of generating structurally sound objects via optimal material placement. Algorithms for this task define optimality using some measure of an object's strength—most often attempting to minimize an object's compliance under a given load [Bendsøe and Sigmund 2009; Freund 2004] while satisfying constraints on the amount of material used.

2.1 Voxel and Level-Set Optimization

Algorithms for structural optimization can be differentiated based on their chosen discretization for the material distribution. One oft-used material discretization is a dense voxel grid (though other mesh structures can be used [Gain et al. 2015; Ha and Cho 2008]). This approach has recently been used to create extremely detailed designs such as bone-like structures [Wu et al. 2017] and even airplane wings [Aage et al. 2017]. Variations on a theme include using image slice stacks to generate infill for 3D prints [Mao et al. 2018]. While capable of generating a wide range of stable designs, these methods can be difficult to control, requiring regularization to avoid non-physical "checkerboarding" artifacts (high frequency patterns of solid and empty voxels) and disconnected components (which make the designs un-manufacturable) [Schumacher et al. 2015]. They also require a choice of density cutoff which determines when a cell is considered empty or not. Voxel-based topology optimization is also unsuitable for many mission-critical engineering applications as it can create internal cavities and tiny internal features making surface inspection impossible [Todorov et al. 2014; Waller et al. 2014].

Level-sets have recently become a popular material distribution representation, in part because they help avoid some of the artifacts of a discrete voxel representation [Wang et al. 2003]. Level-sets can represent smoother geometry and their use mitigates "checkerboarding". However, such methods must rely on foliation terms to instigate topology change with the final outcome depending on the topology of the initial solution [Allaire et al. 2004].

There have also been efforts to improve the performance of these types of high resolution material optimization methods. For instance, by assuming that the outer shape is fixed, Ulu et al. [2017] show how to leverage model reduction to speed up internal structure optimization by reducing the number of variables.

2.2 Truss Optimization

Truss-based optimization methods [Bendsøe et al. 1994; Freund 2004] are attractive for their small number of design variables (compared to voxel-based methods) and ease of manufacturing. Michell [1904] first discovered that an optimal truss layout (in terms of strength-to-weight ratio) is given when trusses are aligned with the principal stress directions induced by loading conditions. Intuitively, this aligns elements with the directions of pure compression and tension minimizing stress due to bending. In certain cases, it is possible to solve for this optimal layout analytically [Jacot and Mueller 2017], but no analytical solution is known for the general case, so the ground structure method (GSM) [Dorn et al. 1964; Zegard and Paulino 2015] is used. Here, an initial layout of a finite number of trusses is specified, and the radii and connectivity of the

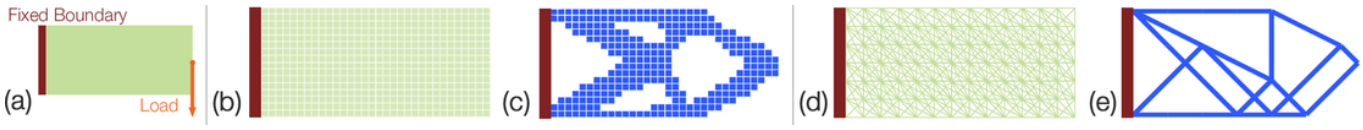


Fig. 3. Optimization of a cantilever beam (a) using voxel-based continuum-method (c) and a ground structure method (e). Voxel-based continuum-methods optimize for material placement (c) on the voxelized domain (b), while the ground structure method over samples the domain with truss members (d) and solves for the optimum set of members (e).

trusses are optimized to minimize the total weight. The traditional GSM formulation suffers from several problems: (1) an initial layout of node positions needs to be specified, which can limit the solution space; (2) it can yield self-intersecting beams; (3) it assumes that the cross-section of each truss member can be set independently, which makes large-scale manufacturing challenging.

Multiple heuristic methods such as particle swarm optimization [Li et al. 2009], ant colony optimization [Kaveh et al. 2008], teaching-learning-based optimization [Camp and Farshchin 2014], and genetic algorithms [Kawamura et al. 2002] have been proposed to address the third problem by limiting the cross-sections of the trusses to a small set. Mixed-integer programming techniques have been used to achieve global optima for this problem [Achtziger and Stolpe 2007; Rasmussen and Stolpe 2008; Stolpe and Svanberg 2003]. However, these methods were only demonstrated on small models. Jiang et al. [2017] recently demonstrated much larger examples by dividing the mixed-integer problem into three subproblems that were solved iteratively. This works well in practice, but does not guarantee a globally optimal solution. The method also optimizes initial node positions and connectivity to avoid self-intersections, but still requires an oversampled initial mesh, the design of which remains challenging.

2.3 Stochastic and Spectral Approaches

Martinez et al. [2017] generate procedural anisotropic "foams" by warping the local distance metric. They show that by controlling this warping they can generate 3D printable metamaterials with anisotropic mechanical properties, and that they can align their metamaterials with the stress field resulting from a 2D topology optimized structure. In contrast to this approach, our method operates on a larger, macroscopic scale, does not require an initial topology optimization pass, and provides us with much greater control of the resolution of the produced structure. We also show that our method can produce fabricable output from 3D stress fields, not just 2D. Nguyen et al. [Nguyen et al. 2012] generate a conformal cell structure from a predefined set of cells. However, the radii of trusses can still vary uniquely, which makes manufacturing difficult. Wang et al. [2013] generate skin-frame structures with a solid outer shell and strut filled interior to reinforce objects for 3D printing. The method requires an initial internal sampling of nodes; struts are filled in with an ANN method, and an ℓ_0 sparsity optimization is used to minimize the number of struts.

There have also been more user-centric algorithms proposed. For instance Zehnder et al. [2016] propose an interactive design tool for constructing ornamental curve networks on surfaces. They use a spectral approach to determine structural stability, ensuring the design has no low energy deformation modes.

2.4 Michell Trusses

Though some questions have been raised regarding the optimality of Michell Trusses [Sigmund et al. 2016], they still play an important role in engineering design. While the methods above try to approximate a truss layout via optimization, [Bendsøe et al. 1994], other methods more directly attempt to generate Michell layouts. Tam et al. [Tam et al. 2015; Tam and Mueller 2017] generate principal stress lines directly by integrating the stress field, and develop a novel robotic arm printer capable of printing along these lines directly. However, currently the method only applies to 2.5D structures (i.e., structures that are 3D but only need one layer of trusses, such as shells or membranes), and cannot handle 3D volumetric cases. Li and Chen [2010] begin with a (very simple) user provided beam network which connects the contacts to the points of application of external forces. Then, an iterative method subdivides this structure and better approximates principal stress lines until the desired compliance/strain energy is achieved. While motivational, this method only works for 2D structures. It is also unclear if the user interaction is amenable to more complicated shapes or contact/load configurations. Li et al. [2017] produce rib like reinforcements aligned with principle stress directions but again, only for 2.5D structures.

2.5 Parameterization-Based Mesh Generation

Our method replaces structural optimization with automatic mesh generation and provides the first (to the authors' knowledge) algorithm for computing Michell Trusses in arbitrary 3D shaped domains under arbitrary loading conditions.

Our method is inspired by recent developments in hex and quad meshing for 3D geometries (for instance [Nieser et al. 2011; Panozzo et al. 2014]). These algorithms use prescribed frame fields to align the gradients of a volumetric function such that a hex mesh can be extracted. The general hex meshing problem is hard and still an active area of research. None of the currently available algorithms satisfy the criteria necessary for solving our particular problem.

The seminal paper, CubeCover [Nieser et al. 2011], solves a generalized version of the parametrization and mesh extraction problem we solve. However, their method must introduce discrete optimization variables in order to compute a well aligned frame field. We confirmed via personal communication that CubeCover does not generate approriate frame fields and thus "it's impossible for the software in its current state to be used as stand-alone solution" [Nieser 2018]. Ray et al. [2016] and Solomon et al. [2017] tackle the issue of frame field generation by introducing functional representations of

frames. Ray et al. align their frame field with a mesh boundary and smoothly interpolate into the object volume. Solomon et al. also smoothly interpolate inside the object volume using a boundary element based approach. However, our problem requires the opposite objective as we care little for boundary alignment and much more for accurate alignment within the mesh volume itself. L_p-centroidal voronoi tessellation [Lévy and Liu 2010] can generate hex-dominant meshes that take a background anisotropy field into account. However, it does not follow the anisotropy as closely as our method does. Lyon et al. [2016] present a method for mesh extraction; that is, given a parametrization on a tetrahedral mesh, they extract out a 3D-embedded graph. However, their method requires that the input parametrization is boundary-aligned. Again, our method requires good alignment in the interior of the object, not the boundary, making this method unsuitable. Many of our results such as a bridge or beam (Figs. 15, 10) are naturally stronger when trusses are not normal to the boundary. Unlike all of the methods above, ours is the first to generate global, structurally sound parameterizations.

3 BACKGROUND AND PRELIMINARIES

We begin by introducing the technical background necessary to understand our formulation, starting with an introduction to truss optimization.

3.1 Truss Optimization

A *truss* is a structure consisting of a network of members, each of which is under purely axial stress. Typically, the forces only act at the joints between these members, known as the *nodes* of a truss. Given a domain $\Omega \subset \mathbb{R}^d$, and boundary conditions consisting of a set of static forces applied on the boundary and anchoring parts of the boundary to fixed supports, truss optimization is the problem of finding a structurally sound truss minimizing the volume of material utilized. In general, this involves optimizing over three design parameters:

(1) the *topology*, which specifies the connectivity of the truss members;
(2) the *geometry*, specifying the positions of the nodal points; and
(3) the *size*, which gives the cross-sectional area of the members.

We use the term *truss layout* to refer to the choice of topology and geometry of the truss. Mathematically, this is equivalent to a graph embedded in Ω.

In the classical truss optimization formulation, known as the Ground Structure method, an a priori chosen set of uniformly spaced nodal points and members cover the problem domain Ω, forming the so called *ground structure*. The topology of the optimal truss is generated by varying the cross-sectional areas of the members, allowing for zero areas which effectively removes those members. Since the nodal points are assumed to be fixed, the classical approach only solves for the topology and size parameters.

3.2 Michell Truss Theory

Michell's theorem Michell [1904] characterizes the fundamental properties of the optimal truss structure, called the Michell truss, for the problem defined above. The theorem states that the members of the optimal truss structure lie along the *principal directions* of the *virtual* stress field. The virtual stress field is defined as the stress induced by the given external forces if the domain were to be uniformly filled with material, and principal stress directions simply refer to the eigenvectors of the stress tensor matrix. Owing to the continuity of the stress tensor field and the orthogonality of eigenvectors of a Hermitian matrix, the principal stress directions form a set of families of orthogonal curves called the principal stress lines. In the continuous setting, an optimal Michell Truss consists of an infinite set of infinitesimally small members tracing these curves. Computationally, the optimal truss layout for the chosen discretization consists of finite sized members approximating the principal stress lines (see Figure 4).

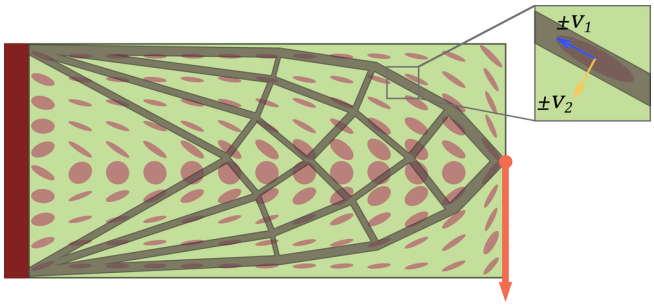


Fig. 4. Michell Truss members are aligned with the principal directions of the underlying stress field. Here, we show the example of a cantilever. The ellipses visualize the stress tensor, and the optimal Michell truss for a chosen discretization is overlaid on the problem domain. Inset: Michell truss member aligned with a stress eigenvector.

In the Ground Structure method, the approximation error is determined by the density and connectivity of the initially chosen ground structure. Unfortunately, it is difficult to choose an appropriate discretization balancing accuracy and computational time for complex domain geometries.

4 STRESS-ALIGNED TRUSS NETWORK GENERATION

We take a parametrization-based approach to generating stress aligned trusses. Our algorithm consists of four independent phases: (1) stress tensor field generation using finite element analysis (FEA), (2) stress-aligned frame field fitting, (3) volumetric texture parametrization, and (4) structural member extraction (Figure 5). Optionally, we can choose to sparsify a previously generated truss layout to reduce material use.

One might ask why follow such an approach given that robust and reliable hex meshing for arbitrary geometries is, as yet, unsolved. Fortunately, our problem is more amenable to solution than that of general hex meshing as we have a volumetric tensor field to guide us. We also do not require hexahedral cells everywhere in our domain, or a boundary aligned structure.

These considerations eliminate some of hex-meshing's most aggravating difficulties, allowing us to develop a flexible algorithm, which, as we will demonstrate, can be applied to a wide variety of geometries. In the remainder of this section, we will detail each step of our truss generation method.

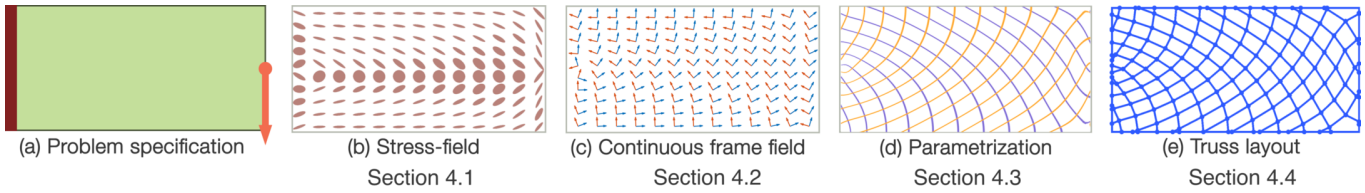


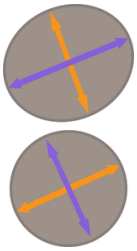
Fig. 5. Overview of our method: (a) starting with a problem domain (green) with fixed points (red) and loads (orange) as boundary conditions, we first perform FEM analysis to compute the stress field (b). A continuous and smooth frame field (c) aligned with the principal stress directions is then computed. The components of this frame field are then used to compute a texture parametrization on the domain (d), whose isocurves (show in orange and purple) are traced to extract a stress-aligned truss structure (e).

4.1 Finite Element Analysis

The first step of our method is to generate a stress tensor field for an input geometry. We use standard linear elastic [Gould and Feng 1994] finite element analysis with tetrahedral discretization [Belytschko et al. 2013; Levin et al. 2017] for this task. Both Dirichlet and Neumann boundary conditions are manually determined based on the expected loading conditions of the given shape. We then compute the Cauchy stress tensor field (for our elements, a single tensor per tetrahedron) for use in subsequent algorithmic stages. We use the same discretization for all steps of the method.

In the remaining sections, we refer to the continuous input domain as $\Omega \subset \mathbb{R}^3$ and the tetrahedral discretization of the domain as the mesh $\mathcal{M} = (\mathcal{V}, \mathcal{T})$.

4.2 Stress Aligned Frame Field Generation



Naïvely, a Cauchy Stress tensor field $\sigma(\mathbf{x})$ can be interpreted as a frame field by representing each tensor by its three eigenvectors. Because each $\sigma(\mathbf{x})$ is a Hermitian matrix, its eigenvectors are guaranteed to form an orthogonal basis. However, such a frame field is almost certain to be non-smooth as the direction of each eigenvector can be arbitrarily flipped or interchanged (see inset). Previous algorithms for frame field generation [Nieser et al. 2011] handle such reflection symmetries by searching over all possible symmetric frame configurations. This is effective but complicates optimization by introducing discrete variables.

Rather than using discrete variables, we are inspired by methods which work with inherently symmetric, functional representations of frame fields [Ray et al. 2016; Solomon et al. 2017]. We are also influenced by Levin et al. [2011] which shows that by using the Rayleigh Quotient [Horn and Johnson 1990] as an objective, one can produce vector fields that smoothly align with the most locally anisotropic direction of a tensor field. The key observation is that the tensor itself is a useful, symmetry agnostic frame field representation and here we leverage this notion and extend it to frame field fitting.

A 3D frame can be encoded using three unit vectors. We define the notion of alignment of a single unit vector with the stress tensor using the square root of the absolute value of the Rayleigh Quotient, and introduce the notation $\|\cdot\|_M$, where $M \in \mathbb{R}^{3\times3}$. Now, for unit vectors $\mathbf{v}$, $\|\mathbf{v}\|_M = (|\mathbf{v}^T M \mathbf{v}|)^{1/2}$ is maximized when $\mathbf{v}$ is aligned with the primary eigenvector of M.

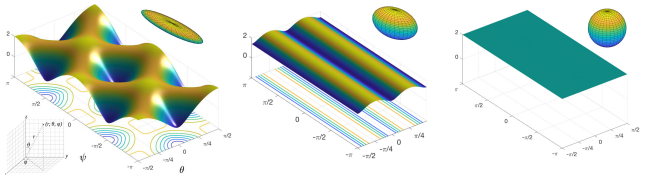


Fig. 6. Our cost function has identical local minima corresponding to any orthogonal transformation that aligns a frame with the second and third eigenvectors of a tensor. This behavior is consistent for tensors with 3 (left) 2 (center) and 1 (right) distinct eigenvalues.

Let the eigendecomposition of σ be given by $\sigma = Q\Lambda Q^T$, where Λ is a diagonal matrix whose diagonal elements are the sorted (in decreasing order) eigenvalues of σ. We define

$$\sigma_+ = Q\lambda Q^T, \tag{1}$$

where the operation $\lambda = (|\Lambda_{ij}|)$ returns a matrix with entries of Λ replaced by their absolute values.

We are now ready to define alignment of frames and stress tensors. We define a "good" fit between a frame and a stress tensor as one where the first axis of the frame is aligned with the primary eigenvector of the stress tensor and the other two axes are aligned with the second and third eigenvectors (though it does not matter which aligns with which). To this end we define the following frame-tensor matching function:

$$E^i_{data}(R = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)) = \|\mathbf{r}_2\|_{\sigma^i_+} + \|\mathbf{r}_3\|_{\sigma^i_+}, \tag{2}$$

where $\sigma^i \in \mathbb{R}^{3\times3}$ is the stress tensor of the i^{th} tetrahedron in our mesh $\mathcal{M}$ (we use the simulation discretization for the fitting stage of our method as well), R is our frame with $\mathbf{r}_j \in \mathbb{R}^3$ its j^{th} direction vector. The positive-definite matrix σ^i_+ is defined according to Equation 1. This cost function has a set of identical minima at every frame alignment which satisfies our criteria (Figure 6). In order to improve the numerical behavior of this energy function, we rescale λ' so that the eigenvalues lie in the range $[1, 30]$ (chosen experimentally). Note that this rescaling alters neither the eigenvectors, nor the ordering or multiplicity of the eigenvalues of M, making $\|\cdot\|_M$ a true norm.

Next we need a method for disambiguating the local minima in Equation 2. Typically this is done combinatorially, but here we follow the approach of Solomon et al. [2017] and instead use a smoothness energy to produce a well-fitted, consistently aligned frame field. Solomon et al. represent rotations using canonical axis

functions and use a standard Laplacian smoothing term. While we borrow their smoothing energy, we cannot use their frame representation as it requires an extra, approximate projection to produce proper, orthogonal frames. In our problem, alignment with the data is critical, and introducing error via such a projection is not acceptable.

Instead we represent a frame at the centroid of a tetrahedron using rotation matrices, parameterized via the matrix exponential, $R^i = \text{expm}\left(\sum_{j=1}^{4} \left[\omega^j\right]\right) \in \mathbb{R}^{3\times3}$. Here, $\omega^j \in \mathbb{R}^3$ are angular velocity vectors at the vertices of each tetrahedron and the $[\cdot]$ operator computes the cross product matrix from an input vector. This allows us to define a smoothness energy in the following manner:

$$E_{smooth}(\omega) = \frac{1}{2}\omega^T L\omega \; + \; \frac{1}{2}\omega^T\omega, \tag{3}$$

where ω is the stacked vector of all per-vertex ω^i's, and L is a block diagonal cotangent Laplacian matrix. The second term regularizes ω and prevents it from taking arbitrarily large values. In practice we found this helped with the stability of the line search of our optimization scheme (fmincon [The MathWorks, Inc. 2017])

Initially we attempted to perform frame fitting using a weighted sum of Equation 2 and Equation 3:

$$E_\alpha(\omega) = \sum_{i=1}^{N} E^i_{data}(\boldsymbol{r}_1(\omega), \boldsymbol{r}_2(\omega), \boldsymbol{r}_3(\omega)) + \alpha E_{smooth}(\omega) \tag{4}$$

$$\omega^* = \underset{\omega}{\text{argmin}} \; E_\alpha(\omega) \tag{5}$$

where $N = |\mathcal{T}|$ is the number of tetrahedra in $\mathcal{M}$, α is a scalar weight, and $R^i = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$. Minimizing this cost function, using an L-BFGS Hessian approximation, revealed issues in choosing an appropriate α. To alleviate this problem, we again lean on the fact that our sole concern is minimizing the data term. The only purpose of the smoothness energy is to help us choose an appropriate local minima to descend into. To this end, our final fitting algorithm (Algorithm 1) is Augmented Lagrangian-esque [Nocedal and Wright 2006] in that we repeatedly minimize Equation 5 with increasingly smaller α until the data cost stops decreasing. In practice our termination criteria was not complex: a fixed thirty iterations after each of which α was reduced to $(2/3)\alpha$. This proved to be more than enough for all our examples and yielded excellent results. There is a minor implementation detail which arises when using matrix exponentials as a parametrization of rotation matrices—their gradient is undefined at $\omega = \mathbf{0}$. We avoid this problem by perturbing any 0 length angular velocity vector by the square root of machine epsilon (a standard work-around). Other terms in the gradient ensure good numerical behavior near the singularity.

4.3 Induced Parametrization Computation

We use our smooth, data-aligned frame field to compute a stress-aligned parametrization from which we will create our Michell Truss. We define $\Omega \subset \mathbb{R}^3$ as the world space that our object occupies and $\mathbf{u} \in \mathbb{R}^3$ as a volumetric texture domain. We chose our structural members to lie along the coordinate lines of $\mathbf{u}$ and seek to find a parametrization $\mathbf{u} = \phi(\mathbf{x}) : \Omega \to \mathbb{R}^3$ that aligns these coordinate

ALGORITHM 1: Iterative method for computing a stress-aligned frame field.

$\omega \leftarrow \mathbf{0}$;
$\alpha \leftarrow 10|\mathcal{T}|$;
repeat
  /* Initialize L-BFGS with previous estimate of ω to solve (5) */
  $\omega \leftarrow$ L-BFGS(E_α, ω);
  $\alpha \leftarrow \frac{2}{3}\alpha$;
until *convergence*;
return ω;

lines with our frame field. Formally we seek a $\phi(\mathbf{x})$ such that

$$\begin{aligned} \frac{\partial\phi}{\partial\mathbf{x}}\mathbf{r_1} &= \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T \\ \frac{\partial\phi}{\partial\mathbf{x}}\mathbf{r_2} &= \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T \\ \frac{\partial\phi}{\partial\mathbf{x}}\mathbf{r_3} &= \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \quad , \end{aligned} \tag{6}$$

at the center of each tetrahedron in our mesh.

We can write these requirements as a linear system of equations by constructing the discrete directional gradient operator for each tetrahedron in our mesh:

$$G^i(\mathbf{v}) = \left[\mathbf{v}^i_x \cdot G^i_x + \mathbf{v}^i_y \cdot G^i_y + \mathbf{v}^i_z \cdot G^i_z\right], \tag{7}$$

where G_x, G_y and G_z are the discrete gradient operators of our tetrahedral mesh, $\mathbf{v} \in \mathbb{R}^3$ is the direction in which the derivative is to be measured (at the centroid of a tetrahedron) and i indexes our tetrahedra. We can assemble these local directional derivative operators into global matrices to produce the global operator $G(\mathbf{v})$.

We proceed by constructing three directional derivative operators, one for each frame director

$$\begin{aligned} \mathbf{G}_1 &= G(\mathbf{r_1}) \\ \mathbf{G}_2 &= G(\mathbf{r_2}) \\ \mathbf{G}_3 &= G(\mathbf{r_3}) \quad . \end{aligned} \tag{8}$$

The discrete version of Equation 6 can then be stated as the following constrained minimization problem.

$$\phi^* = \underset{\phi}{\text{argmin}} \left\| \begin{bmatrix} \mathbf{G}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_3 \end{bmatrix} \phi - \begin{bmatrix}\mathbf{1}\\\mathbf{1}\\\mathbf{1}\end{bmatrix} \right\|_2^2, \tag{9}$$

$$\text{s.t.} \begin{bmatrix} \mathbf{0} & \mathbf{G}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_1 \\ \mathbf{G}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_2 \\ \mathbf{G}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_3 & \mathbf{0} \end{bmatrix} \phi = \begin{bmatrix}\mathbf{0}\\\mathbf{0}\\\mathbf{0}\\\mathbf{0}\\\mathbf{0}\\\mathbf{0}\end{bmatrix}, \text{ and } \begin{bmatrix} \mathbf{G}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_3 \end{bmatrix} \phi > \begin{bmatrix}\mathbf{0}\\\mathbf{0}\\\mathbf{0}\end{bmatrix}. \tag{10}$$

That is, we constrain the gradients of the parametrization to follow the frame field (eq. 10), while optimizing for a regular, equi-spaced, solution (eq. 9). Unfortunately, the above formulation turns out to be infeasible for most of our test cases. A counting argument suggests a probable cause: our target solution has $3|\mathcal{V}|$ variables, but we apply $9|\mathcal{T}|$ constraints. Since, typically $|\mathcal{T}| \gg |\mathcal{V}|$ for manifold meshes (with low genus), the problem is likely to be overconstrained.

Therefore, we reformulate Equations 9–10 into an unconstrained weighted quadratic minimization.

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\ \beta \left\| \begin{bmatrix} G_1 & 0 & 0 \\ 0 & G_2 & 0 \\ 0 & 0 & G_3 \end{bmatrix} \phi - \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\|_2^2 \dots + \left\| \begin{bmatrix} 0 & G_1 & 0 \\ 0 & 0 & G_1 \\ G_2 & 0 & 0 \\ 0 & 0 & G_2 \\ G_3 & 0 & 0 \\ 0 & G_3 & 0 \end{bmatrix} \phi - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right\|_2^2 \tag{11}$$

where the parameter β provides user control over the regularity of the truss spacing. Figure 7 shows how the value of β influences the solution.

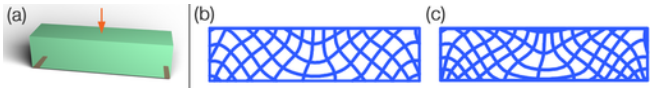


Fig. 7. Varying β to control truss spacing for the bar with fixed ends problem: the problem domain and boundary conditions (a), a slice of the truss extracted by setting $\beta = 1$, thus preferring regular spacing over orthogonality of the parametrization (b), and the same slice with $\beta = 0.1$, which favours parameter orthogonality (c).

4.4 Truss Layout Extraction

In the final step of our algorithm, the parametrization is used to extract the truss layout as a graph embedded in Ω. In order to avoid confusion with the vertices and edges of the input geometry, we exclusively use the words *nodes* and *elements* to refer to the vertices and edges of the graph extracted from the parametrization. Similar to parametrization based approaches for hex meshing [Lyon et al. 2016; Nieser et al. 2011], our aim is to trace the integer isocurves of the parametrization. That is, we want the nodes $\mathcal{N}$ of the extracted graph to be the points mapped to integers, i.e.,

$$\mathcal{N} = \{\mathbf{x} \in \Omega \mid \phi(\mathbf{x}) \in \mathbb{Z}^3\}, \tag{12}$$

and the elements $\mathcal{E}$ connect adjacent points on an integer grid, i.e.,

$$\mathcal{E} = \{\{\mathbf{x}, \mathbf{y}\} \mid \mathbf{x}, \mathbf{y} \in \mathcal{N},\ \phi(\mathbf{x}) - \phi(\mathbf{y}) \in \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}\}, \tag{13}$$

where $\mathbf{e}_i$'s are the standard basis vectors.

Note that only the gradient directions of our parametrization have any physical meaning, and therefore, applying an arbitrary translation and/or scale to the parametrization essentially keeps the physical information intact. In order to provide user control over the density of the extracted truss, we first translate and scale ϕ to normalize it to the range $[0, 1]$, and then scale by a user-defined "resolution" parameter ρ. We refer to this translated and scaled parametrization as $\widetilde{\phi} = (\widetilde{\phi}_1, \widetilde{\phi}_2, \widetilde{\phi}_3)$.

As noted earlier, existing work on hex-meshing assumes that the parametrization is defined such that for all points on the domain boundary, at least one of the parameter values is an integer. Since this is not true for our parametrizations, we also have to include additional nodes on the boundary, along with elements connecting these to each other and to the internal nodes. We defer the discussion of the special considerations for the boundary for now, and describe our algorithm for tracing the internal elements first.

Both 2D and 3D	
$\partial\Omega$	Boundary of the input domain
$\partial\mathcal{M}$	The simplicial 2-complex bounding $\mathcal{M}$
$\mathcal{N}$	Points on the integer grid defined by $\widetilde{\phi}$ (the set of nodes of the truss layout)
$\mathcal{E}$	The set of elements of the truss layout
2D	
γ_i	An integer isocurve of $\widetilde{\phi}_i$
Γ_i	The set of all γ_i
$\mathcal{N}_{Ei}$	Intersection points b/w curves in Γ_i and all edges of $\mathcal{M}$
$\mathcal{N}_E$	Disjoint union of $\mathcal{N}_{E1}$ and $\mathcal{N}_{E2}$
$\mathcal{N}_e$	Points in $\mathcal{N}_E$ lying on a particular edge e
$\mathcal{N}_{f,\gamma_i}$	Points on the integer grid, lying on the intersection b/w γ_i and a particular face f
$\mathcal{N}_F$	Union of $\mathcal{N}_{f,\gamma_1}$ and $\mathcal{N}_{f,\gamma_2}$ over all faces of $\mathcal{M}$
$\mathcal{E}_i$	Set of all elements tracing integer isocurves of $\widetilde{\phi}_i$
3D	
$\mathcal{S}_i$	An integer isosurface of $\widetilde{\phi}_i$
γ_{ij}	An integer isocurve of $(\widetilde{\phi}_i, \widetilde{\phi}_j)$
$\mathcal{N}_F$	Points of intersections between all $\{\gamma_{ij}\}$ and faces of $\mathcal{M}$
$\mathcal{N}_E$	Points of intersections between all $\{\gamma_{ij}\}$ and edges of $\partial\mathcal{M}$

Table 1. Notation for the truss layout extraction procedure described in subsection 4.4.

For ease of exposition, we start by describing a 2D truss layout extraction algorithm. Recall that our stress fields do not have singularities in the interior of the domain since forces are only applied at the boundary. This implies that the principal stress lines must end at the boundary, and cannot end abruptly or form closed surfaces inside the domain. Since the previous steps of the algorithm ensure that the isosurfaces of $\widetilde{\phi}$ follow the principal stress lines, we make the assumption that they do not form internal closed loops as well. Therefore, for tracing these isocurves, we start at their end points on the boundary, and trace until we hit the boundary again.

We use Γ_i to refer to the set of end-to-end integer isocurves of $\widetilde{\phi}_i$, and γ_i to refer to an arbitrary curve from this set. Note that $\widetilde{\phi}$ is a piecewise linear field stored at the vertices, and its value at arbitrary $\mathbf{x} \in \Omega$ can be found using Barycentric interpolation. However, Barycentric interpolation on a triangle is equivalent to linear interpolation along edges followed by interpolating across a line segment between two edges. We utilize this series of linear interpolations to trace out the integer isocurves of our parametrization.

In order to make this approach work, we require that an isocurve and a face intersect in exactly two points (or do not intersect at all). The only excluded cases are when an isocurve just touches a face at a single vertex, or is aligned with one of the edges (Figure 8).

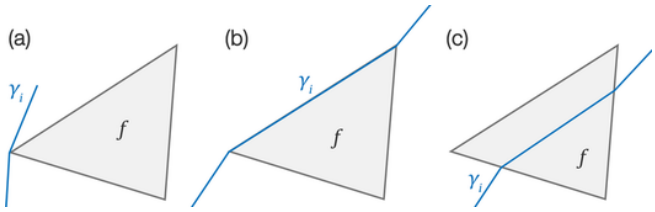


Fig. 8. An integer isocurve γ_i can intersect a face f either on a single vertex (a), on an edge (b), or go through its interior (c). We perturb the parametrization by an infinitesimal amount to eliminate the first two cases.

While we never encountered these cases with our paramterizations, we can easily eliminate the theoretical possibility as well. We find the parameter values on vertices which are close to integers up to machine precision, and translate them by $-\epsilon$, where ϵ is a small positive number (we choose 10^{-7}). If the vertex also has the minimum parameter value in its 1-ring neighbourhood, we translate by $+\epsilon$ instead, ensuring that we do not remove part of an integer isocurve. This ensures that no γ_i passes through a vertex, eliminating both the problematic cases.

Starting from the parameter values stored at the vertices, we use linear interpolation along edges to find the intersections of curves from Γ_1 and Γ_2 with the edges to form the sets of nodes $\mathcal{N}_{E1}$ and $\mathcal{N}_{E2}$, respectively (Figure 9a). These nodes are then used to find nodes in the interior of faces, and their connectivity, as described below.

Consider a face f and an isocurve $\gamma_1 \in \Gamma_1$ intersecting with it. Let $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{N}_{E1}$ be the end points of the line of intersection. We linearly interpolate the values of $\widetilde{\phi}_1$ on $\mathbf{x}_0$ and $\mathbf{x}_1$ to find the points of intersection of this isoline with all $\gamma_2 \in \Gamma_2$:

$$\mathcal{N}_{f,\gamma_1} = \left\{ \mathbf{y} \in f \cap \gamma_1 \mid \widetilde{\phi}_2(\mathbf{y}) \in \mathbb{Z} \cap \left(\widetilde{\phi}_2(\mathbf{x}_0), \widetilde{\phi}_2(\mathbf{x}_1) \right) \right\}, \tag{14}$$

where $\widetilde{\phi}_2(\mathbf{x}_0) \leq \widetilde{\phi}_2(\mathbf{x}_1)$ *wlog.* $\mathcal{N}_{f,\gamma_1}$ is then sorted by $\widetilde{\phi}_2$, the two extrema are connected to $\mathbf{x}_0$ and $\mathbf{x}_1$, and consecutive points in the ordered set are connected to each other. Doing this for all admissible pairs (f, γ_1) gives the set $\mathcal{N}_F$ of nodes lying on the intersections between all pairs (γ_1, γ_2), and the set of elements $\mathcal{E}_1$ tracing all $\gamma_1 \in \Gamma_1$ (Figure 9b–c).

Then, for each pair of intersecting face and γ_2, we search for the nodes among $\mathcal{N}_F$ lying on the intersection. The nodes lying on each γ_2 are then connected to form the set of elements $\mathcal{E}_2$ (Figure 9d). Define $\mathcal{N}_E$ to be the disjoint union of $\mathcal{N}_{E1}$ and $\mathcal{N}_{E2}$, and $\mathcal{N}_{\partial\Omega}$ to be the subset of $\mathcal{N}_E$ restricted to nodes lying on the boundary. In the final step of the algorithm, we insert all nodes from $\mathcal{N}_{\partial\Omega}$ into a queue and trace out the integer isocurves emanating from them, going through the faces it intersects until we hit the boundary again. For each traced curve, we remove its endpoints from the queue.

4.4.1 3D Truss Layouts. In 3D we use $\mathcal{S}_i$ to denote an arbitrary end-to-end integer isosurface of $\widetilde{\phi}_i$, and γ_{ij} to denote an arbitrary end-to-end curve where both $\widetilde{\phi}_i$ and $\widetilde{\phi}_j$ are constant integers. After perturbing the parametrization, we compute the points of intersection of isosurfaces of each of the three parameters with the edges. Then, we compute the intersection points of each γ_{ij} with all the

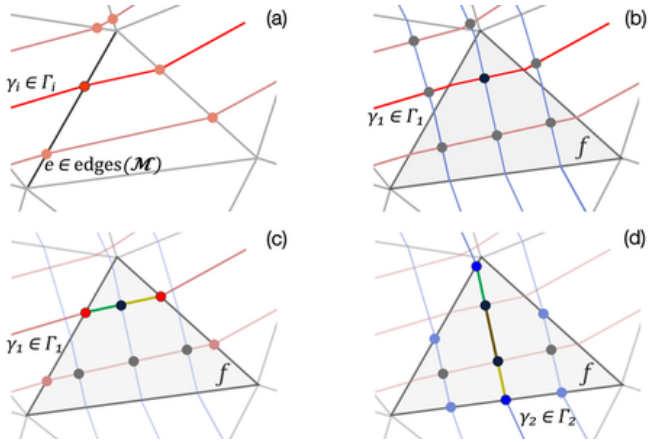


Fig. 9. In 2D, the truss extraction process begins by finding all points of intersections between integer isocurves of $\widetilde{\phi}$ and edges of the input mesh (a). Then, for each face of the mesh f, and an integer isocurve of $\widetilde{\phi}_1$ intersecting with it, points mapped to the integer grid are located (b). All such points form the set of nodes $\mathcal{N}$ for the truss. Finally, the linear section of each $\widetilde{\phi}_1$ isocurve is cut along these points to form the elements $\mathcal{E}_1$ for the output truss (c), followed by a similar discretization and tracing process for $\widetilde{\phi}_2$ isolines to form $\mathcal{E}_2$ (d). The union of $\mathcal{E}_1$ and $\mathcal{E}_2$ is the set of elements $\mathcal{E}$ of the truss.

faces of the mesh. Finally, we linearly interpolate $\widetilde{\phi}_k (k \neq i, j)$ along these isolines in each tet, and compute the intersections with all $\mathcal{S}_k$ to find the elements of the truss. Note that while the perturbation does not guarantee that every γ_{ij} passes through the interior of all tets—it may just touch it at a face—we never encountered this case in practice.

4.4.2 Handling the boundary. The described procedure already traces out the nodes on the boundary, as well the elements which touch it. In 2D, we can add the boundary by tracing the elements on each mesh edge individually—for an edge $e = (\mathbf{x}_0, \mathbf{x}_1)$, sort all the points in $\{\mathbf{x}_0, \mathbf{x}_1\} \sqcup \mathcal{N}_e$ by $\widetilde{\phi}_1$ (or equivalently, by $\widetilde{\phi}_2$) and connect adjacent points in the sorted list.

In 3D, we need to trace the intersection of each $\mathcal{S}_i$ with the boundary $\partial\mathcal{M}$, which comes down to performing the full 2D truss extraction procedure for each pair of parameters $\{i, j\} \in \binom{3}{2}$ on the triangle mesh $\partial\mathcal{M}$.

4.4.3 Simplification. Our extraction procedure can result in many nodes which are geometric duplicates of each other, or are very close to each other. However, we take care to get the correct graph topology in $\mathcal{E}$ so that such duplicates can be easily removed by performing edge contraction operations on the graph with a small element length threshold. Optionally, we can contract elements until we have no nodes from $\mathcal{N}_F$ left ($\mathcal{N}_E$ in 2D), except those on the boundary. This gives us the graph similar to that in Equations (12)–(13), but with additional nodes and elements on the boundary.

We can simplify the boundary elements as well by contracting elements containing the nodes in $\mathcal{N}_E$ (boundary vertices in 2D). We perform both these steps for all our results, but we do not remove boundary nodes which lie on feature edges (feature vertices in 2D).

These features are currently selected using dihedral angles with a selection threshold of $\cos^{-1}(0.9)$ (approx. 25°), but one could easily plug in user-provided features. Finally, in 3D, we trace these feature edges as well to preserve surface features.

4.5 Implementation Details

Our pipeline is implemented using a mix of MATLAB [The MathWorks, Inc. 2017] and C++ functions. Notably, we rely on C++ finite element analysis code and some underlying C++ functions of the libigl library [Jacobson et al. 2016b] called via the matlab gptoolbox code base [Jacobson et al. 2016a]. Our frame fitting, texture parametrization, and mesh extraction algorithms are implemented entirely in MATLAB. We use MATLAB's `fmincon` to solve our frame fitting optimization and `quadprog` to solve for the texture paramaterization. We pledge to release all code and data pertaining to this submission as opensource.

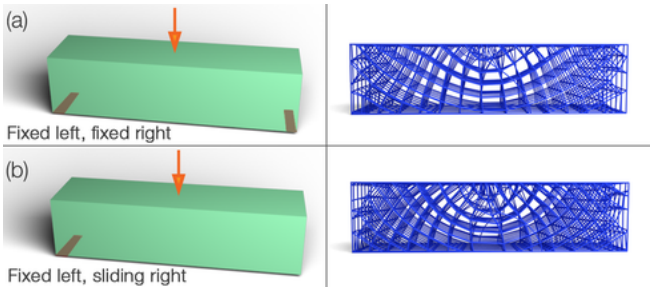


Fig. 10. A point compression is applied at the center of the top face of the bar, with two different boundary conditions: fixed left and right edges on the bottom face (top), and fixed left edge with right edge allowed to slide horizontally (bottom). Notice the change in the slope of the curves with the change in the boundary conditions.

5 GEOMETRY CREATION

The final stage in our pipeline is to create geometry from our extracted truss networks. To do this we specify a radius for each integer isocurve. We then replace each piecewise linear segment of the truss graph with a triangulated cylinder. The union of these cylinders, computed using libigl's [Jacobson et al. 2016b] robust boolean operations, results in a full 3D geometry

We show virtual renders of the optimized trusses produced by our method in Figures 11 and 12. In all the figures in the paper, the input geometry is consistently shown in green, the forces applied with orange arrows, the fixed points in maroon, and the locations of load application with a striped orange pattern. For 3D results optimized purely under gravity, the ground contacts are always set to be fixed, and are not explicitly shown in the figures.

Constructing volumetric Michell Trusses can be useful for many engineering applications. Figure 12 shows some applications of our method in the aerospace industry. We also show furniture design applications (Figure 14).

We also performed FEM analysis to visualize the stress fields in several of our examples (Figure 15).

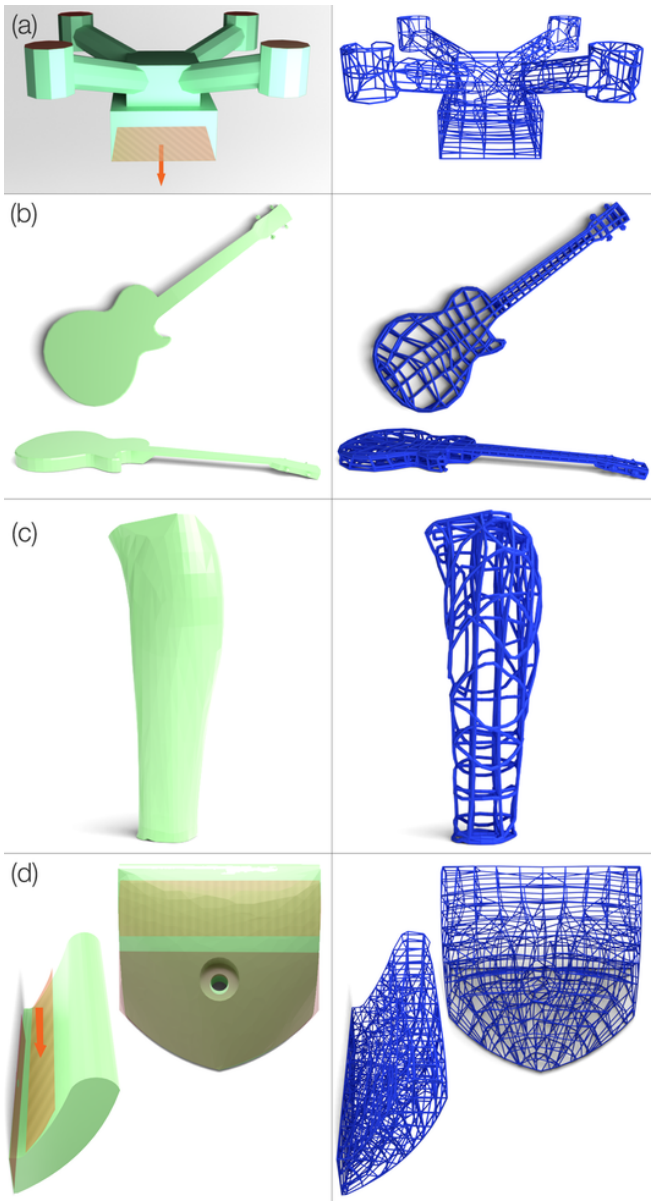


Fig. 11. A selected set of trusses produced by our method. (Top to bottom) quadcopter fixed at the propeller mounts and optimized for carrying a parcel hanging from the bottom face, a guitar optimized for load applied by a guitar strap, a prosthetic leg and a rock climbing hold (shown in two views for clarity). The left column shows inputs, and the right column shows the corresponding rendered results.

6 EXPRESSIVITY OF MICHELL TRUSSES

In this section we explore the effect of geometry and boundary conditions on the output of our method. Figure 10 shows the results of our algorithm on two bars. The top bar has fixed Dirichlet boundary conditions at each end while the bottom allows free sliding of its right-hand side. Note the difference in the produced trusses.

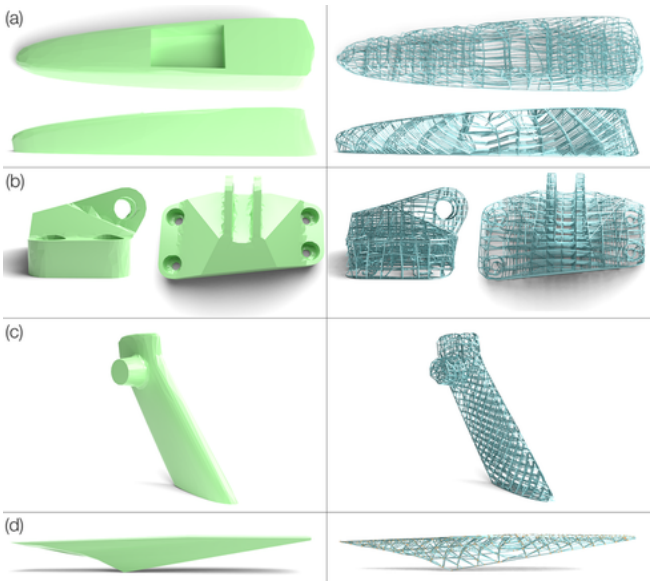


Fig. 12. Aerospace applications: helicopter top pylon, airplane engine bracket, helicopter rear pylon, and airplane engine/turbine pylon.

Figure 13 shows the effect of changing the loading conditions. Two identical bars are subjected to twisting and compression respectively. The resulting truss structures adapt to provide maximum structural strength.

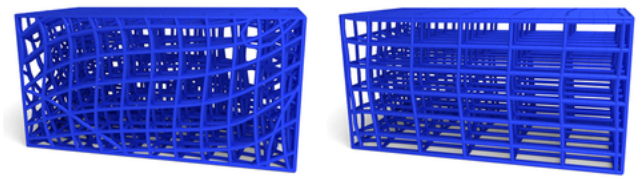


Fig. 13. The difference caused by varying Neumann BCs, using bars under torsion (left) and compression (right).

Figure 14 shows two chairs with the same loading condition applied. The left chair is a more standard, swivel chair design while the right chair is produced using generative design software. Note the differences in the trusses due to the changes in geometry.

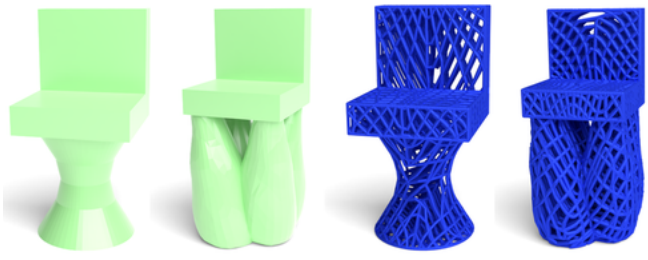


Fig. 14. Two chairs under identical loading but with different geometries.

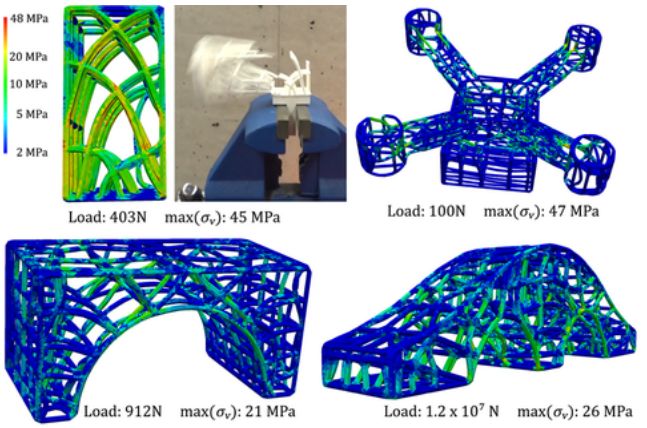


Fig. 15. We used linear FEM to simulate our trusses. The material assumed for the visualizations is ABS-M30 plastic (yield strength 48 MPa), which we used for fabricating the bridge, quadcopter frame, and bending bar. The bending bar yielded at 403N. The simulation (a) shows excellent accuracy in predicting the stress concentration regions, which agree with experimentally observed fracture regions (b). The quadcopter frame is predicted to hold up 100N (10.2kg) of load successfully (c). The miniature (20cm wide) flat bridge simulation (d) predicts no fracture at 912N (93kg) load. A real-world scale arch-bridge made of 2.5cm thick elements (c) is predicted to hold 1.2×10^7 N (10 firetrucks).

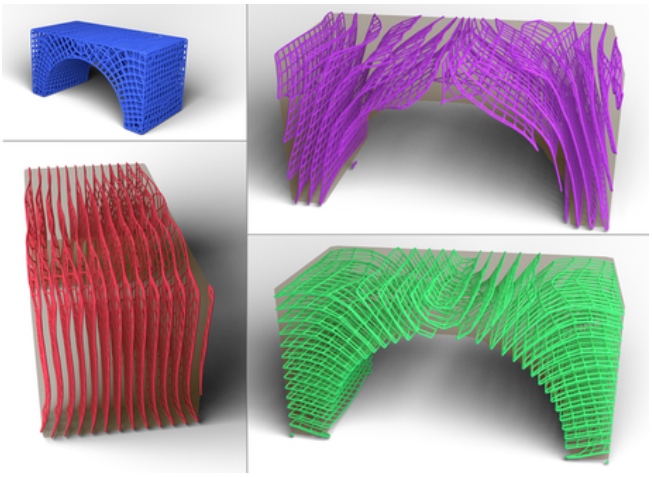


Fig. 16. Visualization of the trusses extracted by our method by tracing the integer isosurfaces of $\widetilde{\phi}$. The full truss for the bridge problem (blue), and truss elements traced on integer isosurfaces of each $\widetilde{\phi}_i$ in red, green, and violet.

7 PERFORMANCE

Table 2 reports the performance of our algorithm for all the tested 3D problems—problem size, total running time, and the time taken by individual steps of the algorithm. The results indicate running time on a 2017 Macbook Pro with Intel Core i7 Processor and 16 GB of RAM. Note that most of our code is written in MATLAB and is not optimized for speed. Based on previous studies, we expect an

order of magnitude or more speedup simply by implementing the algorithm in C++ [Kjolstad et al. 2016].

8 CONCLUSION

By adopting a parametrization-based approach we have crafted the first algorithm for the design of volumetric Michell Trusses and shown that the algorithm can produce complex, aesthetically pleasing output that is also strong. We believe our method serves as an important companion to traditional approaches while also providing engineers, architects, and designers with an exciting new algorithmic tool.

Limitations and Future Work. The most significant limitation of our approach is that it does not incorporate fabrication constraints. Incorporating such constraints into design optimization is an ongoing area of research. We are motivated by recent works to investigate this further [Allaire et al. 2016; Martínez et al. 2018].

Our method produces trusses whose members are almost equally spaced. This can be problematic when the input geometry contains very thin regions, such as the dragon shown in Figure 17, where we completely miss the dragon's horns. Fortunately, if the aim is to build objects with a solid shell as the surface, we can still optimize the internal structure.

In general, efficient and fine-grained user control of topology optimization remains an interesting direction for future research.

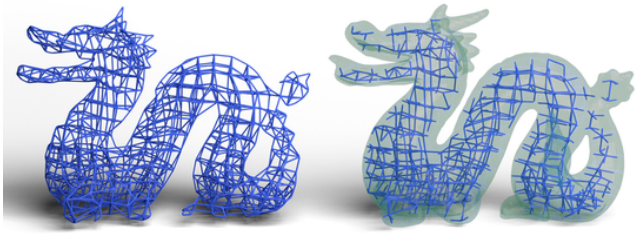
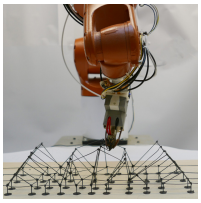


Fig. 17. The trusses produced by our method may fail to cover thin features like the dragon's horns (left). However, it is still useful for optimizing the interior of an object meant to be fabricated with a solid surface (right).



Lastly, parameterized trusses can be used to fabricate structures using new, exotic, fabrication methods such as extruders mounted on robot arms with many degrees of freedom [MX3D 2017; Tam et al. 2015]. Such robot arms can build large, self-supported structures in a fraction of time required by current generation 3D printers, and therefore, hold promise to be the additive manufacturing technique of the future.

REFERENCES

Niels Aage, Erik Andreassen, Boyan S Lazarov, and Ole Sigmund. 2017. Giga-voxel computational morphogenesis for structural design. *Nature* 550, 7674 (2017), 84–86.

Wolfgang Achtziger and Mathias Stolpe. 2007. Truss topology optimization with discrete design variables—guaranteed global optimality and benchmark examples. *Structural and Multidisciplinary Optimization* 34, 1 (2007), 1–20.

Grégoire Allaire, François Jouve, and Georgios Michailidis. 2016. *Molding Direction Constraints in Structural Optimization via a Level-Set Method.* Springer International Publishing, Cham, 1–39. https://doi.org/10.1007/978-3-319-45680-5_1

Grgoire Allaire, Franois Jouve, and Anca-Maria Toader. 2004. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.* 194, 1 (2004), 363 – 393. https://doi.org/10.1016/j.jcp.2003.09.032

Autodesk. 2018. Fusion 360. (2018). https://www.autodesk.com/products/fusion-360/overview

Ted Belytschko, Wing Kam Liu, Brian Moran, and Khalil Elkhodary. 2013. *Nonlinear finite elements for continua and structures.* John Wiley & Sons.

M. P. Bendsøe, A. Ben-Tal, and J. Zowe. 1994. Optimization methods for truss geometry and topology design. *Structural optimization* 7, 3 (01 Apr 1994), 141–159. https://doi.org/10.1007/BF01742459

Martin P Bendsøe and Ole Sigmund. 2009. *Topology Optimization.* World Scientific.

C.V. Camp and M. Farshchin. 2014. Design of space trusses using modified teaching–learning based optimization. *Engineering Structures* 62-63, Supplement C (2014), 87 – 97. https://doi.org/10.1016/j.engstruct.2014.01.020

W. S. Dorn, R. E. Gomory, and H. J. Greenberg. 1964. Automatic design of optimal structures. *Journal de Mecanique* 3 (1964), 25–52.

Z. Doubrovski, J.C. Verlinden, and J.M.P. Geraedts. 2011. Optimal Design for Additive Manufacturing: Opportunities and Challenges. In *Proc. 23rd International Conference on Design Theory and Methodology.* 635–646.

Robert M Freund. 2004. Truss design and convex optimization. *Massachusetts Institute of Technology* (2004).

Arun L. Gain, Glaucio H. Paulino, Leonardo S. Duarte, and Ivan F.M. Menezes. 2015. Topology optimization using polytopes. *Computer Methods in Applied Mechanics and Engineering* 293 (2015), 411 – 430. https://doi.org/10.1016/j.cma.2015.05.007

Phillip L Gould and Yuan Feng. 1994. *Introduction to linear elasticity.* Springer.

Seung-Hyun Ha and Seonho Cho. 2008. Level set based topological shape optimization of geometrically nonlinear structures using unstructured mesh. *Computers & Structures* 86, 13 (2008), 1447 – 1455. https://doi.org/10.1016/j.compstruc.2007.05.025 Structural Optimization.

Roger A Horn and Charles R Johnson. 1990. *Matrix analysis.* Cambridge university press.

Alec Jacobson et al. 2016a. gptoolbox: Geometry Processing Toolbox. (2016). http://github.com/alecjacobson/gptoolbox.

Alec Jacobson, Daniele Panozzo, et al. 2016b. libigl: A simple C++ geometry processing library. (2016). http://libigl.github.io/libigl/.

Benjamin P Jacot and Caitlin T Mueller. 2017. A strain tensor method for three-dimensional Michell structures. *Structural and Multidisciplinary Optimization* 55, 5 (2017), 1819–1829.

Caigui Jiang, Chengcheng Tang, Hans-Peter Seidel, and Peter Wonka. 2017. Design and Volume Optimization of Space Structures. *ACM Trans. Graph.* 36, 4, Article 159 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073619

A Kaveh, B Farhmand Azar, and S Talatahari. 2008. Ant colony optimization for design of space trusses. *International Journal of Space Structures* 23, 3 (2008), 167–181.

H Kawamura, H Ohmori, and N Kito. 2002. Truss topology optimization by a modified genetic algorithm. *Structural and Multidisciplinary Optimization* 23, 6 (2002), 467–473.

Fredrik Kjolstad, Shoaib Kamil, Jonathan Ragan-Kelley, David I. W. Levin, Shinjiro Sueda, Desai Chen, Etienne Vouga, Danny M. Kaufman, Gurtej Kanwar, Wojciech Matusik, and Saman Amarasinghe. 2016. Simit: A Language for Physical Simulation. *ACM Trans. Graph.* 35, 2, Article 20 (May 2016), 21 pages. https://doi.org/10.1145/2866569

Timothy Langlois, Ariel Shamir, Daniel Dror, Wojciech Matusik, and David I. W. Levin. 2016. Stochastic Structural Analysis for Context-aware Design and Fabrication. *ACM Trans. Graph.* 35, 6, Article 226 (Nov. 2016), 13 pages. https://doi.org/10.1145/2980179.2982436

David I.W. Levin et al. 2017. GAUSS: Gaggle of Algorithm for Simulating Stuff. *https://github.com/dilevin/GAUSS* (2017).

David I.W. Levin, Benjamin Gilles, Burkhard Mädler, and Dinesh K. Pai. 2011. Extracting skeletal muscle fiber fields from noisy diffusion tensor data. *Medical Image Analysis* 15, 3 (2011), 340 – 353. https://doi.org/10.1016/j.media.2011.01.005

Bruno Lévy and Yang Liu. 2010. Lp Centroidal Voronoi Tessellation and Its Applications. *ACM Trans. Graph.* 29, 4, Article 119 (July 2010), 11 pages. https://doi.org/10.1145/1778765.1778856

L. J. Li, Z. B. Huang, and F. Liu. 2009. A Heuristic Particle Swarm Optimization Method for Truss Structures with Discrete Variables. *Comput. Struct.* 87, 7-8 (April 2009), 435–443. https://doi.org/10.1016/j.compstruc.2009.01.004

Yongqiang Li and Yong Chen. 2010. Beam structure optimization for additive manufacturing based on principal stress lines. In *Solid Freeform Fabrication Proceedings.* 666–678.

Max Lyon, David Bommes, and Leif Kobbelt. 2016. HexEx: Robust Hexahedral Mesh Extraction. *ACM Trans. Graph.* 35, 4, Article 123 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925976

Yuxin Mao, Lifang Wu, Dong-Ming Yan, Jianwei Guo, Chang Wen Chen, and Baoquan Chen. 2018. Generating hybrid interior structure for 3D printing. *Computer Aided Geometric Design* 62 (2018), 63 – 72. https://doi.org/10.1016/j.cagd.2018.03.015

Jonàs Martínez, Samuel Hornus, Haichuan Song, and Sylvain Lefebvre. 2018. Polyhedral Voronoi diagrams for additive manufacturing. *ACM Transactions on Graphics* 37, 4

Example	# Vertices	# Tets	Total (s)	Sim (s)	Frames (s)	Tex (s)	Extract (s)
Bar (bending)	3457	16704	618.0	0.5	20.8	1.2	595.4
Bar (simply supported)	6913	34128	583.1	1.3	69.1	3.7	509.0
Climbing Hold	13753	68773	2189.7	3.7	209.8	14.8	1961.5
Guitar	10171	46840	689.9	1.3	54.1	7.2	627.2
Bar (torsion)	3457	16704	2239.2	0.4	17.3	1.0	2220.4
Bar (compression)	3457	16704	1411.4	0.5	15.5	1.0	1394.4
Chair (swivel)	7372	35021	4587.1	1.2	102.7	4.0	4479.3
Chair (generative)	9801	46187	6278.7	1.7	99.5	6.7	6170.8
Bar (fixed)	6913	34128	511.2	1.2	51.0	3.6	455.4
Pylon (helicopter, top)	8268	35449	831.9	1.0	44.4	4.3	782.3
Pylon (helicopter, rear)	4038	17358	609.4	0.3	17.8	1.2	590.0
Prosthetic	9097	44478	1380.3	1.5	77.2	5.9	1295.6
Bridge	4326	18808	648.5	0.5	89.8	1.5	556.6
Pylon (turbine)	2520	8932	225.9	0.2	8.0	0.4	217.3
Antenna arm	3264	13459	1345.6	0.3	13.0	0.7	1331.6
Jet engine bracket	28904	135649	3996.4	4.0	1501.2	52.1	2439.1
Bunny	19866	87443	1471.2	3.7	175.3	23.6	1268.6
Quadcopter	7121	32826	671.2	1.0	45.9	3.7	620.7
Bridge (arch)	6457	31164	621.2	0.9	73.7	3.1	543.5

Table 2. Performance results for all the 3D testcases. All reported runtimes have been rounded off to the nearest second.

(Aug. 2018), 15. https://doi.org/10.1145/3197517.3201343

Jonàs Martínez, Haichuan Song, Jérémie Dumas, and Sylvain Lefebvre. 2017. Orthotropic K-nearest Foams for Additive Manufacturing. *ACM Trans. Graph.* 36, 4, Article 121 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073638

A. G. M. Michell. 1904. LVIII. The limits of economy of material in frame-structures. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 8, 47 (1904), 589–597. https://doi.org/10.1080/14786440409463229 arXiv:https://doi.org/10.1080/14786440409463229

MX3D. 2017. Metal - MX3D. (2017). http://mx3d.com/projects/metal/

Jason Nguyen, S Park, David W Rosen, Luis Folgar, and James Williams. 2012. Conformal lattice structure design and fabrication. In *Solid Freeform Fabrication Symposium*. Austin, TX, 138–161.

Matthias Nieser. 2018. personal communication. (2018).

M. Nieser, U. Reitebuch, and K. Polthier. 2011. CubeCover—Parameterization of 3D Volumes. *Computer Graphics Forum* 30, 5 (2011), 1397–1406. https://doi.org/10.1111/j.1467-8659.2011.02014.x

Jorge Nocedal and Stephen Wright. 2006. *Penalty and Augmented Lagrangian Methods*. Springer Science & Business Media.

Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. 2014. Frame Fields: Anisotropic and Non-orthogonal Cross Fields. *ACM Trans. Graph.* 33, 4, Article 134 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601179

M.H. Rasmussen and M. Stolpe. 2008. Global optimization of discrete truss topology design problems using a parallel cut-and-branch method. *Computers & Structures* 86, 13 (2008), 1527 – 1538. https://doi.org/10.1016/j.compstruc.2007.05.019 Structural Optimization.

Nicolas Ray, Dmitry Sokolov, and Bruno Lévy. 2016. Practical 3D Frame Field Generation. *ACM Trans. Graph.* 35, 6, Article 233 (Nov. 2016), 9 pages. https://doi.org/10.1145/2980179.2982408

Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. 2015. Microstructures to Control Elasticity in 3D Printing. *ACM Trans. Graph.* 34, 4, Article 136 (July 2015), 13 pages. https://doi.org/10.1145/2766926

Ole Sigmund, Niels Aage, and Erik Andreassen. 2016. On the (Non-)Optimality of Michell Structures. *Struct. Multidiscip. Optim.* 54, 2 (Aug. 2016), 361–373. https://doi.org/10.1007/s00158-016-1420-7

Justin Solomon, Amir Vaxman, and David Bommes. 2017. Boundary Element Octahedral Fields in Volumes. *ACM Trans. Graph.* 36, 3, Article 28 (May 2017), 16 pages. https://doi.org/10.1145/3065254

Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. 2012. Stress Relief: Improving Structural Strength of 3D Printable Objects. *ACM Trans. Graph.* 31, 4, Article 48 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185544

Mathias Stolpe and Krister Svanberg. 2003. Modelling topology optimization problems as linear mixed 0–1 programs. *Internat. J. Numer. Methods Engrg.* 57, 5 (2003), 723–739. https://doi.org/10.1002/nme.700

Kam-Ming Mark Tam, James R Coleman, Nicholas W Fine, and Caitlin T Mueller. 2015. Stress line additive manufacturing (SLAM) for 2.5-D shells. In *Proceedings of International Symposium on Shell and Spatial Structures*.

Kam-Ming Mark Tam and Caitlin T Mueller. 2017. Additive Manufacturing Along Principal Stress Lines. *3D Printing and Additive Manufacturing* 4, 2 (jun 2017), 63–81. https://doi.org/10.1089/3dp.2017.0001

The MathWorks, Inc. 2017. MATLAB. (2017). https://www.mathworks.com/products/matlab.html

Evgueni Todorov, Roger Spencer, Sean Gleeson, Madhi Jamshidinia, and Shawn M. Kelly. 2014. *Nondestructive Evaluation (NDE) of Complex Metallic Additive Manufactured (AM) Structures.* Technical Report AFRL-RX-WP-TR-2014-0162. Air Force Research Laboratory, Wright-Patterson Airforce Base, Ohio, USA. 70 pages. http://www.dtic.mil/dtic/tr/fulltext/u2/a612775.pdf

Erva Ulu, James Mccann, and Levent Burak Kara. 2017. Lightweight Structure Design Under Force Location Uncertainty. *ACM Trans. Graph.* 36, 4, Article 158 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073626

Nobuyuki Umetani and Ryan Schmidt. 2013. Cross-sectional Structural Analysis for 3D Printing Optimization. In *SIGGRAPH Asia 2013 Technical Briefs (SA '13)*. ACM, New York, NY, USA, Article 5, 4 pages. https://doi.org/10.1145/2542355.2542361

Jess M. Waller, Bradford H. Parker, Kenneth L. Hodges, Eric R. Burke, and James L. Walker. 2014. *Nondestructive Evaluation of Additive Manufacturing*. Technical Report NASA/TM—2014–218560. National Aeronautics and Space Administration (NASA), Hampton, Virginia, USA. 47 pages. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140016447.pdf

Michael Yu Wang, Xiaoming Wang, and Dongming Guo. 2003. A level set method for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering* 192, 1 (2003), 227 – 246. https://doi.org/10.1016/S0045-7825(02)00559-5

Weiming Wang, Tuanfeng Y. Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. 2013. Cost-effective Printing of 3D Objects with Skin-frame Structures. *ACM Trans. Graph.* 32, 6, Article 177 (Nov. 2013), 10 pages. https://doi.org/10.1145/2508363.2508382

Li Wei, Zheng Anzong, You Lihua, Yang Xiaosong, Zhang Jianjun, and Liu Ligang. 2017. Rib Reinforced Shell Structure. *Computer Graphics Forum* 36, 7 (2017), 15–27. https://doi.org/10.1111/cgf.13268 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13268

Jun Wu, Niels Aage, Ruediger Westermann, and Ole Sigmund. 2017. Infill Optimization for Additive Manufacturing–Approaching Bone-like Porous Structures. *IEEE Transactions on Visualization and Computer Graphics* (2017).

Tomás Zegard and Glaucio H. Paulino. 2015. GRAND3 – Ground Structure Based Topology Optimization for Arbitrary 3D Domains Using MATLAB. *Struct. Multidiscip. Optim.* 52, 6 (Dec. 2015), 1161–1184. https://doi.org/10.1007/s00158-015-1284-2

Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2016. Designing Structurally-sound Ornamental Curve Networks. *ACM Trans. Graph.* 35, 4, Article 99 (July 2016), 10 pages. https://doi.org/10.1145/2897824.2925888

Qingnan Zhou, Julian Panetta, and Denis Zorin. 2013. Worst-case Structural Analysis. *ACM Trans. Graph.* 32, 4, Article 137 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461967