BeNTO: Beam Network Topology Optimization

Abstract

We present an optimization framework that allows designers and engineers to develop conceptual designs for structures that can be manufactured using a network of connected beams. We leverage level set-based topology optimization along with a graph representation extracted from the skeleton of the level set. This graph is used to create an idealized beam network of the structure and the level set and the beam networks co-evolve during the course of the optimization. We propose a constraint that moves the level set geometry closer to this beam network and ultimately drives the two representations to converge. The graph representation also enables us to enforce manufacturability in the beam network, including minimum and maximum cross-section size, angle snapping and cross-section continuity. By applying the beam constraint during topology optimization we are able to generate designs that satisfy the structural requirements and can also be fabricated as beam networks. We present several example sets that illustrate the beam networks that can be generated, how they compare to results without the beam constraint as well as the impact of the manufacturability methods.

Contents

1. Introduction

Structural topology optimization is a powerful tool for conceptual design, allowing users to specify mechanical loads and fixed surfaces on an initial domain, and then by minimizing a given objective such as compliance, automatically generate a physically plausible geometry that can support these loads. One school of thought is to initialize the domain with primitives such as beams or trusses that are locally connected, often referred to as the ground structure method [1], and then to optimize for the cross-section sizing and nodal positions of the beams. Topology changes are typically applied intermittently as heuristic decisions based on local geometric features, local connectivity, strain energy, or nodal forces [2, 3, 4, 5]. The advantage of these methods is that the resulting geometry is guaranteed to be composed of the desired primitives and it is relatively simple to enforce constraints on the primitives and their connections. However, topological changes are non-differentiable making the choice of heuristics and initial domain crucial for obtaining a good solution.

Another popular form of topology optimization we refer to as continuum topology optimization. Such methods work by discretizing a domain into a regular grid or mesh and allocating each grid cell to be either empty, partial or solid. There are two primary methods for converging to a final shape—density-based methods iteratively modify the material density in each cell until it converges to either empty or solid [6, 7], while boundary-based methods track a boundary between solid and empty cells and use shape derivatives to iteratively refine

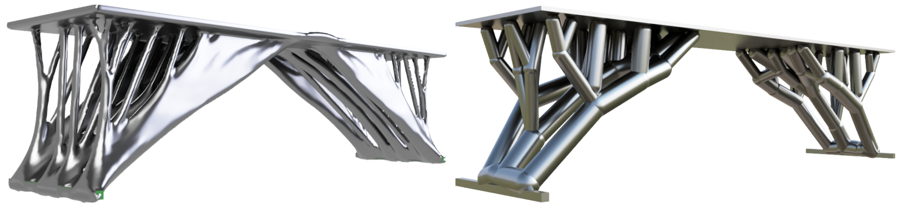


Figure 1: Left: Naïve topology optimization result for a footbridge. Right: CAD visualization of our beam network constrained topology optimization result for the same footbridge.

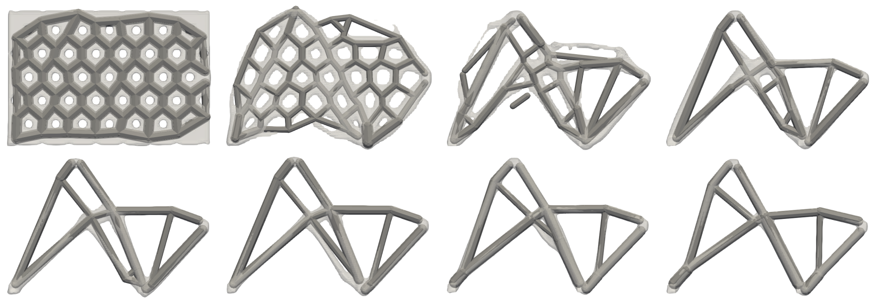


Figure 2: Shape co-evolution of the level set boundary in light gray and the idealized beam network in dark gray.

the boundary to convergence [8, 9]. One of the major advantages of continuum topology optimization methods is in their geometric initialization: a simple uniform allocation of material across the domain is usually sufficient to begin an optimization that will produce a meaningful design. In addition, topology changes are built-in and occur as cells in the grid are reduced to empty. Since material can be placed anywhere in the grid, continuum topology optimization implementations can generate optimal but also highly free-form geometry (Figure 1 Left).

The kind of geometric output produced by continuum topology optimization methods is often used as inspiration for a manual redesign or must be heavily processed before it is useful to the designer. One of the main reasons for this is to account for the method of manufacture of the final design. There have been many previous approaches to manufacturing-aware topology optimization for different manufacturing methods such as additive, milling, or casting [10, 11, 12, 13, 14, 15]. While these methods have greatly enhanced the practical applicability of continuum topology optimization, they have a number of limitations especially when designing for large-scale structures. Additive parts are limited in size by their print volume and speed [16], milled parts have a relatively high cost and their size is also limited by the milling machine [17] and cast parts are limited by their mold tool size and cost [18]. Additionally, most large structures are actually composed of a set of parts, typically extruded beams connected to each other by joints. There are a number of advantages to such structures: they can be arbitrarily large and most infrastructure is designed in this format; the beams can be composed of standard cross-sections and use standard or custom joint connections.

1.1. Our Approach

As we have seen, the ground structure method has the advantage of producing a connected beam structure that is readily manufacturable, however it has the disadvantage of non-smooth topology changes, and non-trivial initialization. Continuum methods are easy to initialize and have built-in topology changes, but in their näive form produce complex geometry that must be post-processed to manufacture. We now present a new method that utilizes the advantages of a continuum method but that also generates a connected beam structure that is easy to manufacture (Figure 1 Right).

Our work builds on the boundary-based continuum method called level set-based topology optimization (LSTO) where we constrain the shape boundary such that the resulting geometry can be modeled with a set of connected, uniform beams (note: we expect similar results can be achieved using density-based methods where the boundary is estimated using an iso-contour of the densities). This is achieved by extracting a graph of topological connectivity from the skeleton of the free-form level set geometry during topology optimization. The graph is used to construct an idealized beam network where cylindrical primitives are created along the graph edges and whose radii are matched to the level-set geometry. We then apply a shape-differentiable constraint that penalizes the difference between the shape of the idealized beam structure and the level set geometry. Initially this constraint has a light weight but it gets stronger as the optimization progresses. We also periodically update the graph and idealized beam structure during the optimization, thus allowing for topological changes and a co-evolution of both the level set geometry and the idealized beam structure. Figure 2 shows how the level set shape and the idealized beam network co-evolve during the optimization. Note that no special initialization of the beam network is needed and its topology and connectivity evolves along with the level set geometry, taking advantage of the best features of this continuum topology optimization method. At the same time, the shape is eventually forced to conform to the beam network and any non-beam like structures are suppressed.

Since we explicitly track the beam network during the optimization, we are also able to incorporate additional manufacturing constraints into the optimization, such as minimum and maximum cross-section sizing as well as conditions on the angles between beams connected at joints and enforce continuity along cross-sections. As an added benefit of explicitly representing the beam network, we can export it as a set of primitives that can easily be edited in a CAD tool, rather than having to edit or convert a dense mesh.

To summarize, our work introduces the following:

- A shape-differentiable framework for generating connected beam networks.
- A methodology for incorporating several requirements for manufacturability into the final beam networks.

1.2. Related Approaches

Our approach is closely related to the method of ‘Moving Morphable Components’ (MMC) [19, 20], where the design space is initially seeded with a set of primitives components that connect the loads to the fixed boundary conditions. During optimization the parameters of these components such as rotation, translation and scaling along each axis are optimized to satisfy the objective. The resulting shape, as in our method is a set of primitives, however there is no explicit connection between the primitives and the final shape is a Boolean union of the primitives which may overlap arbitrarily. These overlapping primitives may generate complex geometry that would be challenging to manufacture or for our purposes, replace with a network of beams even when only cylinder primitives are used. Additionally, the maximum number of primitives must be determined up front and their layout decided by the user during initialization, whereas in our method the graph conforms to the geometric complexity as it unfolds during the optimization and no special initialization is required.

In order to minimize the manual work required to process topology optimization results, Yin et al [21], Yi et al. [22] and Xia et al. [23] proposed methods to fit this geometry to a set of connected beams, trusses or primitives, which were used to construct parametric models of the geometry. However, in these methods there was no explicit control over the geometry generated by the topology optimization, and the beam and primitive generation was done as a post-processing stage. This can be problematic since traditional topology optimization results often result in complex, free-form shapes. These shapes optimally resist the loading conditions while minimizing a structural property such as compliance while being constrained to a particular mass or volume. Thus, they frequently have an organic structure, similar to bones or tree branches and can be very difficult to cleanly model with a set of connected beams, especially of uniform cross-section. Cleaning up the results of topology optimization to build the beam network and setting up boundary conditions correctly for post-process simulation can also be tedious [24], especially when the optimized shape is bulky near to the boundary conditions, which is quite common. Even if such a result can be obtained in post-process, it is difficult to preserve the properties of the optimized shape. Our method does not suffer from this limitation as it is integrated within the framework of topology optimization. While these methods do perform a post extraction optimization on the beams, the geometry may radically change if nodal positions are allowed to vary and thus constraints such as regions that should be free from geometry (*keep-out* regions) must be handled separately. Moreover the topology of the network cannot change. Again, our method avoids these issues since we generate the beam network geometry in a holistic manner within the topology optimization framework.

2. Level Set-Based Topology Optimization

In this work, we formulate a topology optimization problem as the search for an optimal shape Ω_* that minimizes the objective $\mathcal{F}(\Omega)$ subject to a set of equality constraints $\mathcal{G}_i(\Omega)$:

$$\begin{aligned} \min_{\Omega} \quad & \mathcal{F}(\Omega) \\ \text{s.t.} \quad & \mathcal{G}_i(\Omega) = 0 \quad \forall\, i = 1, \ldots, k \end{aligned} \tag{1}$$

where we are satisfied with a *feasible* (i.e. constraint-satisfying) *local minimum* of Equation 1. In structural topology optimization problems, at least one of these functions is formulated in terms of the solutions of the linear elasto-static partial differential equations with respect to one or more load cases, namely a surface traction (Neumann boundary condition) applied to a region of the domain boundary $\partial\Omega$ and a prescribed displacement (Dirichlet boundary condition) applied to another region of the boundary.

We use level set-based topology optimization [8, 9] with the *augmented Lagrangian algorithm* [25] to solve the problem in Equation 1. In this iterative strategy, a shape Ω is represented as the zero-sublevel set of a piecewise smooth *level set function* ϕ defined implicitly on the background grid. Thus $\Omega := \{x : \phi(x) \leq 0\}$. The function ϕ is updated in each iteration in order to improve the optimization objective and reduce the constraint violation until a feasible, locally optimal shape is achieved. This is done by alternating between two procedures. The first procedure is to perform *shape gradient descent* on the *augmented Lagrangian* shape function, which is the following algebraic combination of the objective and constraint functions:

$$\mathcal{L}(\Omega) := \mathcal{F}(\Omega) + \sum_i \mu_i \mathcal{G}_i(\Omega) + \sum_i \frac{c_i}{2}\big(\mathcal{G}_i(\Omega)\big)^2 .$$

Here μ_i are the Lagrange multipliers and c_i are the penalty parameters. Note that unlike [25], we have separate penalty parameters for each constraint. The second procedure is to adaptively update these parameters as described in [25] (with a modification described in Section 4.3 below). After alternating between these two procedures, the shape converges to within tolerance to a locally optimal, feasible solution of Equation 1.

Here is more detail on the update of the level set function of the shape. In each iteration, we solve the standard level set Hamilton-Jacobi equation for a well-chosen normal speed function v and pseudo-time ε, and initialized at the current level set function ϕ. This has the effect of displacing the zero-contour in the normal direction by the given speed over the given pseudo-time interval. The normal speed function is chosen to be a *descent direction* for the augmented Lagrangian, which is done as follows. First, we set the values of the normal speed function on the boundary of the shape in the current iteration equal to the *negative shape gradient* $-d\mathcal{L}$ of the augmented Lagrangian. Then we set the speed function to be zero on any part of the boundary of Ω we wish to hold fixed, such as the surface patches where the non-zero surface loads and prescribed displacements of each load case are applied (denoted *keep-in regions*). Finally, we extend the speed function to a narrow band around the boundary by insisting that the extension be constant along the normal direction (achieved by solving another Hamilton-Jacobi equation). Let ϕ_ε denote the solution of the level set Hamilton-Jacobi equation, and let $\Omega_\varepsilon := \{x : \phi_\varepsilon(x) \leq 0\}$ be its zero-sublevel set. Then according to the first-order Taylor formula (see [26]), we have:

$$\mathcal{L}(\Omega_\varepsilon) = \mathcal{L}(\Omega) + \varepsilon \int_{\partial\Omega} v \, d\mathcal{L} + \mathcal{O}(\varepsilon^2) = \mathcal{L}(\Omega) - \varepsilon \int_{\Gamma} \big(d\mathcal{L}\big)^2 + \mathcal{O}(\varepsilon^2) , \qquad (2)$$

where Γ is the subset of $\partial\Omega$ that is allowed to move. This formula implies that

$\mathcal{L}(\Omega_\varepsilon) < \mathcal{L}(\Omega)$ for sufficiently small ε. We thus choose the new level set function for the next iteration to be ϕ_ε, with a suitable ε found in practice by means of a line search.

3. Beam Network Constraint

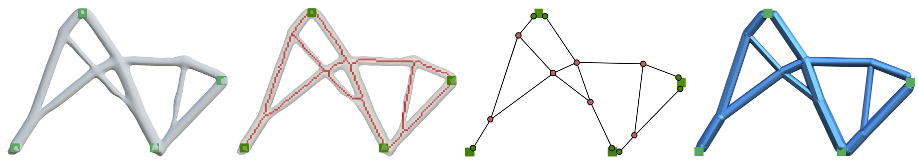

(a) LSTO geometry (b) Skeleton (c) Graph (d) Idealized beams

Figure 3: Stages of beam network extraction. Parts of the shape where boundary conditions are defined are to be kept in the final result and are shown in green.

3.1. Definition

In this section, we formulate a constraint function that can be added to any level set-based topology optimization problem that penalizes shapes that do not conform to an idealized *beam network*—that is, a shape consisting of a collection of beams connected to each other at junctions, along with some user-specified geometry. Each beam is represented as a regular geometric primitive with a uniform standard cross-section (e.g., cylinders). By driving the value of this shape constraint function to zero (subject to a tolerance) during the course of the optimization means that the converged shape Ω_* will conform to a beam network. Moreover, it should be emphasized that this beam network *emerges* during the optimization. The resulting shape then can be edited in a standard CAD system and can be easily manufactured using standard methods, such as welding or bolting extrusions together.

We can summarize the definition of the constraint function as follows. Here, we denote the signed distance function of the boundary of a shape Ω by ϕ_Ω to highlight its shape-dependence. The first step is to extract a beam network from Ω: this means to find a collection of regular primitives (and user-specified geometry) whose union best approximates Ω. We then optionally modify this beam network to account for the manufacturability of the beam network. We denote the resulting beam network by $\mathit{beam}(\Omega)$ where we again highlight the dependence on Ω.

We will fill in the details of the construction of the beam network in the next section. For now, let $\phi_{\mathit{beam}(\Omega)}$ be the signed distance function of the boundary of the beam network. We form the constraint shape function by penalizing the discrepancy between ϕ_Ω and $\phi_{\mathit{beam}(\Omega)}$. Specifically, we define

$$\mathcal{G}_{\mathit{beam}}(\Omega) := \frac{1}{N_D} \int_D H_{\mathit{beam}(\Omega)}(x) \big(\phi_\Omega(x) - \phi_{\mathit{beam}(\Omega)}(x)\big)^2 \, dx$$

where D is a fixed "universal" domain containing all geometry of interest, $N_D = \mathit{Volume}(D)L^2$ is a normalization factor and L is a characteristic size for the problem. This normalization is important since it makes the above problem unit-less and size independent and we can use a universal tolerance for the constraint in the optimization algorithm. $H_{beam(\Omega)}(x)$ is a smoothed Heaviside function that limits the influence of the constraint (see section Section 3.4 for details).

3.2. Extracting the Graph of the Beam Network

The construction of the beam network $\mathit{beam}(\Omega)$ proceeds in two steps. First, we derive from Ω a representation of the beam network as a *graph*. The nodes of the graph correspond to the junctions of the beam network, while the edges represent the individual beams. This graph can be obtained by applying a medial axis transform on Ω to obtain the skeleton which is the centerline that runs through the interior of the shape.

We use the robust skeleton method from [27] that has the important property of preserving topology, while extracting the skeleton voxels lying along the medial axis. Note that while medial sheets are common in 3D, this method is guaranteed to produce a chain of medial axis voxels by design. Since we allow the user to specify arbirarily shaped regions of Ω that will be excluded from the beam representation, we ensure that the skeleton is connected to each of these regions forming endpoints. The result is a connected set of single-width voxels between junctions and endpoints (Figure 3b) that can easily be configured to fit with other user geometry.

We then classify the skeleton voxels into three categories based on the $3\times3\times3$ neighborhood: *junctions* are defined as skeleton voxels with more than two neighbors, *endpoints* are defined as skeleton voxels with a single neighbor, and *slabs* are skeleton voxels with exactly two neighbors. We then use the center voxel of each junction as well as any endpoint voxels to form the nodes of our graph representation. Edges are placed along the connecting lines of voxels (Figure 3c) by traversing from each node through the slab skeleton voxels using a breadth-first traversal until we hit other junctions or endpoints. This graph matches the topology of the geometry but the junction voxels sometimes do not precisely lie along linear fits to the edge voxels since the skeletonization algorithm only yields an approximate medial axis. In order to match the edges, we perform a refinement step here where the coordinates of the junctions are updated using a few iterations of gradient descent. We use the following objective that minimizes the distance of each edge converging at a junction to its respective skeleton voxels, while being robust to outliers using the Geman-McClure M-estimator [28]:

$$\sum_{j=1}^{m} \frac{1}{n} \sum_{i=1}^{n} \frac{||x_i - p_j(x_i)||^2}{\sigma^2 - ||x_i - p_j(x_i)||^2}. \tag{3}$$

Here m is the number of edges at the junction, n is the number of skeleton voxels corresponding to the j^{th} edge, and $p_j(x_i)$ is the projection of x_i onto the line through edge j and σ is a constant that controls the contribution of voxels that are far from the edge. We found that $\sigma = 1$ worked well in our examples.

3.3. Defining the Geometry of the Beam Network

While in principle we could fit the geometry of the edges with curved or other complex geometry, for simplicity and ease of manufacture we model each edge as a straight cylindrical beam of uniform radius. For every edge in our graph, we set the end-points of the beam to lie at the appropriate graph junctions, and then for each voxel associated with the edge, we sample its value in the signed distance field of Ω. The radius of the beam, r is then set to:

$$r = \frac{1}{n}\sum_{i=1}^{n} -\phi_{\Omega}(x_i) + z, \tag{4}$$

where x_i is one of the n voxels associated with the beam and $z = 0.8dx$ is a constant offset that we determined experimentally since the voxels do not lie exactly on the medial axis (dx denotes the length of a voxel). The idealized beam network associated with Ω is then simply the union of all the individual beams identified above. We have denoted this beam network by $beam(\Omega)$.

To conclude, we construct a signed distance field of $beam(\Omega)$. Given a beam j in the beam network graph, we use the endpoints and radius to generate a signed distance field ϕ_b^j of a hemisphere-capped cylinder that represents the beam. We then use the Boolean union of the individual beams for the signed distance function of the beam network:

$$\phi_{beam(\Omega)}(x) := \min\{\phi_b^e(x) : e \in E\} \tag{5}$$

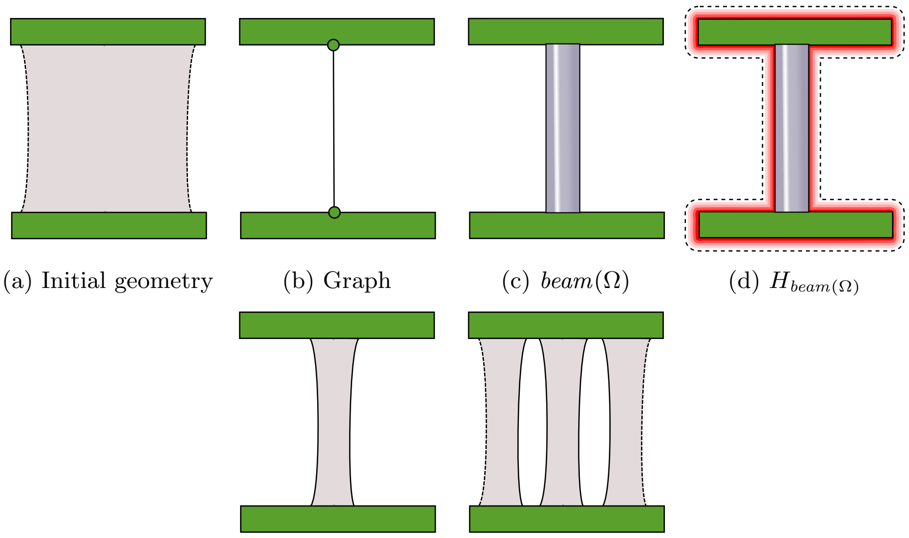
where E is the set of edges in the beam network graph (Figure 3d). Note that this approach applies no explicit modeling of the junctions, which simply form naturally from the union of all the intersecting beams.

3.4. Beam constraint influence

Recall that we restrict the influence of the beam constraint with the function $H_{beam(\Omega)}(x)$. This is a smoothed Heaviside function that applies the beam constraint up to a distance k away from $\phi_{beam(\Omega)} = 0$:

$$H_{beam(\Omega)}(x) = \begin{cases} 0 & \text{if } k - \phi_{beam(\Omega)}(x) < 0 \\ 1 & \text{if } k - \phi_{beam(\Omega)}(x) > 2 \\ \frac{k-\phi_{beam(\Omega)}(x)}{2} & \text{otherwise.} \end{cases}$$

This plays an important role when the constraint is applied to a thin 'sheet-like' structure. In this case, the skeleton will detect a single path through the sheet, resulting in a beam network with just one beam connection to replace the sheet. This beam is often far weaker than the sheet it replaces. However, when $H_{beam(\Omega)}(x)$ is included, the beam constraint does not act on the remaining geometry of the sheet, allowing additional beams to form. The end result is often that the sheet is replaced by a small network of beams instead of a single beam and is much better at resisting the loads. See Figure 4 for an illustration of the Heaviside influence function and its effect on the beam constraint. The distance k affects how densely beams can be formed within the sheet and in our examples, we select it to be equal to the maximum desired beam thickness.



(a) Initial geometry (b) Graph (c) $beam(\Omega)$ (d) $H_{beam(\Omega)}$

(e) Optimized without $H_{beam(\Omega)}$ (f) Optimized with $H_{beam(\Omega)}$

Figure 4: Illustration of the effect of Heaviside function $H_{beam(\Omega)}$. Initial 'sheet-like' geometry is shown in (a) along with its extracted graph (b) and idealized beam geometry (c). The Heaviside influence function $H_{beam(\Omega)}$ is shown with its dashed extents in (d). When optimized without $H_{beam(\Omega)}$, the whole geometry moves to match the idealized beam shape (e). When the beam constraint is limited by $H_{beam(\Omega)}$, only the geometry close to the idealized beam will match it by forming new holes and the rest will be free to support the given loading with unconstrained material (f).

3.5. Modifying the Beam Network to Account for Manufacturability

In this section, we present a methodology for incorporating manufacturability requirements into the beam network. This amounts to various modifications that can be made to the beams making up $beam(\Omega)$, and thus leading to modifications of $\phi_{beam(\Omega)}$.

Since our method builds a graph out of the beam network during topology optimization, this allows us great flexibility to control and limit the geometry of the beam network ultimately reached by our algorithm. This is in contrast to approaches such as MMC where individual elements are adjusted to form the structure and no explicit connectivity is enforced. We discuss three methods to leverage the graph to enforce desired geometrical properties: controlling sizing/thickness of the cross-sections, snapping the angle between neighboring beams to a target angle, and encouraging near-collinear beams to have the same cross-section size.

Sizing. When manufacturing a beam-based structure there are typically limits on the cross-section sizes available. Previous work in topology optimization in this area has typically attempted to enforce minimum and maximum thickness control as an explicit constraint, with some success but some caveats, often producing undesirable behaviour such as pinching at junctions [29]. Within our framework, such constraints become trivial since we can simply limit the maximum and minimum radii of the idealized beams that we use to construct $\phi_{beam(\Omega)}$ (see Section 3.3).

Angle snapping (AS). Typically the most expensive and challenging part of a beam structure are the junctions and so a highly desirable feature of any beam generating algorithm needs to be control over manufacturability of the junctions themselves and not just the arrangement of beams.

We have built in additional functionality to manage the complexity of the beam structures generated by snapping angles between neighboring beams at a junction when they are close to a specified angle such as multiples of 45° or 90°. Having junctions with regular angles allows off-the-shelf parts to be used or repeated use of a custom cast junction, saving budget [30].

We perform angle snapping by solving an optimization problem that iteratively updates the nodal positions of the graph. Given a target snapping angle $\theta_t \in [1, \pi]$, the objective function measures the difference between the current angle and the closest angle to snap to for every connected pair of edges. The energy is defined as:

$$\min \sum_{e_i \in E} \sum_{e_j \in \mathcal{N}(e_i)} \left(\left\langle \frac{e_i}{\|e_i\|}, \frac{e_j}{\|e_j\|} \right\rangle - \cos(\hat{\theta}_{ij}) \right)^2, \tag{6}$$

where E is the set of all edges in the graph, and $\hat{\theta}_{ij}$ is a multiple $k\theta_t$ (where k is an integer) closest to the current angle $\angle(e_i, e_j)$ if it is within a threshold (set to $0.1 \times \theta_t$), and $\angle(e_i, e_j)$ otherwise. Equation 6 is nonlinear in the nodal positions of the graph, and minimized using gradient descent. The angle snapping is performed in every outer loop of our optimization before constructing the idealized beams $\phi_{beam(\Omega)}$ (see Section 3.3).

Cross-section continuity (CC). We are also able to enforce continuity in cross-section sizes for edges that share a node, and where the angle between the edges approaches 180 degrees. When combined with the angle snapping, it enables 'T-junction' structures, where the two edges can be replaced with a single beam in construction. As a result, this can significantly reduce material and construction costs. This method also produces geometry that appears more natural and clean, which can be an important aesthetic consideration.

We treat the cross-section continuity as an iterative smoothing problem considering the target radii of the idealized beams (Equation 4). Let r_i denote the radius of the beam corresponding to the edge $e_i \in E$ in the graph. We iteratively update r_i while considering the angle between the edge e_i and all its neighbors $\{e_j \in \mathcal{N}(e_i)\}$ as follows:

$$r_i \leftarrow r_i + \eta \left(-W_i r_i + \frac{1}{W_i} \sum_{e_j \in \mathcal{N}(e_i)} r_j \ \sigma \left(\left\langle \frac{e_i}{\|e_i\|}, \frac{e_j}{\|e_j\|} \right\rangle \right) \right). \tag{7}$$

This update equation is reminiscent of Laplacian smoothing, except that we choose the weights W_i such that the most influence comes from neighboring edges that are almost collinear to the current edge. The logistic sigmoid function is used the compute the weights in range $[0, 1]$:

$$\sigma(x) = \frac{m}{1 + e^{k(x - x_0)}},$$

and is parametrized by m that denotes the maximum value of the function, k which controls the steepness of the function, and x_0 that controls where the midpoint of the function lies at. We set $m = 0.5$, $k = 50$, and $x_0 = -0.9$ so that the function output rapidly drops as the dot product of the neighboring edges goes above -0.9 ($< 155°$ approximately).

$$W_i = \sum_{e_j \in \mathcal{N}(e_i)} \sigma \left(\left\langle \frac{e_i}{\|e_i\|}, \frac{e_j}{\|e_j\|} \right\rangle \right)$$

is the sum of weights computed from all the neighboring edges of e_i. At each outer loop iteration, we update edge radius iteratively with a step size η until convergence before constructing the idealized beams $\phi_{beam(\Omega)}$ (see Section 3.3).

4. Beam Network Topology Optimization

4.1. Problem Formulation

While the beam constraint function is very general and can be applied in many topology optimization problems, we will focus on applying it to structural optimizations where we combine it with a *structural compliance* objective and a volume constraint. In other words, we will solve the following version of Equation 1:

$$\begin{aligned} \min_{\Omega} \quad & \mathcal{F}(\Omega) \\ \text{s.t.} \quad & \mathcal{G}_{vol}(\Omega) = 0 \\ & \mathcal{G}_{beam}(\Omega) = 0 \end{aligned} \tag{8}$$

where the objective and constraint functions are defined as follows.

- The structural compliance of Ω with respect to a given load case is

$$\mathcal{F}(\Omega) := \int_\Omega \sigma(u_\Omega) : e(u_\Omega)$$

where $\sigma(u_\Omega)$ and $e(u_\Omega)$ are respectively the stress and strain tensors of the displacement u_Ω. This satisfies the linear elasto-static problem

$$\begin{aligned} -\mathrm{div}(\sigma(u_\Omega)) &= f && \text{in } \Omega \\ u_\Omega &= 0 && \text{on } \Gamma_D \\ n \cdot \sigma(u_\Omega) &= g && \text{on } \Gamma_N \\ n \cdot \sigma(u_\Omega) &= 0 && \text{on } \partial\Omega \setminus (\Gamma_D \cup \Gamma_N), \end{aligned}$$

where f is the prescribed body load, g is the prescribed surface traction on Γ_N, and fixity is prescribed on Γ_D. We denote $\Gamma := \Gamma_D \cup \Gamma_N$.

- The volume constraint function is simply

$$\mathcal{G}_{vol}(\Omega) := \int_\Omega 1 - V_{targ}$$

where V_{targ} is the target volume fraction.

- The beam network constraint function is as defined in the previous section, and may include the modifications to account for the manufacturability of the beam network.

Since we will use the augmented Lagrangian algorithm to find a locally optimal and feasible solution of (8), we need the shape gradients of the objective and constraint functions. It is well-known (see e.g., [8, 9]) that the shape gradient of the compliance at the shape Ω is

$$d\mathcal{F} = -\sigma(u_\Omega) : e(u_\Omega)$$

on the boundary $\partial\Omega$, while the shape gradient of the volume is

$$d\mathcal{G}_{vol} = 1$$

on the boundary $\partial\Omega$. It remains to derive a formula for the shape gradient of the beam network constraint. Since it turns out to be very difficult to derive an exact formula, we settle for a good approximation in practice.

4.2. Approximate Shape Gradient of the Beam Network Constraint

Recall that we have defined the beam network constraint as

$$\mathcal{G}_{beam}(\Omega) = \frac{1}{N_D} \int_D H_{beam(\Omega)}(x)[\phi_\Omega(x) - \phi_{beam(\Omega)}(x)]^2 dx,$$

where ϕ_Ω and $\phi_{beam(\Omega)}$ are the signed distance functions of Ω and of the idealized beam network *beam*(Ω) respectively. We have highlighted the dependence on Ω in this shape function.

We observe that the shape derivative of ϕ_Ω can be found by straightforward though somewhat involved geometric calculations (see e.g., [26, 31]) but that the shape derivative of $\phi_{beam}(\Omega)$ will be much more difficult to calculate or may not even exist. This is because we have not phrased the relationship between Ω and *beam*(Ω) in a shape-differentiable manner, and perhaps we can expect topological changes to occur in the graph underlying *beam*(Ω) even when Ω changes smoothly. Rather than try to resolve these issues, we instead find a shape-differentiable approximation of $\mathcal{G}_{beam}$ whose shape derivative we will use in the shape gradient descent phase of the augmented Lagrangian algorithm. This approximation is constructed adaptively and improves throughout the course of the augmented Lagrangian algorithm. Of course, there is no theoretical guarantee that we end up with a descent direction for the Lagrangian in this way. However, in practice, we always observe a decrease of the Lagrangian for sufficiently small pseudo-time (i.e., the line search to choose the pseudo-time is always successful, except when the shape has converged to within a tolerance to the locally optimal, feasible solution).

Our approximation is defined as follows. In each "inner" iteration of the augmented Lagrangian algorithm (i.e., when the Lagrange multipliers and penalty parameters are kept fixed) we freeze the beam network to be the beam network corresponding to the shape at the first inner iteration, which we denote here by Ω_{init}. Now we define the approximate beam network constraint for this inner iteration as

$$\widetilde{\mathcal{G}}_{beam}(\Omega) := \frac{1}{N_D}\int_D H_{beam(\Omega_{init})}(x)[\phi_\Omega(x) - \phi_{beam(\Omega_{init})}(x)]^2 dx \tag{9}$$

Then when this inner iteration is complete, we update $\widetilde{\mathcal{G}}_{beam}$ by replacing Ω_{init} with the shape reached at the end of this inner iteration.

The approximate beam network constraint $\widetilde{\mathcal{G}}_{beam}$ is shape-differentiable since it only depends on Ω through the signed distance function ϕ_Ω. Also, observe that $\widetilde{\mathcal{G}}_{beam}$ has the mathematical form

$$\widetilde{\mathcal{G}}_{beam}(\Omega) := J(\phi_\Omega)$$

where J is the functional defined on *any* function $\phi : D \to \mathbb{R}$ by

$$J(\phi) := \frac{1}{N_D}\int_D H_{beam(\Omega_{init})}(x)\big[\phi(x) - \phi_{beam(\Omega_{init})}(x)\big]^2 dx\,.$$

Therefore, we can use the methodology presented in [32] to compute the shape derivative of $\widetilde{\mathcal{G}}_{beam}$.

In [32], the authors derive an elegant formula linking the derivative of $J(\phi_\Omega)$ with respect to variations of Ω to the derivative of J with respect to arbitrary variations of ϕ. We denote this latter derivative by $d_\phi J$ and in our case we have

$$d_\phi J := \frac{2H_{beam(\Omega_{init})}}{N_D}\big[\phi - \phi_{beam(\Omega_{init})}\big]\,.$$

To do this, they view the derivative of $J(\phi_\Omega)$ with respect to variations of Ω as the derivative of $J(\phi)$ with respect to the constrained set of variations of ϕ maintaining the signed distance property of ϕ. The formula they derive is essentially a projection of $d_\phi J$ onto the tangent space of this constrained set of variations. Applied to our case, their formula (or rather its three-dimensional counterpart) yields

$$d\widetilde{\mathcal{G}}_{beam}(x) = -\int_{l(x)} d_\phi J(x(r)) \left|1 - r\, K_{\partial\Omega}(x_0) + r^2\, G_{\partial\Omega}(x_0)\right| dr \qquad \forall\, x \in D$$

where $l(x)$ is the line segment passing through x and the nearest point to x on $\partial\Omega$ which we have denoted x_0 above. This line segment extends from the medial axis of $\partial\Omega$ to the boundary of D and we have denoted its arc-length parameterization by $x(r)$ above. Finally, $K_{\partial\Omega}(x_0)$ and $G_{\partial\Omega}(x_0)$ are the mean and Gauss curvatures of $\partial\Omega$ at x_0, respectively. Note that the formula for $d\widetilde{\mathcal{G}}_{beam}$ is valid in all of D and is in fact constant along lines normal to $\partial\Omega$. Therefore the normal extension step described in Section 2 for constructing the speed function used in the level set Hamilton-Jacobi equation is not needed for $d\widetilde{\mathcal{G}}_{beam}$. The reference [32] also provides a computationally efficient way to evaluate this integral, which extends straightforwardly to three dimensions.

4.3. Implementation Details of the Augmented Lagrangian Algorithm

The augmented Lagrangian function can be minimized by performing a series of gradient descent loops ("outer" loops) to convergence while holding the Lagrange multipliers and penalty values constant for each loop and updating them before the start of the next loop [25]. This approach still requires initial values for the parameters and tuning these parameters is often cited as a weakness of the algorithm. As mentioned in Section 2, one of the contributions of this work is a reliable parameter initialization for Augmented Lagrangian optimization in the context of Topology Optimization. We propose the following initialization for the penalty parameter associated to the constraint function $\mathcal{G}$:

$$c_{\text{init}} = \frac{w|d\mathcal{F}(\Omega)|}{|d\mathcal{G}(\Omega)|d\mathcal{G}(\Omega)}, \tag{10}$$

where w is a constraint-specific weighting. This penalty parameter initialization has the effect of normalizing the shape gradients of the constraints to the shape gradient of the objective such that all the constraints and the objective have an effect on the initial shape change equivalent to their weighting w. We initialize the Lagrange multipliers to zero and from there the optimization progresses and during parameter updates, the strength of the constraint penalties are increased as needed and the Lagrange multipliers are updated according to [25].

In the Augmented Lagrangian method, each constraint has a user-defined initial tolerance t_0 and a final tolerance t_*. The constraint tolerances are tightened up to the final tolerance at the end of each inner loop whenever a constraint is determined to be within the current tolerance using a tightening factor $\kappa = 0.5$: At the end of each inner loop, t is updated as follows:

$$t_{i+1} = \begin{cases} \max\,(t_*, \kappa t_i), & \text{if } \mathcal{G}(\Omega) \le t_i \\ t_i, & \text{if } \mathcal{G}(\Omega) > t_i \end{cases}$$

We detect convergence in an inner loop when the Lagrangian is reducing by less than 2% between consecutive iterations. Convergence in the outer loop is determined by a converged inner loop and feasibility with respect to all constraints ($\mathcal{G}(\Omega) < t_*$).

4.4. Topology Optimization Algorithm

In this section we put together all of the pieces described above and summarize the final algorithm for beam network topology optimization (Algorithm 1).

We perform an Augmented Lagrangian Optimization of our structural problem using a compliance objective with a volume constraint and the beam constraint. At the start of the optimization we extract the idealized beam network (Equation 5), then evaluate the objectives and constraints in order to set the penalty parameters using (Equation 10). By default we initialize the weight for the volume constraint $w_{vol} = 1$ and the beam constraint weight $w_{beam} = 0.3$ in order to prevent the initial beam network from significantly affecting the convergence. This is important because the geometry tends to be bulky and not beam-like in the early stages, so our extracted beam network is not reliable. Then we begin an "inner" loop where we compute the shape derivatives, update the shape, re-evaluate the Lagrangian. This inner loop repeats until convergence and then we check to see if the constraints have been satisfied. If they are not satisfied, we update the Lagrange multipliers and penalty parameters, extract the beam network and begin a new inner loop. This process repeats until we converge with satisfied constraints. As the optimization progresses, the geometry becomes more beam-like and the extracted beam network becomes more reliable. In turn the penalty parameter for the beam constraint c_{beam} increases, and the geometry is gradually forced to align to the idealized beam network while still minimizing the overall compliance and maintaining the desired volume.

5. Results

We now demonstrate our method on five structural topology optimization problems and compare them against the results obtained without our beam constraint. In all cases, we initialize the geometry with a large block with regularly spaced holes to facilitate quicker convergence. We found that just as with standard continuum topology optimization approaches, variation of the initial geometry may lead to convergence at different local optima.

Algorithm 1: Beam Network Topology Optimization

```
do
    Extract idealized beam network;
    Apply optional modifications to account for manufacturability;
    if first iteration then
        Evaluate objective and constraints;
        Initialize Lagrange multipliers and penalty parameters (10);
    end
    do
        Compute shape derivatives and speed field;
        Update shape by solving Hamilton-Jacobi equation;
        Evaluate objective and constraints;
        Check for convergence;
    while not converged;
    Check constraints;
    if constraints not satisfied then
        Update Lagrange multipliers and penalty parameters;
    end
while constraints not satisfied;
```

5.1. Perpendicular Truss

In Figure 5 we can see a naïve topology optimization of a perpendicularly loaded truss configuration with a target volume of 20%. The loads are each 10N and the material has Young's modulus 193000 MPa, and a Poisson ratio of 0.3. The result is actually far more 'beam-like' than many topology optimized results, however it still presents several challenges for manufacture (see highlighted regions in Figure 5). If the part is to be made from a set of beams, often there is a limitation on the shape and minimum and maximum size of the cross sections and the naïve optimization has no control over these features. Our beam constraint directly addresses this by forcing the topology optimized geometry to conform to a network of beams upon which we enforce these limitations, namely uniform cross section and a minimum and maximum cross section size. By applying the angle snapping constraint to the beam network we can simplify the resulting beam network such that regular angles are enforced between pairs of beams that are close to the snapping angle. Given this snapping, beams are more likely to continue through junctions (when snapping is a multiple of 180°); and when we enforce the cross section continuity, this effect is further enhanced. The result is that fewer, longer beams can be used when manufacturing, saving cost and construction time.

Figure 6 (c)–(f) shows a comparison of results from our method against the naïve topology optimization in Figure 6 (b). Each of these results is optimized with the same volume constraint and minimize compliance subject to the beam constraint with different options. First we show the raw beam constrained

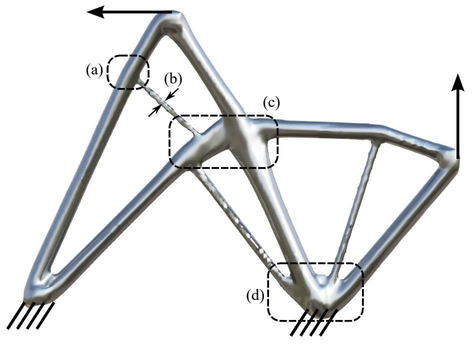


Figure 5: A naïve topology optimized result of the perpendicularly loaded truss with annotated regions that potentially make manufacture with a beam network challenging: (a) This joint is close to a 'T-junction' but it requires three beams joined together instead of two if it were an actual 'T'. (b) This element is thin and may be below the minimum thickness of available beams. (c) This region does not easily match any uniform beam and an approximation may reduce stiffness or increase overall weight. (d) This region has overgrown the connection point so the actual manufactured approximation would likely be less stiff.

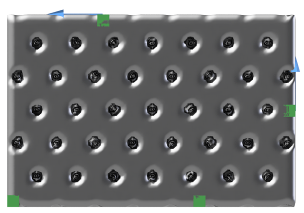
mesh result in (c) and note that a minimum beam size has been enforced and the geometry has converged to truly beam-like linear segments with consistent cross sections, and the exported CAD geometry for this result is shown in (d). In Figure 6 (e) we introduced angle snapping to 45° and we see that the central beam has been snapped, introducing more regularity to the result. Finally, Figure 6 (f) shows the combined result with cross-section continuity. In this case, several 'T-junctions' are evident and the cross sections have been matched across the T's, improving manufacturability.

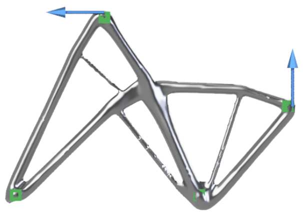
We show the impact on the compliance of these constraints in Table 1. In this case we can see that the application of the beam constraint increases the compliance, which is a more fair reflection of the actual stiffness of the built structure which is what ultimately matters.

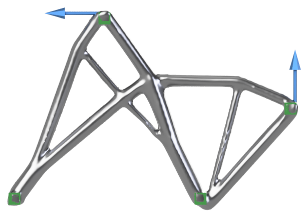
Optimization method	Compliance	Volume fraction
Volume constraint only	0.00511	0.204
(+) Beam constraint	0.00547	0.209
(+) 45° Angle snapping	0.00619	0.205
(+) Cross-section continuity	0.00725	0.236

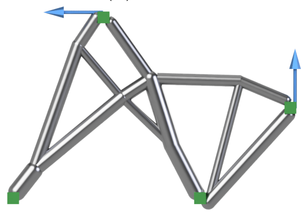
Table 1: Compliance values and volume fraction for the optimized Perpendicular Truss beam geometry that produced the different solutions in Figure 6.

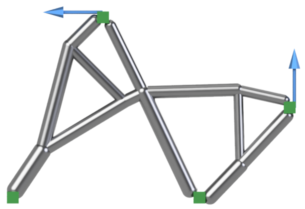
We also evaluated the performance impact of the graph extraction for this use case in Table 2. We can see that even if we were to extract the graph at every iteration, the computation time is a small fraction of the solve and overall


(a) Setup


(b) Naïve TO (levelset)


(c) BeNTO (levelset)


(d) BeNTO (CAD)


(e) BeNTO+45°AS (CAD)

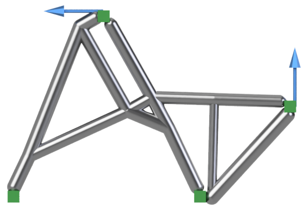

(f) BeNTO+90°AS, CC (CAD)

Figure 6: Perpendicular Truss: (a) initial shape (gray) seeded with holes, and perpendicular loads defined on the top two green boxes, while the bottom two green boxes are fixed. Optimized mesh geometry with (b) only volume constraint, (c) also with beam constraint enforcing thickness control, (d) CAD optimized geometry from (c), (e) CAD optimized geometry with beam constraint enforcing thickness control and 45° angle snapping, and (e) CAD optimized geometry with beam constraint enforcing thickness control, 90° angle snapping and cross-section continuity.

iteration time. As it is, we only perform the graph extraction at each outer loop (on average once every six iterations). Even when considering angle snapping and cross-section continuity the overhead for graph extraction is still quite small compared to one whole iteration.

5.2. Pylon

In this section we compare results for a pylon structure, often modeled in the ground structure method literature. The structure is fixed to the ground and has two load cases with loads applied perpendicularly to the tip of the structure (see Figure 7 (a)). The loads are each 1N and the material has Young's modulus 193000 MPa, and a Poisson ratio of 0.3. We show the topology optimization result with only the volume constraint (set to 24%) in Figure 7 (b) and note that there is a smooth blending between beam structures and even those regions that appear beam-like have highly non-uniform cross-sections. Our beam constraint result enforces regular cross-sections as expected, however the beam layout and angles are quite irregular (Figure 7 (c)–(d)). When angle snapping and cross-section continuity are enforced we get a very regular structure in Figure 7 (e)–(f). In Table 3 we can see the trade-off between manufacturability and compliance.

Optimization Method	Mean graph extraction time	Mean solve time	Mean iteration time
BeNTO	0.69s	15.83s	18.34s
BeNTO (+) AS (+) CC	1.25s	16.08s	19.17s

Table 2: Timings for the stages of the optimization of the perpendicular truss.

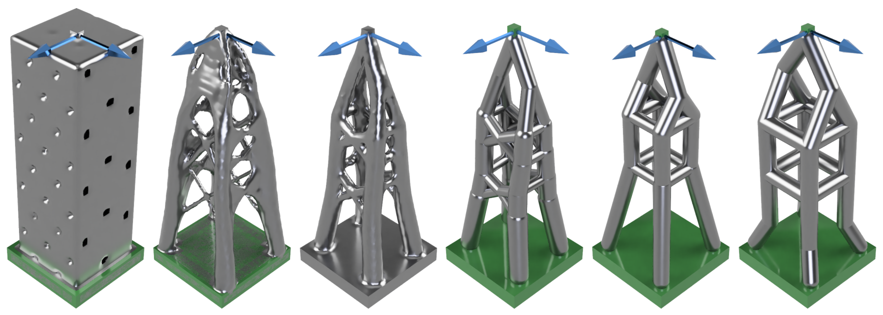


Figure 7: Pylon optimized geometry starting from (a) a cubical shape seeded with holes, and two orthogonal loads defined at the top box while the bottom plate is fixed. Optimized with (b) only volume constraint, (c) also beam constraint, (d) the CAD exported beam network with additionally including 45° angle snapping and (e) the CAD exported beam network with 90° angle snapping and cross-section continuity.

5.3. Torque

Here we modeled a torque-resistant structure that has four blocks, each with traction loads rotated by 90° from their neighbors (Figure 8 (a)). The loads are each 1000N and the material has Young's modulus 193000 MPa, and a Poisson ratio of 0.3. The structure is also fixed to the ground plane and we minimize compliance subject to a target volume fraction of 25%. The optimized result in Figure 8 (b) with only the volume constraint produces a structure with a combination of 'beam-like' structures as well as highly free-form structures blended together. This is actually a very challenging example since the converged result does not have an obvious beam fit for many parts of the structure. However, when applying our method we are able to extract a manufacturable beam network that supports the loads (Figure 8 (c)–(d)). Rotational symmetry is not enforced here and so the symmetry is lost as each side converges to a different local minimum that still supports the loads. We also show the results in Figure 8 (e)–(f) when optimizing without the Heaviside function $H_{beam}(x)$ in Equation 9. We can see in (Table 4) that the beam network in Figure 8 (d), using $H_{beam}(x)$ is much more stiff, compared to the beam network in Figure 8 (f) despite a similar volume.

Optimization method	Compliance	Volume fraction
Volume constraint only	0.00261	0.233
(+) Beam constraint	0.00491	0.233
(+) 45° Angle snapping	0.00430	0.249
(+) 90° Angle snapping, cross-section continuity	0.00476	0.244

Table 3: Compliance values and volume fraction for the optimized Pylon geometry that produced the different solutions in Figure 7.

Optimization method	Compliance	Volume fraction
Volume constraint only	21.5	0.249
(+) Beam constraint, 45° angle snapping, cross-section continuity	38.6	0.238
(+) Beam constraint, 45° angle snapping, cross-section continuity, without $H_{beam}(x)$	193	0.225

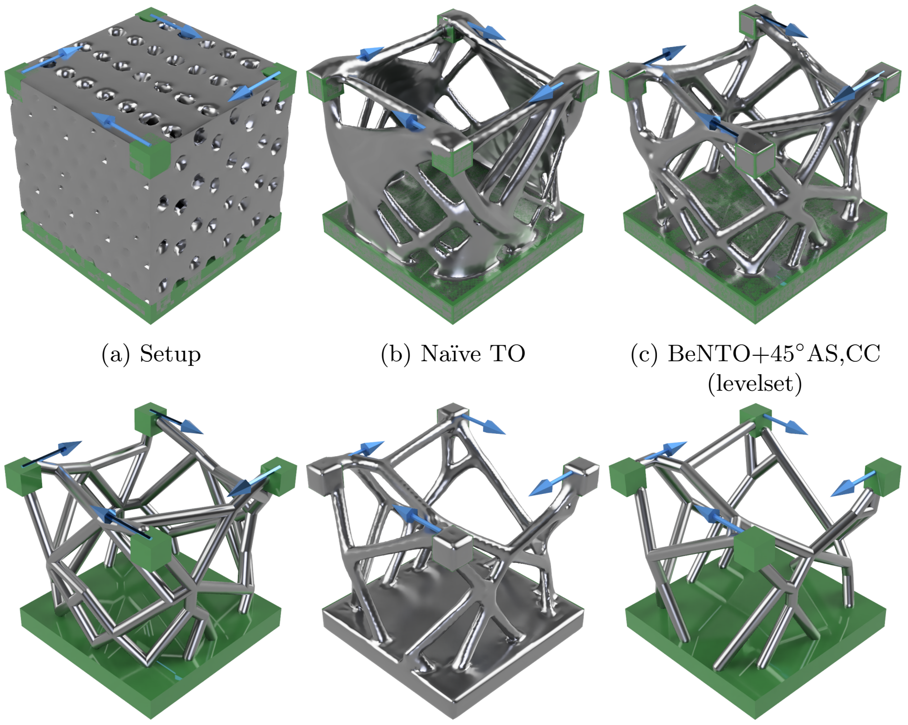
Table 4: Compliance values and volume fraction for the optimized torque geometry that produced the different solutions in Figure 8.

5.4. Bridge

Here we modeled a footbridge structure with a distributed vertical load (10000N) over its surface and is fixed at bars at the two ends below the bridge (Figure 9). The material has Young's modulus 193000 MPa, and a Poisson ratio of 0.3. Material is prevented from obstructing the walk-way and the domain voxel size is 280×80×78. This is a challenging example due to its size and the density of fine features needed to support the walkway. In Figure 1 we can see the optimization result with only the volume constraint compared with the CAD exported result from the beam constraint. Both results ensure that the material does not interfere with the keep-out regions and the beam-constrained result reduces the complexity of the support structure and produces a manufacturable network of beams. Again there is a trade-off in the compliance (Table 5), but the simplicity and manufacturability of the beam constrained result can allow for easy editing and improvement on the way toward a final design.

Optimization method	Compliance	Volume fraction
Volume constraint only	0.295	0.123
(+) Beam constraint	1.054	0.115

Table 5: Compliance and volume fraction for the optimized bridge geometry that produced the different solutions in Figure 1.



(a) Setup (b) Naïve TO (c) BeNTO+45°AS,CC (levelset)

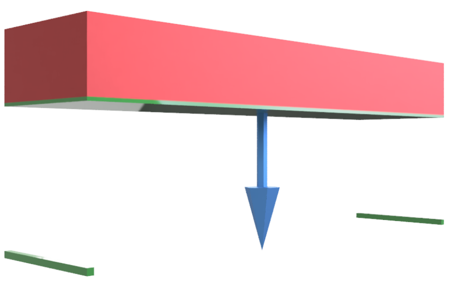
(d) CAD result of (c) (e) BeNTO+45°AS,CC w/o $H_{beam}(x)$ (f) CAD result of (e)

Figure 8: Torque: (a) initial shape (gray) seeded with holes, and loads defined on the top four green boxes, while the bottom plate is fixed. Optimized mesh geometry with (b) only volume constraint, (c) also beam constraint with angle snapping and cross-section continuity, (d) the CAD exported beam network, (e) the same settings as (c) but without the Heaviside function $H_{beam}(x)$, (f) the CAD exported beam network.

5.5. Pavilion

We show how our method can be applied to architectural scale structures with a pavilion example in Figure 10. The pavilion is designed to be supported by three fixed plates and to resist loading from two display screens, whose frames are set as keep-ins, along with several points in space where additional loads are applied. The loads are each 100N and the material has Young's modulus 193000 MPa, and a Poisson ratio of 0.3. There is also a large keep-out region that exposes the pavilion to viewers and those interacting with the display screens.

Due to the interaction with the keep-out region, the naïve topology optimization result produces some large sheet-like structures that hug the bounds of the keep-out as well as curved beams and complex junctions, making design for manufacture challenging. After applying the beam constraint we are able to


(a) Setup

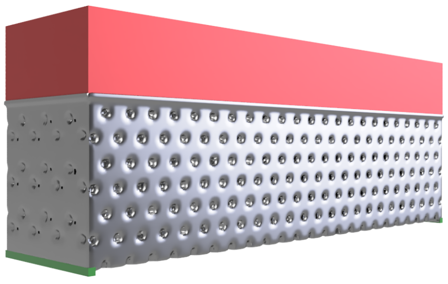

(b) Initial geometry

Figure 9: Bridge: (a) A distributed load is defined on the top surface and green surfaces are all keep-in geometries. The red region defines a keep-out that the optimized shape cannot overlap with. (b) Initial shape (gray) seeded with holes. Results comparing naïve topology optimization and our beam network topology optimization are shown in Figure 1.

generate a much cleaner beam-like result that can be sent for manufacturing. Table 6 compares the final compliance and volume fractions obtained by our method with topology optimization using only volume constraint.

Optimization method	Compliance	Volume fraction
Volume constraint only	0.190	0.0157
(+) Beam constraint	0.290	0.0158

Table 6: Compliance and volume fraction for the optimized pavilion geometry.

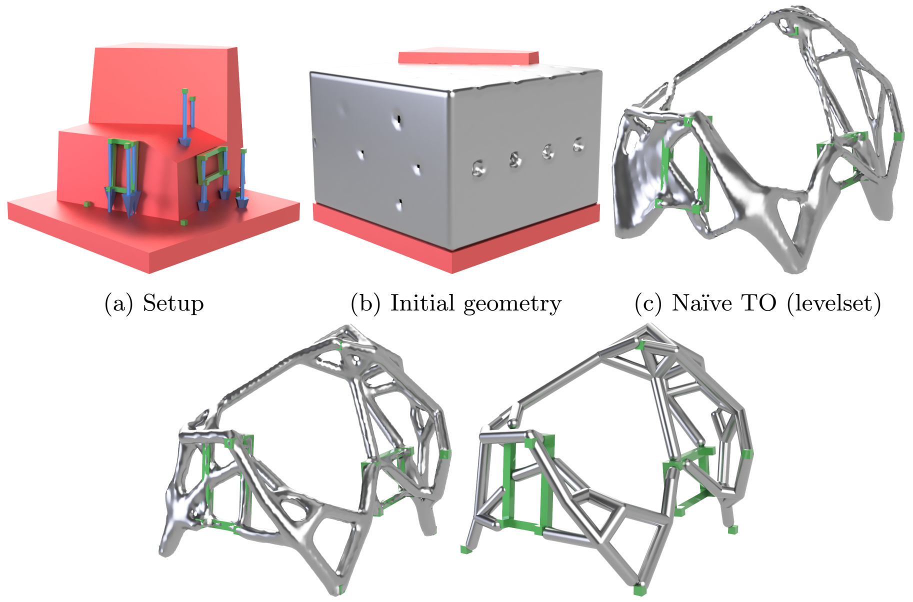
5.6. GE Bracket

To demonstrate how our method behaves on a real-world mechanical part, we present beam-constrained optimization results in Figure 11 from the GE aircraft bracket challenge. Additionally, we note that the user geometry can have arbitrary shapes and the beam network is generated as the required additional support structure. For these results, we used the four load cases specified in [33] with Young's modulus of 113800 MPa and a Poisson ratio of 0.342. We achieved a volume fraction of 0.185 with a compliance value of 189358. We used angle snapping of 90 degrees with cross-section continuity.

5.7. L-Bracket

To demonstrate the flexibility of our method we show that the beam constraint can be paired just as effectively with another objective function. In this case, we replace $\mathcal{F}(\Omega)$ with a p-norm stress objective from [34] such that:

$$\mathcal{F}(\Omega) := \left(\int_{\Omega} \sigma_{vm}^{p} d\Omega\right)^{\frac{1}{p}},$$

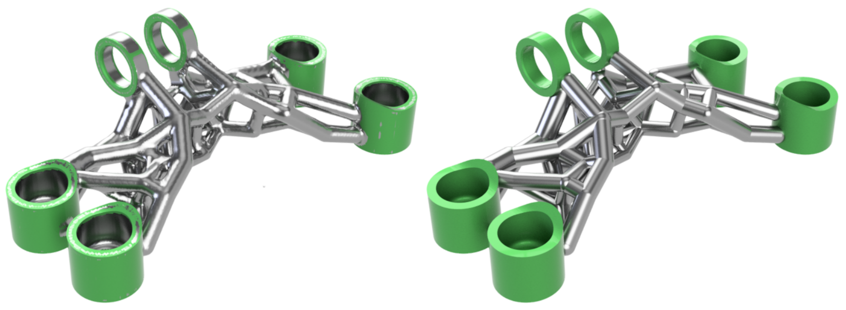


(a) Setup (b) Initial geometry (c) Naïve TO (levelset)

(d) BeNTO (levelset) (e) BeNTO (CAD)

Figure 10: Pavilion: (a) Loads applied to display screen frames as well as point loads on keep-in regions shown in green. The keep-out region is shown in red and marks the portion of the domain that the geometry cannot overlap with. (b) Initial geometry in the portion of the domain that excludes the keep-out regions seeded with holes. (c) Naïve topology optimization result with sheets and bulky parts. (d) Our beam network topology optimization result. (e) CAD result of (d).

where p is the p-norm parameter used to approximate the $\max(\cdot)$ operator and σ_{vm} is the von Mises stress. The L-bracket problem setup, material properties and loading are the same as in [34], but we used a 3D domain size of $100 \times 100 \times 8$ and we used a value of $p = 6$. Since the beam constraint acts as a regularizer we found that the least-squares smoothing of the von Mises stress described in [34] was not necessary for this experiment. In Figure 12 we can see that valid beam networks are generated for both the compliance and p-norm stress objectives. Also note that without the p-norm stress objective we get a stress concentration at the inside corner of the L-bracket, just as in unconstrained compliance-based topology optimization. However, it is pleasing to see that the p-norm stress objective combined with the beam constraint produces a result without stress concentrations, spreading the stress across several beams.



(a) BeNTO (levelset) (b) BeNTO (CAD)

Figure 11: GE Bracket: (a) Our beam network topology optimization result. (b) CAD result of (a).

6. Conclusion

We have proposed a new framework for beam network topology optimization, that works by coupling level set-based topology optimization with a beam constraint to drive the evolving geometry toward an idealized beam network. By including the beam constraint within the optimization, rather than as a post-process, we can take advantage of geometric constraints such as keep-in and keep-out regions and allow the compliance minimization to compensate for the limitations of the beam representation.

Since our beam constraint tracks the underlying graph of the evolving beam network, we can enforce manufacturability requirements, for example by snapping to user-defined angles between neighboring beams and by enforcing cross-section continuity within the beam network. All these factors help to produce more regular, optimized geometry that can be exported as a set of editable primitives into CAD software. There, they can easily be incorporated into user assemblies for manufacture.

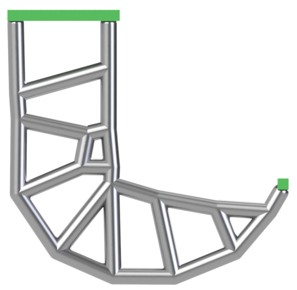
We hope that this work will contribute to more practical usage of topology optimization techniques and save designers' time when it comes to building their optimized structures.

Future work and limitations

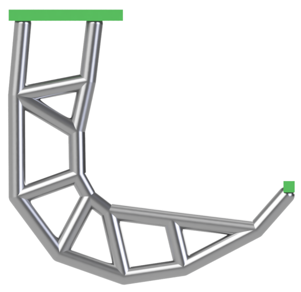
Since we rely on the background grid for structural simulation, the grid spacing should be $\frac{1}{4}$ or less than the minimum beam diameter otherwise the simulation of slender beams will be inaccurate.

We would like to enforce true symmetry in the graph structure when reflection or rotation symmetry is desired. This would require special treatment of nodes lying on the reflection planes.

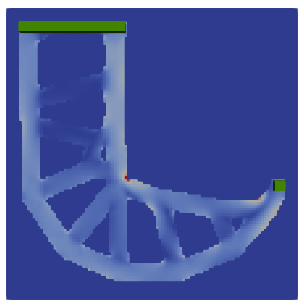
We also see the potential for supporting other cross-section types such as I-beam, square, angled or hollow tubes. We could fit the cross-section to the current geometry to infer the best local orientation of the cross-section, and analyze the local bending of the beam. Curved extrusions could also be enabled



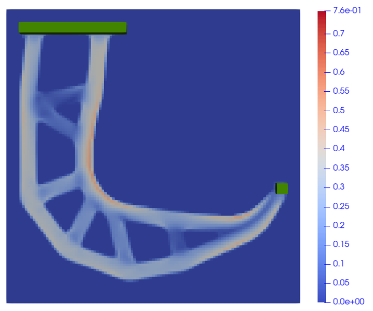
(a) BeNTO w/compliance



(b) BeNTO w/p-norm stress



(c) von Mises Stress distribution of compliance optimized shape



(d) von Mises Stress distribution of p-norm stress optimized shape

Figure 12: L-Bracket: (a) Our beam network topology optimization CAD result with compliance. (b) Our beam network topology optimization CAD result with p-norm stress. (c) Slice of von Mises stress for the compliance result in (a). (d) Slice of von Mises stress for the p-norm stress result in (b).

by fitting a curve to the slab voxels along the edge, and sweeping the cross-section along the fitted curve. Additionally, it would be useful to be able to identify where welded plates or other primitives might enhance the stiffness of the structure, especially at joints. We could detect regions that are not well fit by the beam network and instead replace those regions with a parameterized plate or combination of a plate and beams to form a gusset.

Currently, the angle snapping energy minimization (Equation 6) is solved by gradient descent and may get trapped in local minima for large graphs such as in the Bridge, Pavilion and GE Bracket. To alleviate this, we plan to explore alternative optimization strategies such as simulated annealing or momentum-based approaches.

The initial weight of the beam network constraint determines how soon the constraint begins to control the optimization procedure. If the optimization has not converged sufficiently before the beam network is enforced, sub-optimal results are likely. While we did not observe this in our examples, a sufficiently complex topology optimization problem may need a weight adjustment. We thus plan to investigate how to automate this balancing so that the user has no need to tune the parameter.

References

[1] W. S. Dorn, R. E. Gomory, H. J. Greenberg, Automatic design of optimal structures, Journal De Mecanique 3 (1964) 25–52.

[2] L. He, M. Gilbert, Rationalization of trusses generated via layout optimization, Structural and Multidisciplinary Optimization 52 (4) (2015) 677–694. doi:10.1007/s00158-015-1260-x.

[3] P. O. Ohlbrock, P. D'Acunto, A computer-aided approach to equilibrium design based on graphic statics and combinatorial variations, Computer-Aided Design 121 (2020) 102802. doi:10.1016/j.cad.2019.102802.

[4] I. Mirtsopoulos, C. Fivet, Grammar-based generation of trusses within non-convex domains, in: Proceedings of the International fib symposium on conceptual design of structures, 2019, pp. 193–200.

[5] C. Jiang, C. Tang, H.-P. Seidel, R. Chen, P. Wonka, Computational design of lightweight trusses, Computer-Aided Design 141 (2021) 103076. doi:10.1016/j.cad.2021.103076.

[6] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Computer Methods in Applied Mechanics and Engineering 71 (2) (1988) 197–224. doi:10.1016/0045-7825(88)90086-2.

[7] O. Sigmund, K. Maute, Topology optimization approaches: A comparative review, Structural and Multidisciplinary Optimization 48 (6) (2013) 1031–1055. doi:10.1007/s00158-013-0978-6.

[8] G. Allaire, F. Jouve, A.-M. Toader, Structural optimization using sensitivity analysis and a level-set method, J. Comput. Phys. 194 (1) (2004) 363–393. doi:10.1016/j.jcp.2003.09.032.

[9] M. Y. Wang, X. Wang, D. Guo, A level set method for structural topology optimization, Computer Methods in Applied Mechanics and Engineering 192 (1) (2003) 227–246. doi:10.1016/S0045-7825(02)00559-5.

[10] B. M. Weiss, J. M. Hamel, M. A. Ganter, D. W. Storti, Data-Driven Additive Manufacturing Constraints for Topology Optimization, Journal of Manufacturing Science and Engineering 143 (2), 021001 (10 2020). doi:10.1115/1.4048264.

[11] J. Zou, Y. Zhang, Z. Feng, Topology optimization for additive manufacturing with self-supporting constraint, Structural and Multidisciplinary Optimization 63 (5) (2021) 2341–2353. doi:10.1007/s00158-020-02815-w.

[12] N. Morris, A. Butscher, F. Iorio, A subtractive manufacturing constraint for level set topology optimization, Structural and Multidisciplinary Optimization 61 (4) (2020) 1573–1588. doi:10.1007/s00158-019-02436-y.

[13] M. Langelaar, Topology optimization for multi-axis machining, Computer Methods in Applied Mechanics and Engineering (2019).

[14] Y. Wang, Z. Kang, Structural shape and topology optimization of cast parts using level set method, International Journal for Numerical Methods in Engineering 111 (13) (2017) 1252–1273. doi:10.1002/nme.5503.

[15] C. Wang, B. Xu, Q. Meng, J. Rong, Y. Zhao, Topology optimization of cast parts considering parting surface position, Advances in Engineering Software 149 (2020) 102886. doi:10.1016/j.advengsoft.2020.102886.

[16] L. E. Murr, A metallographic review of 3d printing/additive manufacturing of metal and alloy products and components, Metallography, Microstructure, and Analysis 7 (2) (2018) 103–132. doi:10.1007/s13632-018-0433-6.

[17] Haas 5-axis CNC machining catalog, https://www.haascnc.com/machines/multi-axis/5-axis-mills.html, accessed: 2022-06-06.

[18] A. Vevers, A. Kromanis, E. Gerins, J. Ozolins, Additive manufacturing and casting technology comparison: Mechanical properties, productivity and cost benchmark, Latvian Journal of Physics and Technical Sciences 55 (2018) 56–63. doi:10.2478/lpts-2018-0013.

[19] X. Guo, W. Zhang, W. Zhong, Doing Topology Optimization Explicitly and Geometrically—A New Moving Morphable Components Based Framework, Journal of Applied Mechanics 81 (8) (05 2014). doi:10.1115/1.4027609.

[20] W. Zhang, J. Yuan, J. Zhang, X. Guo, A new topology optimization approach based on moving morphable components (MMC) and the ersatz material model, Structural and Multidisciplinary Optimization 53 (6) (2016) 1243–1260.

[21] G. Yin, X. Xiao, F. Cirak, Topologically robust cad model generation for structural optimisation, Computer Methods in Applied Mechanics and Engineering 369 (2020) 113102. doi:10.1016/j.cma.2020.113102.

[22] G. Yi, N. H. Kim, Identifying boundaries of topology optimization results using basic parametric features, Structural and Multidisciplinary Optimization 55 (5) (2017) 1641–1654. doi:10.1007/s00158-016-1597-9.

[23] Y. Xia, M. Langelaar, M. A. Hendriks, A critical evaluation of topology optimization results for strut-and-tie modeling of reinforced concrete, Comput.-Aided Civ. Infrastruct. Eng. 35 (8) (2020) 850–869. doi:10.1111/mice.12537.

[24] S. C. Subedi, C. S. Verma, K. Suresh, A Review of Methods for the Geometric Post-Processing of Topology Optimized Models, Journal of Computing and Information Science in Engineering 20 (6), 060801 (06 2020). doi:10.1115/1.4047429.

[25] J. Nocedal, S. Wright, Numerical optimization, Springer Science & Business Media, New York, NY, USA, 2006.

[26] M. C. Delfour, J.-P. Zolâsio, Shapes and geometries: metrics, analysis, differential calculus, and optimization, Vol. 22, Siam, 2011.

[27] T. Lee, R. Kashyap, C. Chu, Building skeleton models via 3-d medial surface axis thinning algorithms, CVGIP: Graphical Models and Image Processing 56 (6) (1994) 462–478. doi:10.1006/cgip.1994.1042.

[28] S. Geman, D. E. McClure, Statistical methods for tomographic image reconstruction, Bull. Int. Stat. Inst LII-4 (1987) 5–21.

[29] C. Dapogny, A. Faure, G. Michailidis, G. Allaire, A. Couvelas, R. Estevez, Geometric constraints for shape and topology optimization in architectural design, Computational Mechanics 59 (6) (2017) 933–965. doi:10.1007/s00466-017-1383-6.

[30] T. M. Boake, Architecturally Exposed Structural Steel, Birkhäuser Verlag GmbH, Basel, Switzerland, 2015.

[31] G. Allaire, C. Dapogny, G. Delgado, G. Michailidis, Multi-phase structural optimization via a level set method, ESAIM: control, optimisation and calculus of variations 20 (2) (2014) 576–611.

[32] S. Chen, G. Charpiat, R. J. Radke, Converting level set gradients to shape gradients, in: K. Daniilidis, P. Maragos, N. Paragios (Eds.), Computer Vision – ECCV 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 715–728.

[33] GE jet engine bracket challenge, https://grabcad.com/challenges/ge-jet-engine-bracket-challenge, accessed: 2022-10-11.

[34] R. Picelli, S. Townsend, C. Brampton, J. Norato, H. Kim, Stress-based shape and topology optimization with the level set method, Computer Methods in Applied Mechanics and Engineering 329 (2018) 1–23. doi:10.1016/j.cma.2017.09.001.