Topologically robust CAD model generation for structural optimisation

Ge Yin, Xiao Xiao, Fehmi Cirak*

Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, UK

Abstract

Computer-aided design (CAD) models play a crucial role in the design, manufacturing and maintenance of products. Therefore, the mesh-based finite element descriptions common in structural optimisation must be first translated into CAD models. Currently, this can at best be performed semi-manually. We propose a fully automated and topologically accurate approach to synthesise a structurally-sound parametric CAD model from topology optimised finite element models. Our solution is to first convert the topology optimised structure into a spatial frame structure and then to regenerate it in a CAD system using standard constructive solid geometry (CSG) operations. The obtained parametric CAD models are compact, that is, have as few as possible geometric parameters, which makes them ideal for editing and further processing within a CAD system. The critical task of converting the topology optimised structure into an optimal spatial frame structure is accomplished in several steps. We first generate from the topology optimised voxel model a one-voxel-wide voxel chain model using a topology-preserving skeletonisation algorithm from digital topology. The weighted undirected graph defined by the voxel chain model yields a spatial frame structure after processing it with standard graph algorithms. Subsequently, we optimise the cross-sections and layout of the frame members to recover its optimality, which may have been compromised during the conversion process. At last, we generate the obtained frame structure in a CAD system by repeatedly combining primitive solids, like cylinders and spheres, using boolean operations. The resulting solid model is a boundary representation (B-Rep) consisting of trimmed non-uniform rational B-spline (NURBS) curves and surfaces.

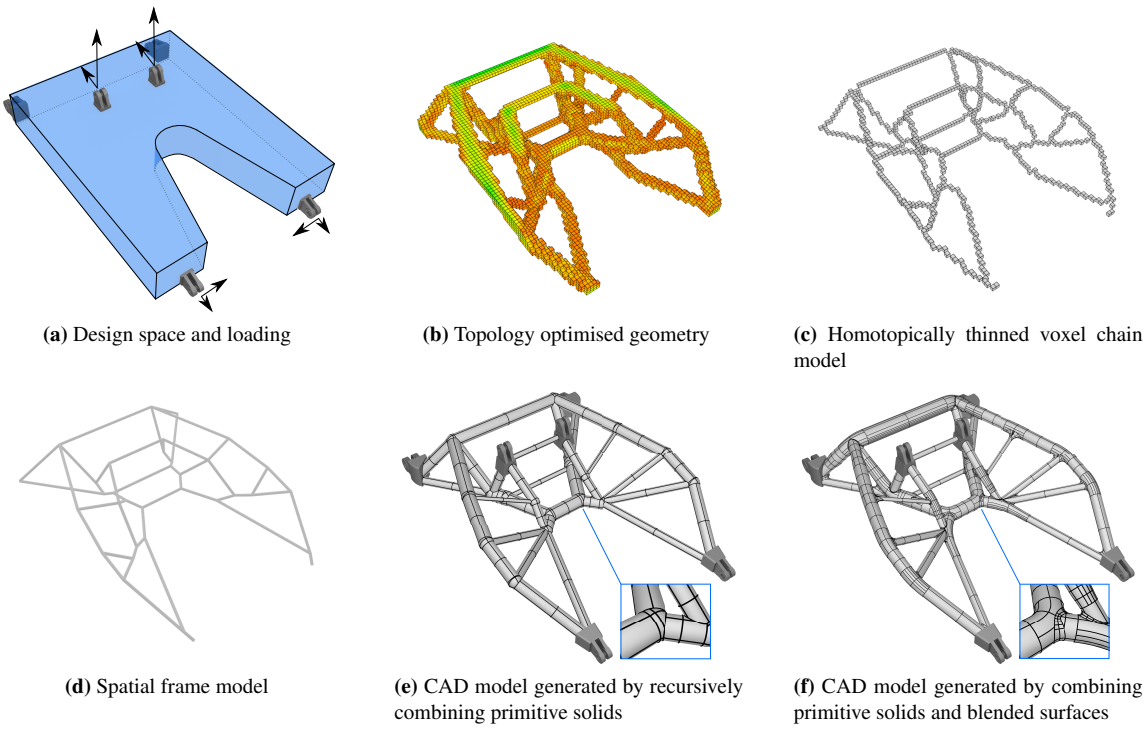
Keywords: topology optimisation, computer-aided design, digital topology, homotopic skeletonisation, CSG tree

1. Introduction

The application of structural optimisation in industrial design requires the optimised geometries to be converted into CAD models. This is necessary because CAD systems are today an integral part of most product development processes. The prevalent parametric CAD systems are based on boundary representation (B-rep) techniques and trimmed NURBS (non-uniform rational B-splines). Geometries are created using constructive solid geometry (CSG) by recursively combining primitive shapes, such as cylinders, cubes etc. using boolean operations. This recursive process is stored in the form of a CSG tree. Although the input to most structural optimisation is a CAD geometry, its output is usually a finite element mesh of the optimised geometry. The finite element meshes consist of too many elements to be processed and edited in a CAD system. For a geometry to be editable in a CAD system, a compact representation in the form of a CSG tree is needed. The need to edit the optimised part geometry arises when additional geometric features have to be added or when the part is to be combined with other components into a functional product. Moreover, in industrial practice there are usually, in addition to structural efficiency and robustness, many equally important, often not explicitly quantified, design requirements, which require the geometry to be edited after optimisation.

The fully automated process we propose can robustly generate a compact CSG tree representation of topology optimised geometries and convert them to CAD models. The entire workflow is illustrated with the help of the

*Corresponding author
Email address: f.cirak@eng.cam.ac.uk (Fehmi Cirak)



(a) Design space and loading

(b) Topology optimised geometry

(c) Homotopically thinned voxel chain model

(d) Spatial frame model

(e) CAD model generated by recursively combining primitive solids

(f) CAD model generated by combining primitive solids and blended surfaces

Figure 1: CAD model generation workflow for a topology optimised rocker arm. The rocker arm is a lever mechanism which pivots around the two of the joints shown in (a) when a force is applied at the other four joints. In (e) and (f) the six detailed joint designs, with each containing a roller bearing, have been added in a CAD system after model generation.

topology optimisation of the rocker arm shown in Figure 1. For topology optimisation we use a standard density based SIMP (solid isotropic material with penalisation) approach on a structured hexahedral finite element mesh to be referred to as a voxel mesh, see e.g. [1–3]. The optimised density field splits the voxels after thresholding into the subsets solid and void, which gives a three-dimensional binary image with two grey levels. Subsequently, we extract from the binary image a voxel chain skeleton using a homotopic, i.e. topology-preserving, skeletonisation algorithm from digital topology [4]. The skeleton has the same number of connected components, handles and cavities as the three-dimensional voxel model. Evidently, topology preservation is critical in structural applications because a change of topology may disrupt the load paths discovered during optimisation. The voxel chain model defines a weighted undirected graph which is processed with graph algorithms to obtain a spatial frame structure. Each of the members of the frame structure is a beam and the beams are rigidly connected at the joints. During the conversion of the voxel model to a spatial frame structure the optimality of the structure is usually compromised. This can, however, be recovered after applying a few steps of size and layout optimisation to the spatial frame structure. Size optimisation adjusts the cross-sections of the members and layout optimisation adjusts the coordinates of the joints. The very compact representation which we generate from the original voxel model consists of the connectivity of the frame structure, its joint coordinates and the member cross-sections. These are used to create a binary CSG tree and a solid model of the spatial frame structure in a CAD system. Figures 1e and 1f show two different solid models generated in a CAD system. In the first model the cylindrical members are connected to spherical joints. In the second model the members are still cylinders, which are, however, smoothly blended at the joints. With the process presented in this paper, the first model is generated in a fully automated fashion while the second requires some user intervention. Note although we used a density based approach for topology optimisation, it is straightforward to apply the proposed approach to the more recent level-set based methods [5, 6] or the historical homogenisation method [7], see also [8] for an overview on optimisation approaches.

There are only a very limited number of structural optimisation approaches which arrive at a geometry in the form of a compact parametric CAD model. In more recent topology optimisation formulations the placement of a finite

number of geometric primitives, such as rectangles or cuboids, embedded in a larger design domain is considered [9–12]. The primitives are allowed to freely move within the design domain usually discretised with a fixed-grid finite element approach. The ensuing optimisation problem is inherently discrete and is converted into a continuous problem by careful regularisation. Although these techniques can provide a compact CAD model they are usually incompatible with the commercially prevalent density based topology optimisation implementations and have so far been applied mostly to 2D problems. There are also other approaches primarily intended for shape optimisation. For instance, it is quite common to use the parameters of a CAD or a reconstructed CAD-like spline model as geometric design variables in shape optimisation, see e.g. [13–16]. Such techniques are especially appealing in an isogeometric analysis context when the same basis functions are used for geometry description and finite element analysis [17–22]. The compact spline representation of the optimised geometry can in principle be imported into a CAD system. In the past, there has been some work in topology optimisation on generating a spline representation based on a given optimised finite element geometry. A common approach is to fit the density isocontours obtained with topology optimisation with a spline curve in two dimensions (2D) or a spline surface in three dimensions (3D) [23–25]. Especially in 3D, these techniques are not robust as they may require the solution of a nonlinear least squares problem and are hard to automate due to the need to manually position the control vertices of the splines. Possibly, therefore, they had no noteworthy impact on commercial software despite being developed around two decades ago. The use of skeletonisation, or thinning, for compact geometric representations has earlier been pioneered in Bremicker et al. [26] for 2D and very recently in Cuillière [27] for 3D. While in [26] the aim was not to arrive at a parametric CAD model, skeletonisation algorithms from digital topology were used to extract a truss structure, which was subsequently size and layout optimised. Homotopic skeletonisation in 3D is substantially more challenging than in 2D, and elimination of possible mechanisms in a spatial truss structure is far from trivial [28]. And so, in our approach, we convert the voxel chain model into a spatial frame structure (with rigidly connected joints) which intrinsically does not exhibit mechanisms. In [27] a curve skeleton is extracted from the surface mesh representing the density isocontour of the optimised geometry using the skeleton extraction method proposed by Au et al. [29]. This specific extraction technique is rather elaborate as it builds on the successive contraction of a given surface mesh using Laplacian smoothing. The obtained skeleton and estimated cross-sections are used to generate a CAD model with no further postprocessing. It is not taken into account that skeletonisation may have impaired the optimality of the structure. A more manufacturing oriented overview of compact parametric representations for topology optimised geometries can be found in the recent review [30].

An essential component of the proposed CAD model generation process is skeletonisation, which is an active research area with applications in computer graphics, animation and volumetric image processing, see the comprehensive reviews [31–35]. The curve-skeleton, briefly the skeleton, of a 3D object is closely related to its medial surface, or surface-skeleton. In finite elements medial surfaces are known from mesh generation applications, see e.g. [36]. Informally, the medial surface is the set of all points that have two or more closest points on a 3D object's surface. That is, a sphere centred on the medial surface touches the object's surface at two or more points. In contrast to the medial surface, the skeleton of a 3D object has no rigorous definition. It is expected that the skeleton captures the essential topology of the object and is centred, i.e. lies on or close to the medial surface. There are many algorithms for determining the skeleton of an object starting from different types of geometry representations, such as implicit signed distances, polygonal surfaces etc. In topology optimisation, it is expedient to assume that the object is given as a voxelised binary image consisting of solid and void voxels. Digital topology provides a rigorous basis to study the topological properties of binary images consisting of pixels or voxels. An accessible introduction to digital topology can be found in the review [37] and the textbook [38]. Homotopic skeletonisation algorithms based on digital topology can extract a one-voxel-wide skeleton with the same topology as the binary image, i.e. the same number of connected components, handles and cavities. See the excellent reviews [37, 39] for early papers and [33, 35] for more recent papers on skeletonisation. Most of these methods start from the domain boundaries and proceed by iteratively removing voxels one by one that do not alter the topology of the object. When the algorithms terminate, the entire structure is represented by a network of one-voxel-wide chains. It is sufficient to inspect only a voxel's immediate neighbourhood consisting of 3×3×3 voxels to decide whether that voxel can be removed. The possible states of such a small cluster can effectively be encoded in a look-up table [4, 40]. Crucially, the skeletonisation process does not rely on any floating-point operations, which makes it exceedingly robust. Recent research on skeletonisation focuses on parallelisation [41, 42] and improving the memory usage [43] of the algorithms. Skeletonisation is however far less computing intensive than finite element analysis so that advanced skeletonisation algorithms are of limited relevance

for the present work. Our approach is based on Lee et al. [4] which is one of the first provably homotopy preserving skeletonisation algorithms. Its implementation is also discussed in the more recent papers [44, 45].

The outline of this paper is as follows. In Section 2, the standard density based topology optimisation and the size and layout optimisation of spatial frames are briefly reviewed. Subsequently, in Section 3 relevant aspects of digital topology are introduced and the implemented specific skeletonisation algorithm is described. The robustness and runtime of our implementation are studied with a relatively complex quadcopter frame geometry. The postprocessing of the obtained voxel chain model with graph algorithms and the generation of first a frame structure and then a CAD model are discussed in Section 4. The entire optimisation and geometry conversion process is summarised in Section 5. The application of the proposed approach to a standard 2D benchmark example from topology optimisation and more complex 3D examples is illustrated in Section 6. We study in particular the change of the compliance cost function during the conversion of the optimised voxel model to a CAD model.

2. Review of structural optimisation

In this Section, we briefly review the standard density based topology optimisation using the SIMP method and the size and layout optimisation of spatial frame structures. In all cases, the cost function is the compliance, and the discussion is restricted to aspects relevant to this paper. For further details see, e.g., the monographs [3, 46].

2.1. Topology optimisation of solids

The topology optimisation problem for finite element discretised solids reads

$$\begin{aligned} &\text{minimise} && J(\boldsymbol{\rho}) = \boldsymbol{f} \cdot \boldsymbol{u}(\boldsymbol{\rho}) && \text{(1a)}\\ &\text{subject to} && \boldsymbol{K}(\boldsymbol{\rho})\boldsymbol{u} = \boldsymbol{f} && \text{(1b)}\\ &&& \frac{V(\boldsymbol{\rho})}{\overline{V}} \leq V_f && \text{(1c)}\\ &&& \mathbf{0} \leq \boldsymbol{\rho} \leq \mathbf{1}\,, && \text{(1d)} \end{aligned}$$

where $J(\boldsymbol{\rho})$ is the compliance cost function, $\boldsymbol{\rho}$ is the vector of relative element densities, $\boldsymbol{u}$ is the displacement vector, $\boldsymbol{K}$ is the global stiffness matrix, $\boldsymbol{f}$ is the global external force vector, $V(\boldsymbol{\rho})$ is the material volume, $\overline{V}$ is the design domain volume and V_f is the scalar prescribed volume fraction. The relative density of each element is constrained to be $0 \leq \rho_i \leq 1$ with $i \in \{1, \ldots, n_e\}$ and the number of elements is n_e.

As usual, the global stiffness matrix $\boldsymbol{K}$ and vector $\boldsymbol{f}$ are assembled from the n_e element contributions $\boldsymbol{K}_i$ and $\boldsymbol{f}_i$ in the mesh. In this paper, we discretise the design domain always with a structured grid and hexahedral linear elements. In each element i the material is isotropic and homogeneous, and the Young's modulus E is penalised depending on the relative density ρ_i with

$$E(\rho_i) = E_{\min} + \rho_i^p(\overline{E} - E_{\min})\,, \tag{2}$$

where $\overline{E}$ is the prescribed Young's modulus of the solid material, p and $E_{\min}$ are two algorithmic parameters. The penalisation parameter $p \geq 3$ ensures that elements with densities close to $\rho_i = 0$ (void) and $\rho_i = 1$ (solid) are preferred. The small Young's modulus $E_{\min} \approx 10^{-9}$ of the void material prevents ill-conditioning of the global stiffness matrix when $\rho_i = 0$. Each element stiffness matrix $\boldsymbol{K}_i$ is computed using the corresponding Young's modulus $E(\rho_i)$ with the relative density ρ_i, which is constant within an element.

Furthermore, the topology optimisation problem (1) is regularised by filtering the element densities ρ_i to prevent checker-board instabilities and mesh dependency of the solution. This is accomplished by convolving the element densities with the kernel function

$$H(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} R - \operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) & \text{if } \operatorname{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq R \\ 0 & \text{else} \end{cases}, \tag{3}$$

which depends on the coordinates of the centroids $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ of the elements i and j, and the prescribed filter length R. With $H(\boldsymbol{x}_i, \boldsymbol{x}_j)$ at hand the filtered densities are given by

$$\hat{\rho}_i = \frac{\sum_j H(\boldsymbol{x}_i, \boldsymbol{x}_j)\rho_j}{\sum_j H(\boldsymbol{x}_i, \boldsymbol{x}_j)} \tag{4}$$

assuming that all elements have the same volume.

For gradient-based optimisation the derivatives of the cost and constraint functions in (1) with respect to the relative densities ρ_i are needed. In the following the relative densities ρ_i in the topology optimisation problem (1) are replaced with the filtered relative densities $\hat{\rho}_i$. The derivative, or sensitivity, of the compliance cost function (1a) reads

$$\frac{\partial J(\hat{\boldsymbol{\rho}})}{\partial \rho_i} = \sum_j \boldsymbol{f} \cdot \frac{\partial \boldsymbol{u}(\hat{\boldsymbol{\rho}})}{\partial \hat{\rho}_j} \frac{\partial \hat{\rho}_j}{\partial \rho_i} \,. \tag{5}$$

Differentiating and rearranging the equilibrium constraint (1b) gives

$$\frac{\partial \boldsymbol{u}(\hat{\boldsymbol{\rho}})}{\partial \hat{\rho}_j} = -\boldsymbol{K}^{-1}(\hat{\boldsymbol{\rho}}) \frac{\partial \boldsymbol{K}(\hat{\boldsymbol{\rho}})}{\partial \hat{\rho}_j} \boldsymbol{u}(\hat{\boldsymbol{\rho}}) \tag{6}$$

and the derivative of the filtered densities (4) is

$$\frac{\partial \hat{\rho}_j}{\partial \rho_i} = \frac{H(\boldsymbol{x}_j, \boldsymbol{x}_i)}{\sum_k H(\boldsymbol{x}_j, \boldsymbol{x}_k)} \,. \tag{7}$$

Introducing both derivatives into (5) yields

$$\frac{\partial J(\hat{\boldsymbol{\rho}})}{\partial \rho_i} = -\sum_j \boldsymbol{u}(\hat{\boldsymbol{\rho}}) \cdot \frac{\partial \boldsymbol{K}(\hat{\boldsymbol{\rho}})}{\partial \hat{\rho}_j} \boldsymbol{u}(\hat{\boldsymbol{\rho}}) \frac{H(\boldsymbol{x}_j, \boldsymbol{x}_i)}{\sum_k H(\boldsymbol{x}_j, \boldsymbol{x}_k)} \,, \tag{8}$$

where the stiffness matrix derivative is to be assembled from element contributions considering the penalised Young's modulus (2).

The derivative of the volume constraint (1c) is obtained analogously with

$$\frac{\partial V(\hat{\boldsymbol{\rho}})}{\partial \rho_i} = \sum_j \hat{\rho}_j \frac{V}{n_e} \frac{H(\boldsymbol{x}_j, \boldsymbol{x}_i)}{\sum_k H(\boldsymbol{x}_j, \boldsymbol{x}_k)} \,. \tag{9}$$

It is assumed here that all the elements are of the same size, namely the design volume $\overline{V}$ divided by the total number of elements n_e.

Finally, with the obtained derivative of the compliance cost function (8) and the volume constraint (9) the optimised density distribution is determined iteratively with the optimality criteria method, see e.g. [3].

2.2. Size and layout optimisation of frames

We consider spatial frame structures consisting of straight beam members connected by joints that can transfer forces and moments. The members can deform by stretching, bending and torsion and are modelled as classical Timoshenko beams, see Appendix A. Without loss of generality, size and layout optimisation can be posed as an iterative sequential optimisation problem. In each optimisation step, either the size, i.e. cross-section area, or layout, i.e. joint coordinates, of the members is optimised.

Both the size and layout optimisation problems have the same structure as topology optimisation (1), namely,

$$\begin{aligned}
&\text{minimise} && J(\boldsymbol{s}) = \boldsymbol{u}(\boldsymbol{s}) \cdot \boldsymbol{K}(\boldsymbol{s})\boldsymbol{u}(\boldsymbol{s}) = \boldsymbol{f} \cdot \boldsymbol{u}(\boldsymbol{s}) && \text{(10a)}\\
&\text{subject to} && \boldsymbol{K}(\boldsymbol{s})\boldsymbol{u}(\boldsymbol{s}) = \boldsymbol{f} && \text{(10b)}\\
& && \frac{V(\boldsymbol{s})}{\overline{V}} \leq V_f && \text{(10c)}\\
& && \boldsymbol{s}_l \leq \boldsymbol{s} \leq \boldsymbol{s}_u \,. && \text{(10d)}
\end{aligned}$$

The vector of design variables $\boldsymbol{s}$ refers either to cross-section areas in case of size optimisation or joint coordinates in case of layout optimisation. The two bounds $\boldsymbol{s}_l$ and $\boldsymbol{s}_u$ have different interpretations in the two cases.

The frame consists of n_e beam elements with the element stiffness matrices $\boldsymbol{K}_i$ and vectors $\boldsymbol{f}_i$. Each stiffness matrix $\boldsymbol{K}_i$ is obtained from a local stiffness matrix $\boldsymbol{K}_i^l$ formulated in a coordinate system attached to the element with the index i. The local matrix $\boldsymbol{K}_i^l \in \mathbb{R}^{12\times12}$ with the displacements and rotations of the two end nodes of the beam element as degrees of freedom is given in Appendix A. In the local x_i^l, y_i^l and z_i^l coordinate system the beam axis is assumed to be aligned with the x_i^l axis. The matrix K_i^l is transformed to the global x, y and z coordinate system according to

$$\boldsymbol{K}_i = \boldsymbol{\Lambda}_i \boldsymbol{K}_i^l \boldsymbol{\Lambda}_i^\mathsf{T} \tag{11}$$

with the block-diagonal transformation matrix

$$\boldsymbol{\Lambda}_i = \mathrm{diag}(\boldsymbol{\lambda}_i,\, \boldsymbol{\lambda}_i,\, \boldsymbol{\lambda}_i,\, \boldsymbol{\lambda}_i)\,. \tag{12}$$

The rotation matrix $\boldsymbol{\lambda}_i \in \mathbb{R}^{3\times3}$ can be chosen with

$$\boldsymbol{\lambda}_i = \begin{pmatrix} \cos\alpha_i \cos\beta_i & -\sin\alpha_i & \cos\alpha_i \sin\beta_i \\ \sin\alpha_i \cos\beta_i & \cos\alpha_i & \sin\alpha_i \sin\beta_i \\ -\sin\beta_i & 0 & \cos\beta_i \end{pmatrix}, \tag{13}$$

which is composed of the two elemental rotations α_i around the z_i^l axis and β_i around y_i^l axis.

Similar as in topology optimisation, the derivative, or sensitivity, of the compliance cost function (10a) reads

$$\frac{\partial J(\boldsymbol{s})}{\partial s_i} = -\boldsymbol{u}(\boldsymbol{s}) \cdot \frac{\partial \boldsymbol{K}(\boldsymbol{s})}{\partial s_i} \boldsymbol{u}(\boldsymbol{s}) = -\sum_{j=1}^{n_e} \boldsymbol{u}_j(\boldsymbol{s}) \cdot \frac{\partial \boldsymbol{K}_j(\boldsymbol{s})}{\partial s_i} \boldsymbol{u}_j(\boldsymbol{s})\,, \tag{14}$$

where $\boldsymbol{u}_j$ is the vector of nodal displacements and rotations of the element j. In size optimisation the derivative of the element stiffness matrix (11) is given by

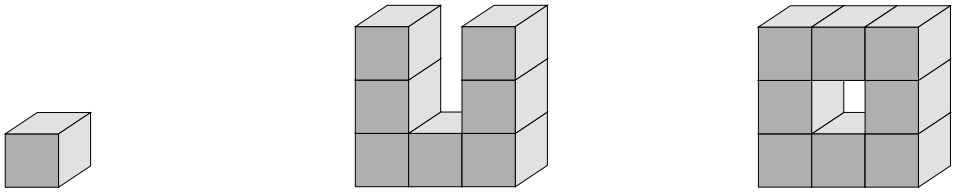
$$\frac{\partial \boldsymbol{K}_i(\boldsymbol{s})}{\partial s_j} = \boldsymbol{\Lambda}_i \frac{\partial \boldsymbol{K}_i^l}{\partial s_j} \boldsymbol{\Lambda}_i^\mathsf{T}\,, \tag{15}$$

whereas in layout optimisation it is

$$\frac{\partial \boldsymbol{K}_i(\boldsymbol{s})}{\partial s_j} = \frac{\partial \boldsymbol{\Lambda}_i}{\partial s_j} \boldsymbol{K}_i^l \boldsymbol{\Lambda}_i^\mathsf{T} + \boldsymbol{\Lambda}_i \frac{\partial \boldsymbol{K}_i^l}{\partial s_j} \boldsymbol{\Lambda}_i^\mathsf{T} + \boldsymbol{\Lambda}_i \boldsymbol{K}_i^l \frac{\partial \boldsymbol{\Lambda}_i^\mathsf{T}}{\partial s_j}\,. \tag{16}$$

3. Digital topology and skeletonisation

The input to skeletonisation is a structured hexahedral finite element grid with all the vertices within the domain having eight incident hexahedra. For digital topology only the connectivity of the mesh is relevant whereas the coordinates of the grid nodes can be arbitrary so that in the following hexahedron and voxel are used interchangeably. The hexahedral grid encodes the binary voxel model of the topology optimised structure. The binary voxel model is obtained by denoting voxel with a density ρ_i above the threshold η as *solid* and the remaining ones as *empty (void)*. Hence, the set of all voxels $v \in \mathcal{V} = \mathcal{V}_s \cup \mathcal{V}_e$ is composed of the solid voxels $\mathcal{V}_s$ and empty voxels $\mathcal{V}_e$. Skeletonisation aims to reassign voxels one by one from $\mathcal{V}_s$ to $\mathcal{V}_e$ until the solid is represented by a network of one-voxel-wide chains. The decision about which voxels to reassign is guided by digital topology and proceeds starting from the voxels at the border between $\mathcal{V}_s$ and $\mathcal{V}_e$.



(a) $\chi(\mathcal{M}_s) = 1;\ \chi(\partial\mathcal{M}_s) = 2$ **(b)** $\chi(\mathcal{M}_s) = 1;\ \chi(\partial\mathcal{M}_s) = 2$ **(c)** $\chi(\mathcal{M}_s) = 0;\ \chi(\partial\mathcal{M}_s) = 0$

Figure 2: Euler characteristics $\chi(\mathcal{M}_s)$ and $\chi(\partial\mathcal{M}_s)$ for three different meshes. The number of different entities for the volume mesh in (a) are $n_0 = 8$, $n_1 = 12$, $n_2 = 6$ and $n_3 = 1$; for the mesh in (b) they are $n_0 = 32$, $n_1 = 60$, $n_2 = 36$ and $n_3 = 7$; and, for the mesh in (c) they are $n_0 = 32$, $n_1 = 64$, $n_2 = 40$ and $n_3 = 8$.

3.1. Review of digital topology

To retain the topology of the voxel model, its Euler characteristic is considered. The solid voxels in $\mathcal{V}_s$ and empty voxels in $\mathcal{V}_e$ yield two hexahedral volume meshes, or more abstractly cell complexes, which will be in the following denoted with $\mathcal{M}_s = \mathcal{M}(\mathcal{V}_s)$ and $\mathcal{M}_e = \mathcal{M}(\mathcal{V}_e)$. The volume mesh of a voxel set consists of the corresponding set of the vertices, edges, faces and voxels. It is also useful to define quadrilateral surface meshes $\partial\mathcal{M}_s$ and $\partial\mathcal{M}_e$ formed by the boundaries of $\mathcal{M}_s$ and $\mathcal{M}_e$, respectively. The Euler characteristic of the voxel model can be determined either from the volume mesh $\mathcal{M}_s$ with

$$\chi(\mathcal{M}_s) = n_0(\mathcal{M}_s) - n_1(\mathcal{M}_s) + n_2(\mathcal{M}_s) - n_3(\mathcal{M}_s)\,, \tag{17}$$

or from the surface mesh $\partial\mathcal{M}_s$ with

$$\chi(\partial\mathcal{M}_s) = n_0(\partial\mathcal{M}_s) - n_1(\partial\mathcal{M}_s) + n_2(\partial\mathcal{M}_s)\,, \tag{18}$$

where n_0 denotes the number of vertices, n_1 the number of edges, n_2 the number of the faces and n_3 the number of hexahedrons in $\mathcal{M}_s$ or $\partial\mathcal{M}_s$. In Figure 2 the Euler characteristics of three voxel meshes are given. Notice that there is also the relation

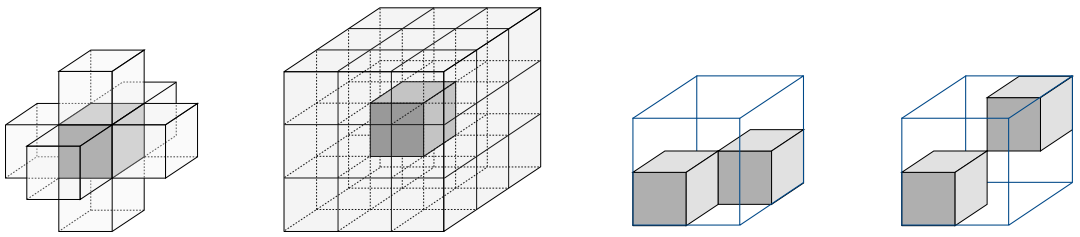
$$\chi(\partial\mathcal{M}_s) = 2\chi(\mathcal{M}_s) \tag{19}$$

between the Euler characteristics of the surface and volume meshes. The Euler characteristic of a voxel model is related to the total number of separate objects (connected components) $o(\mathcal{M}_s)$, handles (tunnels) $h(\mathcal{M}_s)$ and cavities $c(\mathcal{M}_s)$ in the entire model according to the formula

$$\chi(\mathcal{M}_s) = o(\mathcal{M}_s) - h(\mathcal{M}_s) + c(\mathcal{M}_s)\,, \tag{20}$$

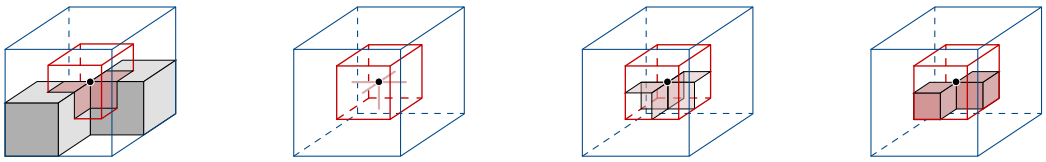
see, e.g., textbooks [47, 48] on topology. For instance, the Euler characteristic $\chi(\mathcal{M}_s)$ of a sphere is 1, of a torus is 0, of a double torus is -1 and so on, compare also Figure 2. Hence, the total number of different entities in the mesh and the described shape are intrinsically connected.

In determining the Euler characteristic of a voxel model, two different types of neighbourship definitions are possible, see Figure 3. These definitions define how pairs of touching voxels such as shown in Figure 3c and 3d are treated, whether connected and forming one surface or disconnected and forming two surfaces. That is, the number of entities in (17) and (18) depends on the chosen definition of the neighbourship. For a voxel v its 6-neighbours consist of all the cells which share a face with v and its 26-neighbours consist of all the cells which share a common vertex with v. The voxels in the 6-neighbourhood of v are denoted with $\mathcal{N}_6(v)$ and the ones in the 26-neighbourhood with $\mathcal{N}_{26}(v)$. In our implementation, two solid voxels are considered to be neighbours when they are 26-neighbours. In contrast, two void voxels are considered to be neighbours when they are 6-neighbours. As pointed out in Kong et al. [37], it is essential not to use the same neighbourship definitions for both the solid and void voxels. The use of one single neighbourhood definition leads to ambiguities, for instance, in the planar case to the violation of the Jordan curve theorem. The Jordan curve theorem states that a simple closed curve divides the plane into an interior and an exterior region.



(a) 6-neighbours ($\mathcal{N}_6(v)$) **(b)** 26-neighbours ($\mathcal{N}_{26}(v)$) **(c)** Not 6-neighbours, but 26-neighbours **(d)** Not 6-neighbours, but 26-neighbours

Figure 3: Voxel neighbourship definitions. In (a) and (b) the light shaded voxels are the $\mathcal{N}_6(v)$ and $\mathcal{N}_{26}(v)$ neighbours of the dark shaded voxel v at the centre. The two voxels in (c) and (d) are each others 26-neighbour but not 6-neighbour.



(a) $n_0^{(q)} = 1$ **(b)** $\frac{n_1^{(q)}}{2} = \frac{5}{2}$ **(c)** $\frac{n_2^{(q)}}{4} = \frac{6}{4}$ **(d)** $\frac{n_3^{(q)}}{8} = \frac{2}{8}$

Figure 4: Contribution $\chi(O(q)) = 1 - \frac{5}{2} + \frac{6}{4} - \frac{2}{8} = -\frac{1}{4}$ of the octant $O(q)$ to the Euler characteristic. Example of an octant with two voxels.

The Euler characteristic (17) and (18) for the entire voxel model can be determined by looping over all vertices in the grid and only considering in each step the voxels attached to one vertex. The 2×2×2 voxels attached to a vertex q are defined as its octant $O(q)$.[1] There are as many octants as vertices in the volume grid (assuming that the voxel domain is padded with empty ghost voxels). The octants are overlapping and the edges, faces and voxels in the hexahedral mesh $\mathcal{M}_s$ appear in several octants. Hence, in computing the Euler characteristic (17) by iterating over the vertices the contribution of each octant has to be suitably weighted such that

$$\chi(\mathcal{M}_s) = \sum_q \chi(O(q)) = \sum_q \left(1 - \frac{n_1^{(q)}}{2} + \frac{n_2^{(q)}}{4} - \frac{n_3^{(q)}}{8}\right), \tag{21}$$

where $n_1^{(q)}$ is the number of edges and $n_2^{(q)}$ is the number of faces neighbouring to the $n_3^{(q)}$ solid voxels in the mesh $\mathcal{M}(O(q)) \setminus \partial\mathcal{M}(O(q))$, see Figure 4. The contribution of an octant $\chi(O(q))$ with different solid-empty voxel states can be precomputed and stored in a look-up table. The eight voxels in an octant have $2^8 = 256$ possible solid-empty states and considering symmetries this reduces to only 22 distinct cases. Tables with contributions of octants to the Euler characteristic can be found, e.g., in [4, 44, 45]

3.2. Skeletonisation by successive voxel removal

The Euler characteristic $\chi(\mathcal{M}_s)$ is critical in determining the voxels at the boundary of the solid mesh $\mathcal{M}_s$ that can be deleted without changing the topology of the voxel model. A *solid border voxel* is defined as a solid voxel with at least one void voxel amongst its 6-neighbours. There is a large amount of research on voxels, also referred to as *simple points*, which can be removed without changing topology [37]. As evident from (20) simply conserving the Euler characteristic $\chi(\mathcal{M}_s)$ of the entire or a portion of the mesh does not ensure that the topology is not changed, i.e. the number of objects $o(\mathcal{M}_s)$, handles $h(\mathcal{M}_s)$) and cavities $c(\mathcal{M}_s)$ all remain the same. For a solid border voxel v to be classified as simple its removal must not change, in addition to the Euler characteristic, the number of objects and handles for both $\mathcal{M}_s$ and $\mathcal{M}_e$ [37]. According to Lee et al. [4] these conditions can be checked by examining the state

[1] Note that it is also possible to associate the octants with the voxels instead of the vertices in the grid. Both viewpoints are equivalent [4].

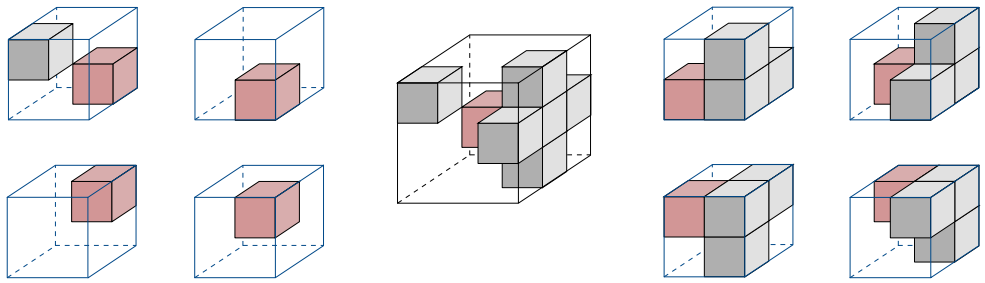


Figure 5: 26-neighbours of a voxel (red) and the eight overlapping octants used for determining the change in the Euler characteristic before and after the voxel is deleted. The contributions of the eight octants with the voxel present are $-\frac{3}{4}$, $\frac{1}{8}$, $-\frac{1}{4}$, $-\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{8}$, $-\frac{1}{4}$ and $-\frac{1}{4}$ with the voxel removed are $\frac{1}{8}$, 0, $-\frac{1}{8}$, $-\frac{1}{8}$, 0, 0, $-\frac{1}{8}$ and $-\frac{1}{8}$ (clockwise), see [4]. Hence, the voxel cannot be removed because this would lead to a change in the Euler characteristic by $\Delta\chi = 1$.

of the voxels in eight octants overlapping voxel v, or the voxels in v's 26-neighbourhood $\mathcal{N}_{26}(v)$. That is, a border voxel v is a simple point if and only if

$$\Delta\chi(\mathcal{M}(\mathcal{N}_{26}(v))) = 0 \quad \text{and} \tag{22a}$$
$$\Delta h(\mathcal{M}(\mathcal{N}_{26}(v))) = 0 \quad \text{or} \tag{22b}$$
$$\Delta o(\mathcal{M}(\mathcal{N}_{26}(v))) = 0\,, \tag{22c}$$

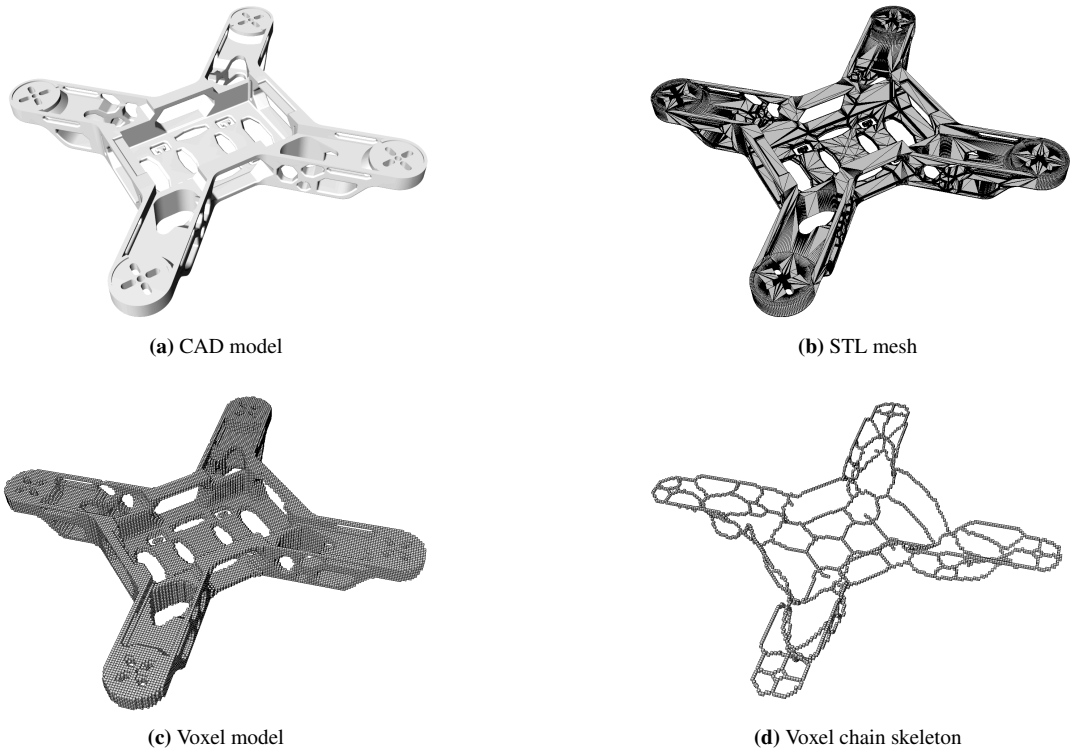
where Δ denotes the change of the respective quantity with and without voxel v present. It is straightforward to determine the Euler characteristic (22a) and it is relatively easy to determine the number of objects (22c) in $\mathcal{N}_{26}(v)$. The change of the Euler characteristic is determined by summing the tabulated contributions of the eight octants belonging to the eight corners of the voxel v from [4], see (21) and Figure 5. To determine the number of objects (22c), we generate an undirected graph with the solid voxels as nodes and introducing edges between voxels that are 26-neighbours. Subsequently, the number of connected components is determined with a depth first search (DFS) algorithm [49].

The skeletonisation proceeds by removing in each step one layer of border voxels which are simple points according to (22), see Algorithm 1 in Appendix B. The removed solid voxels are reclassified as void. One removal step is split into six sub-steps and in each sub-step only the voxels approaching from one of the six grid directions are removed. The geometry of the obtained skeleton depends somewhat on the sequencing of these sub-steps, which is not critical for our purpose. It is straightforward to tag some voxels as non-removable irrespective of (22). In skeletonising topology optimisation results, for instance, the voxels at Dirichlet and non-zero Neumann boundaries are tagged as non-removable. Also, end points with only one voxel in their 26-neighbourhood have to be tagged as non-removable, in order to avoid a complete elimination of voxel chains. The skeletonisation terminates when none of the remaining voxels is removable without violating (22). The final result is a curve skeleton consisting of a network of one-voxel-wide voxel chains connected by joint voxels. We refer to this curve skeleton as the *voxel chain skeleton*.

3.3. Illustrative example and timing

We consider the skeletonisation of the quadcopter frame shown in Figure 6a obtained from GrabCAD. The frame has a non-trivial topology and has been designed in a CAD system. To generate a voxel model the implicit, or level set, representation of the frame on a uniform voxel grid is computed first. This is achieved by determining the distance of each voxel in the grid to the frame surface with the open source openVDB library [50]. In doing so, the frame geometry is approximated with the STL mesh exported from the CAD system depicted in Figure 6b. After thresholding the voxel grid the voxel model of the frame in Figure 6c is obtained. Its skeletonisation with the introduced digital topology algorithm yields the voxel chain skeleton in Figure 6d. It can be visually confirmed that the skeleton appears to have the same topology as the voxel model.

To investigate the efficiency and scaling of our C++ implementation of the introduced skeletonisation algorithm, we consider three different voxel grids for computing the implicit representation and skeletonisation. The size of the three grids, the number of voxels in the different models and the runtimes are given in Table 1. The STL mesh has in all cases 1086791 triangles. The number of solid voxels in the voxel model and the number of removal steps



(a) CAD model

(b) STL mesh

(c) Voxel model

(d) Voxel chain skeleton

Figure 6: Skeletonisation of a quadcopter frame designed in a CAD system. The voxel model in (c) is obtained from the STL mesh in (b) by computing its implicit, or level set, representation on a uniform voxel grid and thresholding. The voxel chain skeleton is determined from the voxel model in (c) with the introduced digital topology algorithm. For further details see under Grid 3 in Table 1.

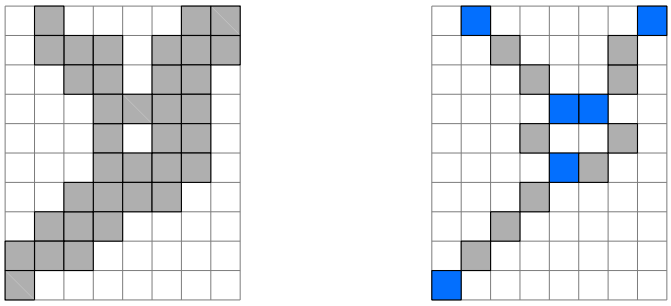
increase with decreasing voxel size because more and more of them cover the frame. The number of skeleton voxels increases because the smaller voxels can capture the topology of the frame better. The reported extremely short runtimes include only skeletonisation (no implicitisation) and confirm the efficiency of the skeletonisation algorithm. Moreover, notice that the average time per removal step is approximately linear with respect to the number of the voxels in the voxel grid. The experiments were performed on a Macbook Pro with an Intel Core CPU i7-4750HQ @ 2.0GHz and 16GB RAM.

4. Model extraction and conversion

The voxel chain skeleton is used to define a structural frame model. The skeleton provides both the connectivity and the geometry of the frame. Although it is feasible to obtain also the member cross-sections from the voxel model, in our implementation cross sections are obtained from size optimisation of the frame, as will be discussed in Section 5. In converting the voxel chain skeleton to a frame we express it as a weighted undirected graph and make use of its incidence matrix. The frame model provides a very compact representation of the optimised structure,

	Voxel grid # voxels	Voxel model # voxels	Skeleton # voxels	removal steps	time per step	total time
Grid 1	$139 \times 16 \times 139$	18222	1278	4	4.43 s	17.73 s
Grid 2	$172 \times 19 \times 172$	38076	1677	5	9.33 s	46.65 s
Grid 3	$197 \times 21 \times 197$	53981	1894	6	13.17 s	78.99 s

Table 1: Efficiency and scaling of the skeletonisation for the quadcopter frame on three different voxel grids.



(a) Voxel model **(b)** Voxel chain skeleton

Figure 7: Two dimensional illustrative example for the skeletonisation of a voxel model. In the voxel chain skeleton the grey voxels with two neighbours denote the regular voxels and the blue voxels with other than two neighbours are the joint voxels.

which can be used to generate a parametric solid model using the scripting interface or API (application programming interface) of a CAD system.

4.1. Graph model

The voxel chain obtained from skeletonisation contains two different types of voxels, see Figure 7. We refer to the voxels with only two voxels in their 26-neighbourhood as *regular voxels*. The remaining voxels are the *joint voxels*, which have other than two voxels in their 26-neighbourhood. As illustrated in Figure 7b, it is not expedient to deduce from every joint voxel a joint for the frame model because this will lead to acutely short members and impractical joint designs.

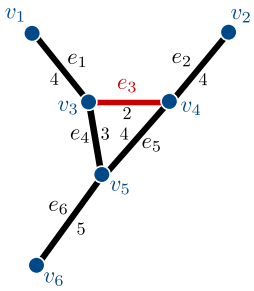
To robustly merge joint voxels in close proximity to a single joint we resort to a weighted undirected graph representation of the voxel chain skeleton. The nodes of the graph are the joint voxels, and the edges are the connections between the joint voxels. Each of the nodes has an associated coordinate vector which is initially equal to the centroid of the corresponding joint voxel. The edge weights are proportional to the number of voxels between the two joints attached to an edge. The graph model is generated with the Algorithm 2 given in Appendix B. In a first step, all the solid voxels in the voxel model are categorised either as regular or joint. Subsequently, starting from the joint voxels we determine the voxel chains and their lengths by marching along the $\mathcal{N}_{26}(v)$ neighbours until a joint voxel is reached. The obtained graph model is stored in the form of an incidence matrix. In Figure 8a and 8d the graph model and its incidence matrix for the voxel chain skeleton in Figure 7b are given. The rows of the incidence matrix correspond to the nodes of the graph and the columns to its edges. Each edge has two identical non-zero entries corresponding to the length of the edge in voxel units.

The joint voxels in close proximity of each other are connected by very short edges. The corresponding graph nodes are merged by successively collapsing the edges of the graph. The edge collapse is implemented with the help of the incidence matrix. Figure 8 illustrates how the incidence matrix evolves during the collapse of the edge e_3 and the merging of its attached nodes v_3 and v_4. The coordinate vector associated to the node v_3 is the average of the node coordinate vectors before merging. After collapsing, the duplicate edge e_5 is detected and merged with e_4. The weight of e_4 is updated to be the closest integer to the mean of the weights of e_4 and e_5.

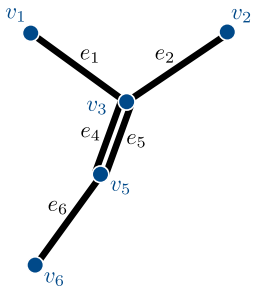
The voxel chain model usually contains branches which lead to structural frame members with zero stress. These are the chains directly connected to a zero Neumann boundary. Their pruning is again implemented with the incidence matrix. As illustrated in Figure 9, after identifying from the incidence matrix the nodes with only one attached edge, nodes and edges are deleted by simply removing the corresponding rows and columns.

4.2. Structural frame and CAD models

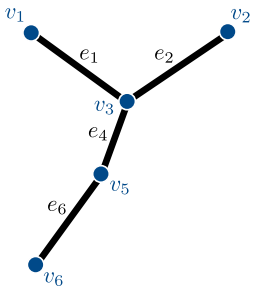
The graph model supplemented by the joint coordinates and member cross-sections provides sufficient information to generate a compact structural frame model. We assume that there is a one-to-one correspondence between nodes and edges of the graph model and joints and members of the frame model. Besides, it is assumed that members are only subjected to end forces and moments, i.e. no distributed loads, so that they can be straight and have uniform



(a) Graph with the edge e_3 to be collapsed



(b) Graph after edge collapse with duplicate edge



(c) Final graph after the removal of duplicate edge

	e_1	e_2	e_3	e_4	e_5	e_6
v_1	4	0	0	0	0	0
v_2	0	4	0	0	0	0
v_3	4	0	2	3	0	0
v_4	0	4	2	0	4	0
v_5	0	0	0	3	4	5
v_6	0	0	0	0	0	5

(d) Incidence matrix

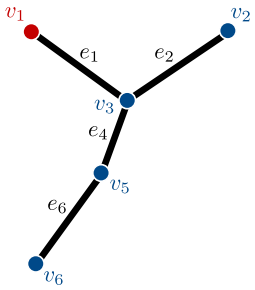
	e_1	e_2	e_3	e_4	e_5	e_6
v_1	4	0	0	0	0	0
v_2	0	4	0	0	0	0
v_3	4	4	4	3	4	0
~~v_4~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
v_5	0	0	0	3	4	5
v_6	0	0	0	0	0	5

(e) Edge collapse

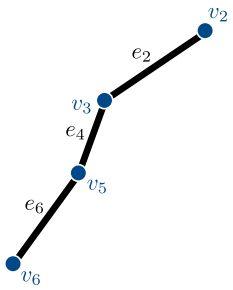
	e_1	e_2	e_4	e_5	e_6
v_1	4	0	0	0	0
v_2	0	4	0	0	0
v_3	4	4	3	4	0
v_5	0	0	3	4	5
v_6	0	0	0	0	5

(f) Duplicate edge removal

Figure 8: Merging of graph nodes connected by short edges using edge collapse. The edge e_3 is collapsed by merging the row v_3 with row v_4 and removing the column e_3 in the incidence matrix. The duplicate edge e_5 after the collapse is merged with e_4 by averaging their weights and deleting the column e_5.



(a) Node v_1 and edge e_1 to be pruned



(b) Graph after pruning

	e_1	e_2	e_4	e_6
~~v_1~~	~~4~~	~~0~~	~~0~~	~~0~~
v_2	0	4	0	0
v_3	4	4	3	0
v_5	0	0	3	5
v_6	0	0	0	5

(c) Incidence matrix

Figure 9: Pruning of branches which lead to a structural frame member with zero stress. The node v_1 and the attached edge e_1 may be deleted if v_1 is not on a Dirichlet or non-homogeneous Neumann boundary. The node and edge are deleted by removing the corresponding row and column of the incidence matrix.

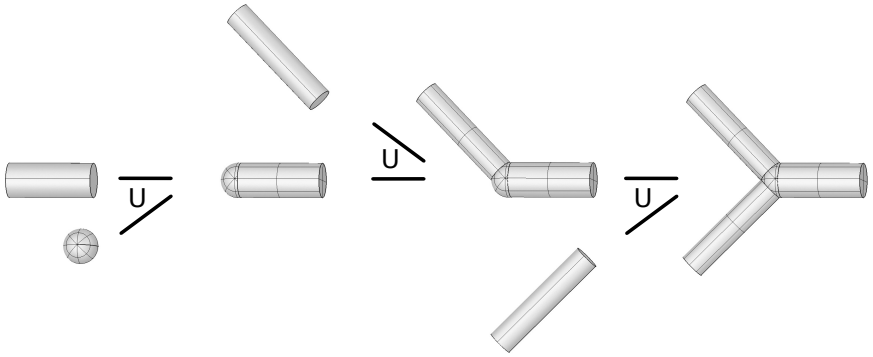


Figure 10: Binary CSG tree of a CAD model generated by successively combining cylinders with spheres using boolean union.

cross-sections along their lengths. Note that these assumptions are fully justified if there is no distributed loading in topology optimisation. Furthermore, for simplicity, we assume in the following that all cross-sections are circular. These restrictions can be mostly relaxed if necessary.

It is straightforward to create a parametric solid CAD model from the structural frame model. To this end, we use either the commercial CAD system Rhinoceros (Rhino) or the open source FreeCAD. However, the proposed approach can be realised in any CAD system which provides a scripting interface or API (application programming interface). In the solid model, the members are represented by cylinders and the joints by spheres, see Figure 10. They are combined with boolean operations. The radius of the sphere at a joint is chosen slightly larger than the largest attached member radius. In this paper, we use a factor of 1.05. A binary CSG tree of the frame model is generated by starting with one of the members and by successively adding spheres and cylinders to the model. Different variations of this approach can be devised and implemented. For instance, it suggests itself to connect two of the most stressed members continuously across a joint or to add fillets to reduce stress concentrations. See also [51, 52] in which a somewhat similar approach was used for generating CAD models for optimised truss structures. After the model is generated it can be edited and postprocessed in the CAD system and be exported, for instance, in STL format for machining or IGES or STEP formats for mesh generation software.

5. Overall optimisation and conversion process

In this Section, we review the sequence of steps from the definition of the topology optimisation problem to obtaining the structurally-sound parametric CAD geometry. We provide additional details for each of the steps and focus on the interplay between the different steps. Each step corresponds to one Section in this paper.

Step 1: Topology optimisation. We assume that the finite element mesh for the optimisation problem (1) is a structured hexahedral grid. In most optimisation problems, the design domain is a parallelepiped which can be discretised with a structured grid. Other more complex design domains can be considered by computing their implicit, or level set, representation and embedding them in a structured hexahedral grid, see the quadcopter frame example in Figure 6. From the outset, the voxels outside the design domain are chosen as void by choosing their Young's moduli with $E_{\min}$.

Step 2: Skeletonisation. The input to the homotopic skeletonisation algorithm is a binary image defined on a hexahedral structured grid. The binary image is obtained by thresholding the topology optimised geometry. The threshold value η is chosen such that the prescribed material volume constraint (1c) in topology optimisation is retained. The output of the skeletonisation is a curve skeleton in the form of a voxel chain with the same topology as the topology optimised geometry.

Step 3: Structural frame model generation. The topology and joint coordinates of the frame model are extracted from the voxel chain model. It is assumed that all the members are straight, have a circular cross-section with the same diameter and their total volume is equal to the prescribed material volume $V_f\overline{V}$ in topology optimisation. It is possible to obtain frame models with more complex member geometries and non-uniform cross-sections. This appears to be, however, usually not desired from an ease and cost of manufacturability viewpoint.

Step 4: Size and layout optimisation. The structural frame model extracted from the skeleton is usually suboptimal. That is, the compliance of the structural frame is larger than the compliance of the topology optimised geometry. To recover the optimality of the frame structure, we apply several steps of sequential size and layout optimisation (10). In size optimisation the diameters d_i of the circular member cross-sections and in layout optimisation the coordinates $\boldsymbol{x}_i$ of the joints are updated. Both optimisation problems are solved with the SQP (sequential quadratic programming) method [53]. Throughout optimisation the volume of the frame is constrained to be equal to the prescribed material volume in topology optimisation $V_f\overline{V}$. It is straightforward to consider additional constraints pertaining to the member cross-sections, the positions of the joints, or positions and orientations of the members. The first step is always size optimisation, which is followed by as many as necessary alternating layout and size optimisation steps until the compliance cost function is converged. If the length of any member reduces to zero during shape optimisation, the member is removed, its end nodes are merged, and the iteration continues.

Step 5: CAD model generation. A compact CAD model of the structural frame can be generated essentially in any parametric CAD system using a fully automated process. The members are represented by cylinders and the joints by spheres which are combined by boolean operations. The underlying binary CSG tree representation makes it easy to edit further and to refine the optimised design.

6. Examples

We consider three examples of increasing complexity to demonstrate the application of the proposed approach. The minimised cost function is the compliance $J(\hat{\boldsymbol{\rho}})$ in topology optimisation and $J(\boldsymbol{d}, \boldsymbol{x})$ in size and layout optimisation, where $\hat{\boldsymbol{\rho}}$ are the filtered element densities, $\boldsymbol{d}$ are the member diameters, and $\boldsymbol{x}$ are the joint coordinates. In all examples, the Young's modulus and Poisson's ratio of the solid material are $\overline{E} = 2.1 \cdot 10^5$ and $\nu = 0.3$. In topology optimisation the penalisation factor is chosen as $p = 3$ and the minimum Young's modulus as $E_{min} = 10^{-9}$. During layout optimisation, a member shorter than $1/20$ of the total length of its immediate neighbouring members is considered as too short, and its two end nodes are merged.

6.1. Cantilever

Cantilevered plates as shown in Figure 11 are one of the most widely studied benchmark examples in topology optimisation, see e.g. [3]. The length, height and width of the chosen design domain are $150 \times 50 \times 4$. The left face of the domain is clamped while all the other faces are free, and a distributed force with a total value of $F = 100$ is applied along the centre of the right face. The finite element discretisation consists of $150 \times 50 \times 4$ linear hexahedral elements. The maximum volume fraction of the optimised structure is prescribed to be $V_f = 0.3$ and the density filter radius is chosen as $R = 3$. Figure 12a depicts the optimised cantilever with only the voxels above a relative density $\eta = 0.3$. The compliance of this voxel model is $J(\hat{\boldsymbol{\rho}}) = 3.40900$, which has been determined after topology optimisation by resetting $p = 1$ in (2).

As a first step in obtaining the structural frame model the homotopic skeletonisation algorithm is employed. The skeletonisation yields after 5×6 voxel removal steps the voxel chain skeleton shown in Figure 12b. As visually apparent, the voxel and voxel chain models have the same topology. The skeleton is converted into the structure depicted in Figure 13a by first identifying the 16 joints and then connecting them with 21 members. As evident from Figure 13a during the conversion to a frame model the optimality of the voxel model is compromised; note, for instance, the non-straight top and bottom members. Optimality is recovered by updating the joint positions, i.e. layout optimisation, and the cross-sectional areas, i.e. size optimisation, see Figure 13b. Initially, all members are assumed to have the same diameter of $d = 4.547$, giving the prescribed total volume of $V = 0.3\overline{V}$. In determining the volume only the member cross-sections and member lengths between the joints are considered.

According to Figure 14b, the frame consisting of beams with uniform diameter has a compliance $J(\boldsymbol{d}, \boldsymbol{x}) = 4.59841$, which is larger than the voxel model. The stretch and bending strain energies of the frame are 2.02202 and 0.247108, respectively. The minimum and maximum axial stresses are 0 and 20.9274. The average axial stress is 8.04859 and its standard deviation is 6.26102. After the first size optimisation step the compliance is reduced to $J(\boldsymbol{d}, \boldsymbol{x}) = 3.59948$ and the subsequent layout optimisation step to $J(\boldsymbol{d}, \boldsymbol{x}) = 2.95264$. Several more steps of size and layout optimisation do not lead to a significant reduction in the compliance. Notice that the obtained final compliance $J(\boldsymbol{d}, \boldsymbol{x}) = 2.90323$

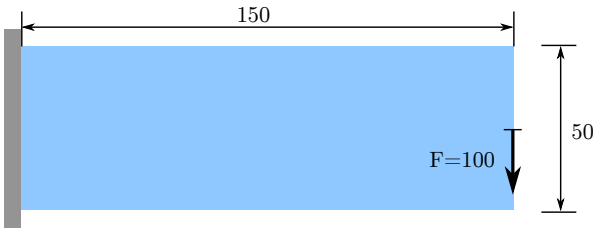
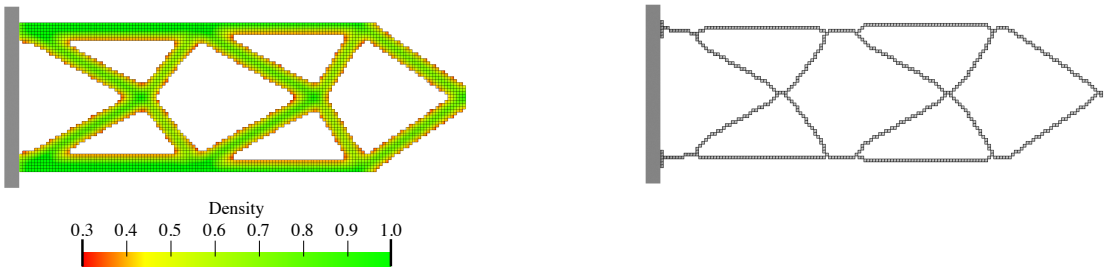
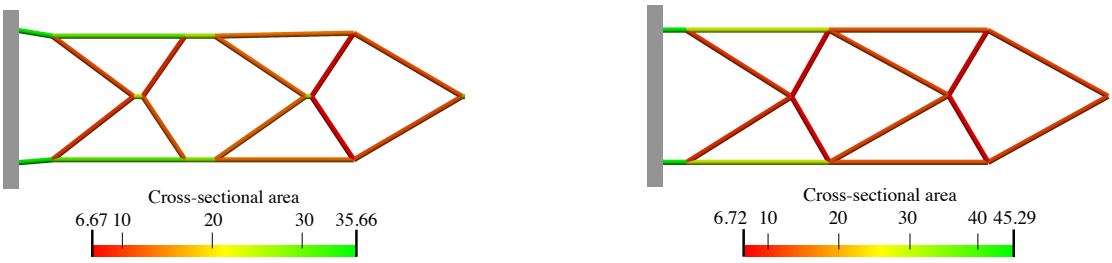


Figure 11: Geometry, boundary conditions and loading of the cantilever.



(a) Thresholded geometry with $\eta = 0.3$ **(b)** Voxel chain skeleton

Figure 12: Topology optimised and skeletonised cantilever.



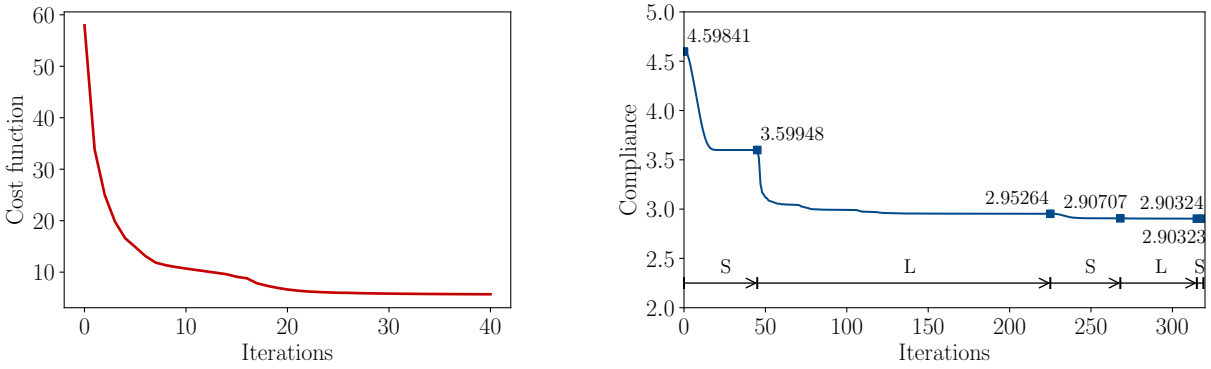
(a) Size optimised (initial) **(b)** Size and layout optimised (final)

Figure 13: Size and layout optimised cantilever frame.

is significantly lower than the compliance $J(\hat{\boldsymbol{\rho}}) = 3.40900$ of the topology optimised voxel model. The stretch and bending strain energies of the final frame are 1.42114 and 0.0190822, respectively. The minimum and maximum axial stresses are 7.00854 and 8.46234. The average axial stress is 8.09638 and its standard deviation is 0.44709. In the final optimised cantilever in Figure 13b the axial stresses become more evenly distributed and the bending stresses are significantly reduced. The final optimised cantilever consists of 11 joints and 16 members. Finally, in Figure 15 the solid CAD model of the frame and a faceted triangular STL mesh exported from FreeCAD are shown. The lines in the CAD model show the layout of the trimmed NURBS patches.

6.2. Pipe bracket

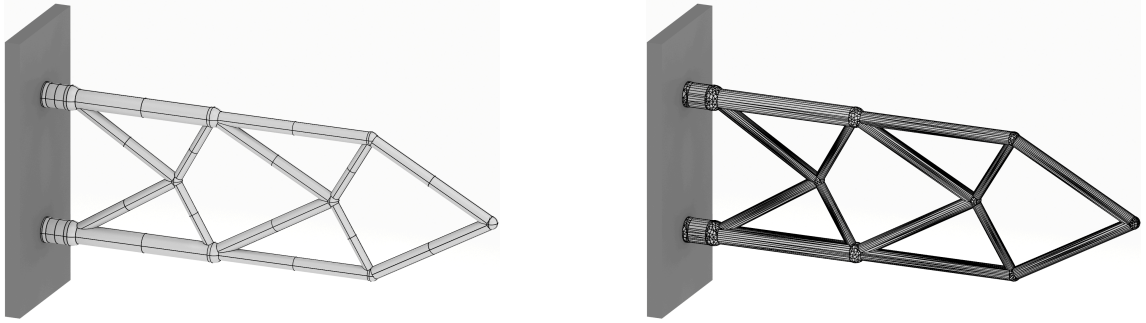
As a structure with a truly three-dimensional load path, the pipe bracket shown in Figure 16 is considered. The design domain has a length, height and width of $120 \times 40 \times 60$ and contains two openings with each of radius of 18 for two pipes passing through the domain. Within the openings each pipe is supported at four points, applying at each support a vertical force of $F = 100$. The four vertical outer edges of the design domain are chosen as fixed. The finite element discretisation consists of $120 \times 40 \times 60$ linear hexahedral elements. To represent the two openings in the finite element model the Young's modulus of the elements within the void region is prescribed with E_{min}. The maximum



(a) Topology optimisation

(b) Size (S) and layout (L) optimisation

Figure 14: Convergence of the compliance during topology and sequential size and layout optimisation of the cantilever frame. Compliance of the topology optimised voxel model is $J(\hat{\boldsymbol{\rho}}) = 3.40900$.



(a) IGES model

(b) STL model

Figure 15: Parametric CAD model of the cantilever.

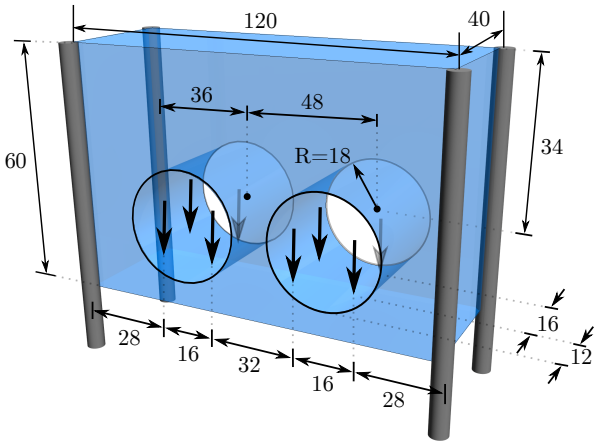
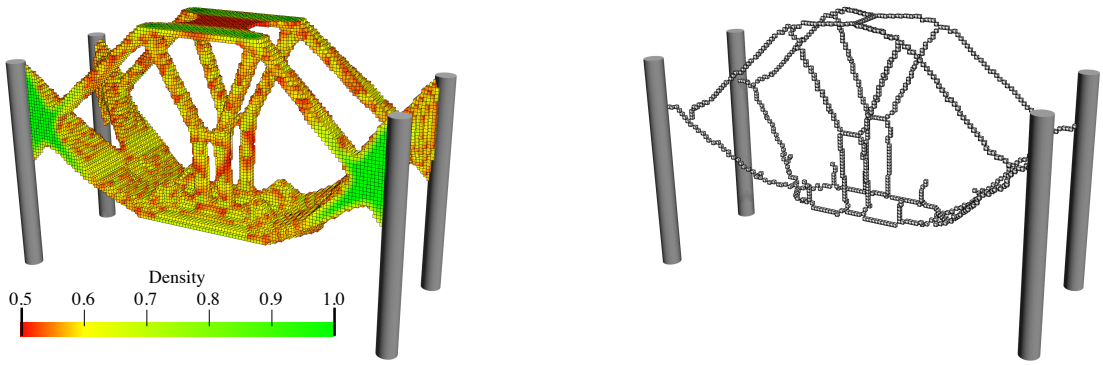


Figure 16: Geometry, boundary conditions and loading of the pipe bracket.



(a) Thresholded geometry with $\eta = 0.5$

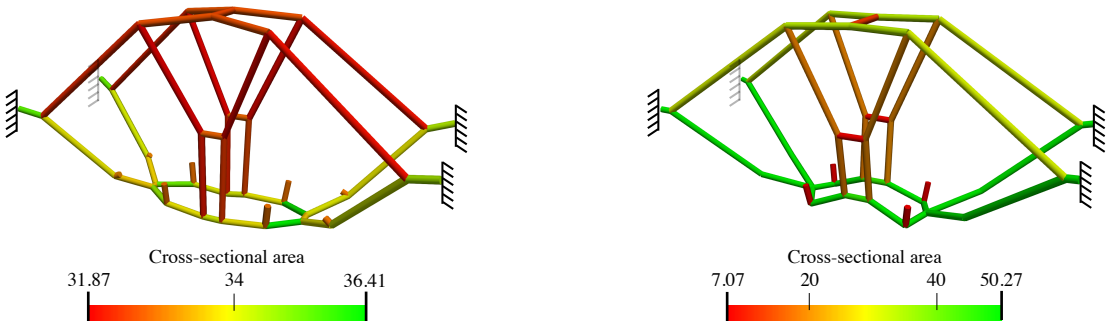
(b) Voxel chain skeleton

Figure 17: Topology optimised and skeletonised pipe bracket.

volume fraction of the optimised structure is prescribed to be $V_f = 0.1$ and the density filter radius is chosen as $R = 3$. Figure 17a depicts the optimised bracket with only the voxels above a relative density $\eta = 0.5$. The compliance of the optimised voxel model is $J(\hat{\boldsymbol{\rho}}) = 2.47602$. As easily recognisable from Figure 17a, the optimised structure is a combination of an arch-like and a cable-like structure with vertical tension members between the two openings.

The skeletonisation algorithm yields after 6×6 voxel removal steps the voxel chain skeleton in Figure 17b. The skeleton is converted into the structure depicted in Figure 18a by first identifying the 42 joints and then connecting them with 50 straight members. Alternatively, the 50 members can be represented with curved Bézier segments, which would lead to a structural frame with curved beam members. This has not been pursued here because such members may lead to an increase in manufacturing costs and are usually not preferred. Initially, all frame members are assumed to have the same diameter $d = 6.296$, giving a total volume of $V = 0.1\overline{V}$.

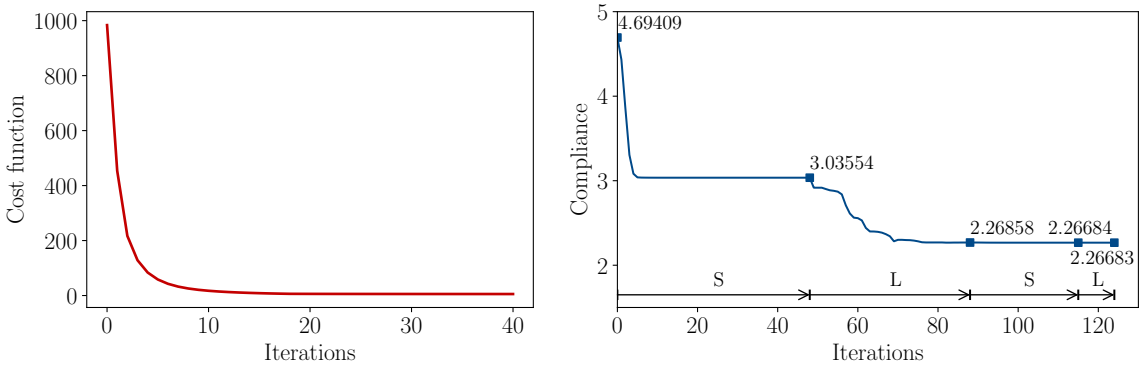
According to Figure 19b, the frame with beams of uniform diameter has a compliance $J(\boldsymbol{d}, \boldsymbol{x}) = 4.69409$, which is significantly higher than $J(\hat{\boldsymbol{\rho}}) = 2.47602$ of the voxel model. After the first size optimisation step the compliance is reduced to $J(\boldsymbol{d}, \boldsymbol{x}) = 3.03554$ and the subsequent layout optimisation step to $J(\boldsymbol{d}, \boldsymbol{x}) = 2.26858$. In layout optimisation the beams are constrained not to move inside the volume occupied by the two cylindrical pipes. The intersection between a beam and a cylinder is determined by integrating the (positive) signed distance of the cylinder along the beam. It is straightforward to differentiate this constraint integral with respect to nodal coordinates of the beam. The intersection constraints for all the beams are aggregated with the Kreisselmeier-Steinhauser (K-S) function [54]. Several more steps of size and layout optimisation do not lead to a significant reduction in the cost function. Notice that the obtained final compliance $J(\boldsymbol{d}, \boldsymbol{x}) = 2.26683$ is lower than the compliance $J(\hat{\boldsymbol{\rho}}) = 2.47602$ of the topology optimised voxel model. The final optimised frame structure depicted in Figure 18b consists of 36 joints and 44 members. In Figure 20 the solid CAD model of the pipe bracket and a faceted triangular STL mesh exported from FreeCAD are shown.



(a) Size optimised (initial)

(b) Size and layout optimised (final)

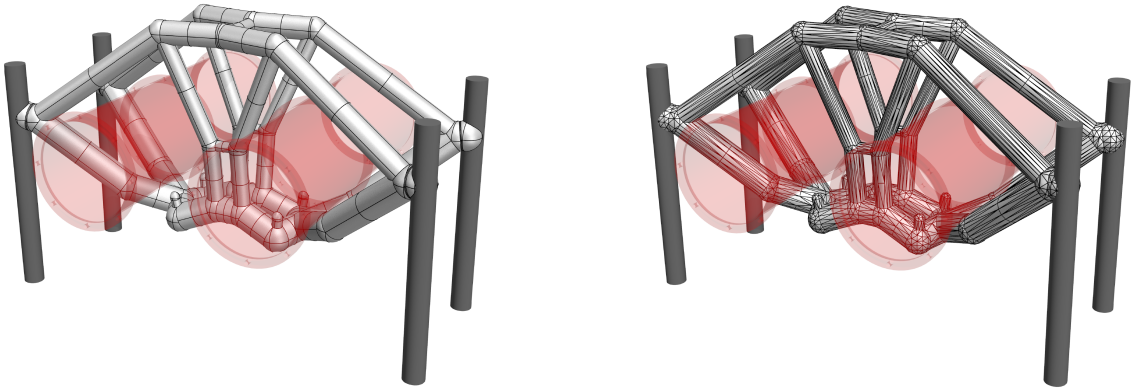
Figure 18: Size and layout optimised pipe bracket.



(a) Topology optimisation

(b) Size (S) and layout (L) optimisation

Figure 19: Convergence of the compliance during topology and sequential size and layout optimisation of the pipe bracket. Compliance of the topology optimised voxel model is $J(\hat{\boldsymbol{\rho}}) = 2.47602$.



(a) IGES model

(b) STL model

Figure 20: Parametric CAD model of the pipe bracket.

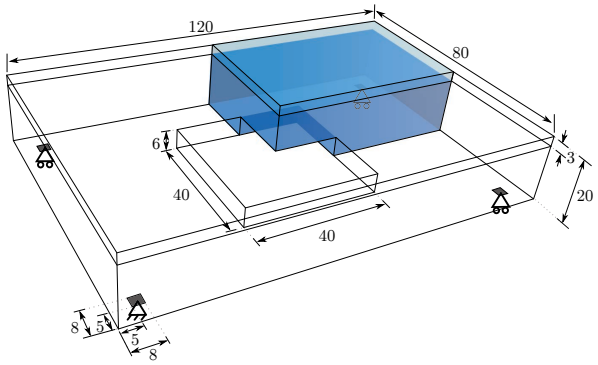
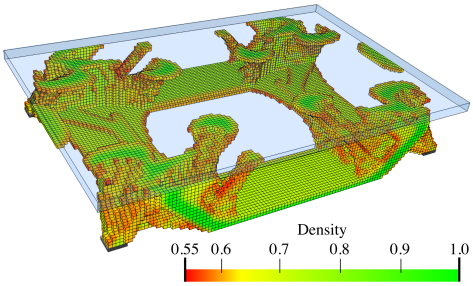
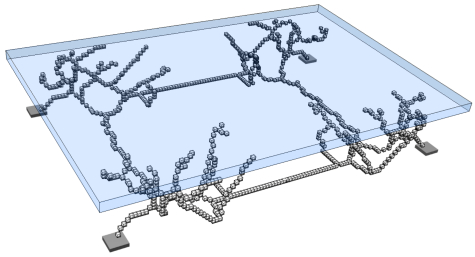


Figure 21: Geometry, boundary conditions and loading of the frame-supported plate. In topology optimisation only one quarter and in size and layout optimisation the entire structure are considered.



(a) Thresholded geometry with $\eta = 0.55$



(b) Voxel chain skeleton

Figure 22: Topology optimised and skeletonised frame-supporeted plate,

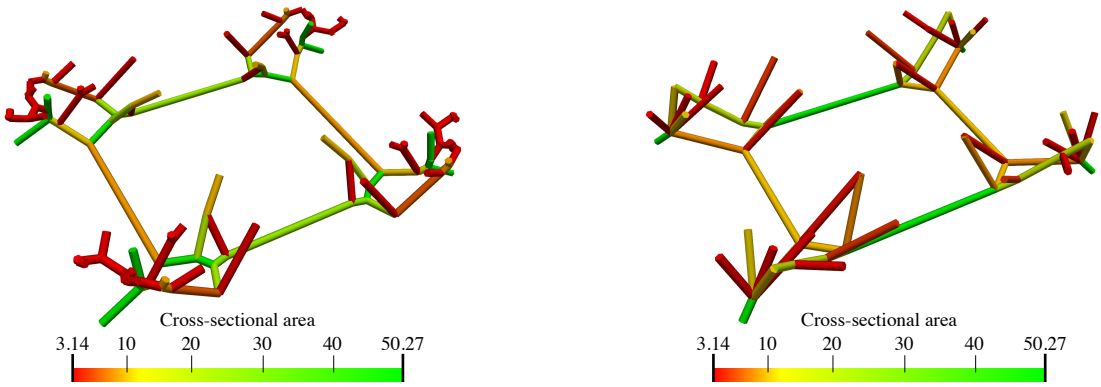
6.3. Frame-supported plate

In structural design it is very common to combine a plate- or shell-like skin structure with a frame-like support structure. In such composite structures the skin can be modelled as a standard plate or shell structure, and topology optimisation is applied only in the remaining part of the design domain. The design domain shown in Figure 21 combines a thin plate-like skin with a solid bottom part to be optimised. The entire domain is of size $120 \times 80 \times 20$ and contains a small opening of size $40 \times 40 \times 6$. The plate is of thickness 3 and is subjected to a uniform pressure load of 1. The four corners of the design domain are vertically supported. Due to symmetry only a quarter of the design domain is considered in topology optimisation. The finite element discretisation consists of $60 \times 40 \times 23$ linear hexahedral finite elements. Across the height of the design domain there are 6 elements with a size of $1 \times 1 \times 0.5$ in the plate-like top part and 17 unit sized elements in the bottom part. To represent the small opening in the finite element model the Young's modulus of the relevant elements is prescribed as E_{min}.

In topology optimisation, the maximum volume fraction is prescribed to be $V_f = 0.25$ and the density filter radius is chosen as $R = 1.5$. The optimised structure with only the voxels above a relative density $\eta = 0.55$ is shown in Figure 22a and its voxel chain skeleton in Figure 22b. Although the voxels in the top part of the design domain are present during optimisation and skeletonisation, they are tagged as non-removable. The voxel chain skeleton is converted to the frame structure shown in Figure 23a. The structural model also contains a top plate not shown in Figure 23. The plate is modelled as a Kirchhoff–Love plate and is discretised with 4×4 quadrilateral elements and Catmull-Clark subdivision basis functions. The frame has 124 joints and 136 members. Initially, all the members are assumed to have the same diameter of $d = 5.010$ which gives a total frame volume of $V = 4800$. The coupling between the plate and the frame is achieved with Lagrange multipliers as described in [55]. The joints between the plate and frame can transfer forces but not moments.

According to Figure 24b, the compliance of the frame-supported plate with members of uniform diameter is $J(\boldsymbol{d}, \boldsymbol{x}) = 1210.01$, which reduces after several sequential size and layout optimisation steps to $J(\boldsymbol{d}, \boldsymbol{x}) = 322.967$.

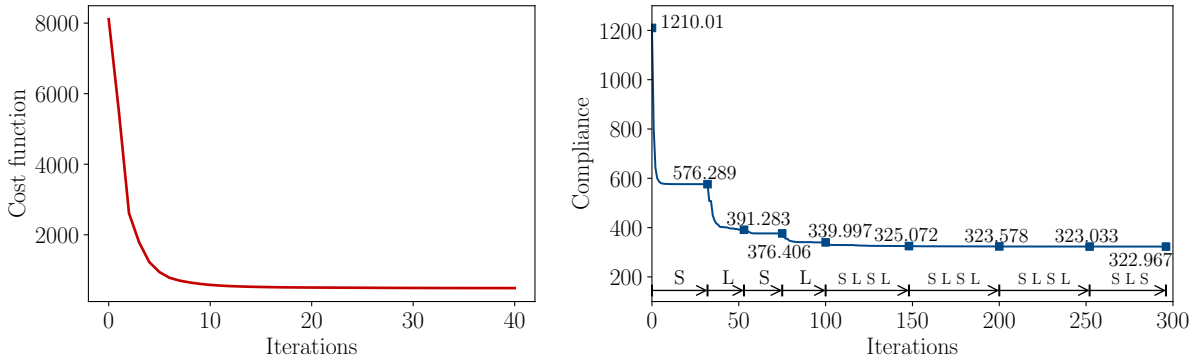
During the layout optimisation, the positions of the joints between the frame and plate are fixed. For comparison, the compliance of the topology optimised voxel model was $J(\hat{\boldsymbol{\rho}}) = 364.596$. The final optimised frame structure contains 56 joints and 64 members, see Figure 23b. Its solid CAD model and faceted triangular STL mesh exported from FreeCAD are depicted in Figure 25.



(a) Size optimised (initial)

(b) Size and layout optimised (final)

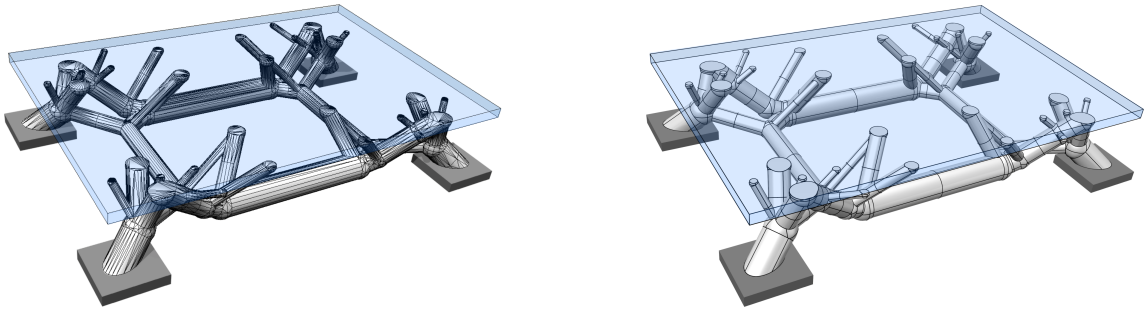
Figure 23: Size and layout optimised frame-supported plate. The plate on the top is not shown for clarity.



(a) Topology optimisation

(b) Size (S) and layout (L) optimisation

Figure 24: Convergence of the compliance during topology and sequential size and layout optimisation of the frame-supported plate. Compliance of the topology optimised voxel model is $J(\hat{\boldsymbol{\rho}}) = 364.596$.



(a) STL model

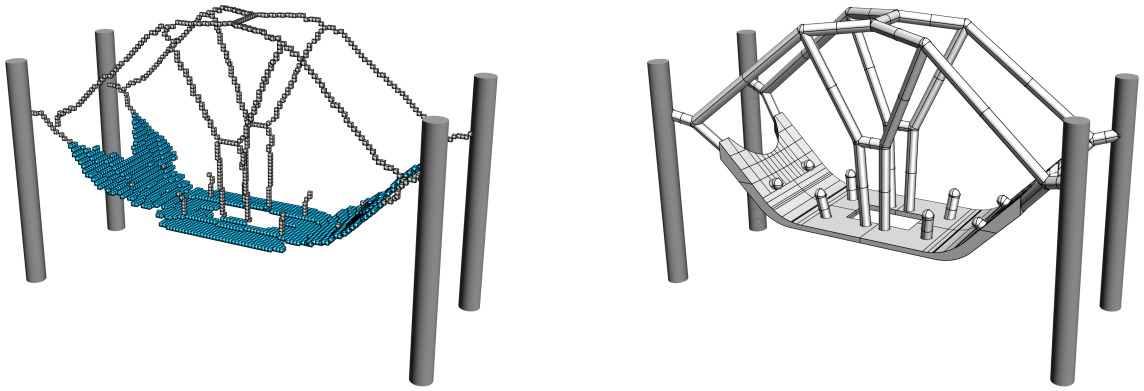
(b) IGES model

Figure 25: Parametric CAD model of the frame-supported plate.

7. Conclusions

The automated conversion of topology optimised geometries into compact parametric CAD models is a missing link in the wide-scale industrial adoption of optimisation. We introduced a fully automated workflow for synthesising a structurally-sound parametric CAD model from a voxelised solid-void binary image obtained with topology optimisation. Currently, there is no commercial design system, such as Altair Inspire, Abaqus CAE or similar, which offers such a fully automated CAD model generation capability. Homotopic skeletonisation of three-dimensional images is a key component of our proposed workflow, which is an extensively well-studied topic in digital topology. The resulting voxel chain skeleton is interpreted as a weighted undirected graph and processed with standard graph algorithms to yield a frame model. Both the skeletonisation and graph algorithms operate on topology rather than geometry and are not subject to floating-point errors, which makes them exceedingly robust. The so-obtained structural frame models are not optimal because the skeletonisation and graph algorithms do not take into account any mechanical considerations. However, as shown in the numerical examples, after several alternating steps of size and layout optimisation, the structural frames can achieve a cost function value similar to the original topology optimised geometry. The frame model is converted into a compact CAD solid model by recursively combining primitive shapes using boolean operations. Although we used in this paper only cylinders and spheres as shape primitives, it is straightforward to use other primitives. There is also scope in improving the CAD solid model itself, expressly, by using the recently proposed quadratic of revolution (quador) representation for lattices and frames [56–58]. A quador CAD model consists only of conic curves and surfaces, which greatly simplifies and accelerates frequently required geometry queries, like closest point computations or slicing, critical to meshing and manufacturing process planning.

The generated parametric CAD solid model and its compact CSG tree representation can be edited to refine the optimised design or to combine it with other components into a product. It is worth reminding that in industrial practice, as supported by user studies [59], geometries obtained from optimisation are often the starting point for design exploration and not the endpoint. This usually requires the ability to edit the optimised geometries in a CAD system. A high-level compact representation is also useful for taking into account geometric manufacturing constraints, like minimum thickness or slope or length of the members [30]. Beyond geometry editing, the frame model may also have advantages in structural analysis of the CAD model. The frame model makes it, for instance, easier to check the structure for buckling and inelastic deformations. As a low-order model, it can be analysed orders of magnitude faster than a three-dimensional solid model. This opens up the possibility of instant analysis of CAD models as it has become recently available in, e.g., Ansys Discovery Live or Creo Simulation Live.



(a) Voxel chain skeleton **(b)** IGES model

Figure 26: Surface-preserving skeletonisation and CAD model of the topology optimised pipe bracket in Figure 17a.

In closing, one particular extension of the presented approach is worth mentioning. We assumed that the topology optimised voxel model can be approximated with tubular members. However, in three-dimensional optimisation problems, large flat or curved surfaces may also appear during optimisation. The introduced skeletonisation algorithm in Section 3 will reduce even those into a network of one-voxel-wide chains. As suggested in Lee et al. [4], the

algorithm can be modified to preserve one-voxel-wide surfaces which appear during skeletonisation. This is achieved by not deleting the so-called surface points/voxels, which are defined based on the arrangement of solid voxels in an octant. In Figure 26 the topology optimised pipe bracket has been skeletonised with the modified skeletonisation algorithm. The curved surface at the bottom is preserved. In the structural model, this surface can be modelled as a thin-shell and in the CAD model it can be easily converted into a solid by providing a thickness. Currently, we do not have an automated process to extract a shell surface from the skeletonised voxel model, but this seems to be possible and is a promising direction for future research.

Appendix A. Member stiffness matrix

We model the members of the structural frame as Timoshenko beams. As mentioned, the members are assumed to be straight and have uniform cross-sections along their lengths. Hence, each member can be approximated with a single beam finite element and its stiffness matrix can be derived with standard structural analysis techniques from undergraduate textbooks, see e.g. [60]. The stiffness matrix we use takes into account axial stretch, transversal shear, bending and torsion effects. The local stiffness matrix $\boldsymbol{K}_i^l$ of a beam i is derived in a local coordinate system in which the beam axis is aligned with the x_i^l axis. Each of the two end nodes of the beam have three displacement and three rotation degrees of freedom, see Figure A.27. To ease the notation, we write in the following for the local coordinate axes only x, y and z.

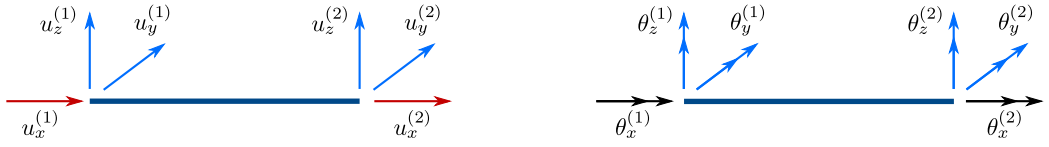
The element stiffness matrix of the beam is composed out of independent axial, bending and torsion stiffness matrices. The axial stiffness matrix corresponding to the displacement degrees of freedom $u_x^{(1)}$ and $u_x^{(2)}$ is given by

$$\frac{EA}{L}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \tag{A.1}$$

where E, A and L denote the Young's modulus, cross-sectional area and length.

The bending stiffness matrix corresponding to the displacement degrees of freedom $u_y^{(1)}$, $u_z^{(1)}$, $u_y^{(2)}$ and $u_z^{(2)}$ and the rotational degrees of freedom $\theta_y^{(1)}$, $\theta_z^{(1)}$, $\theta_y^{(2)}$ and $\theta_z^{(2)}$ is given by

$$\begin{pmatrix}
\frac{12EI_z}{(1+b_z)L^3} & 0 & 0 & \frac{6EI_z}{(1+b_z)L^2} & -\frac{12EI_z}{(1+b_z)L^3} & 0 & 0 & \frac{6EI_z}{(1+b_z)L^2} \\
 & \frac{12EI_y}{(1+b_y)L^3} & -\frac{6EI_y}{(1+b_y)L^2} & 0 & 0 & -\frac{12EI_y}{(1+b_y)L^3} & -\frac{6EI_y}{(1+b_y)L^2} & 0 \\
 & & \frac{(4+b_y)EI_y}{(1+b_y)L} & 0 & 0 & \frac{6EI_y}{(1+b_y)L^2} & \frac{(2-b_y)EI_y}{(1+b_y)L} & 0 \\
 & & & \frac{(4+b_z)EI_z}{(1+b_z)L} & -\frac{6EI_z}{(1+b_z)L^2} & 0 & 0 & \frac{(2-b_z)EI_z}{(1+b_z)L} \\
 & & & & \frac{12EI_z}{(1+b_z)L^3} & 0 & 0 & -\frac{6EI_z}{(1+b_z)L^2} \\
 & & \textit{sym.} & & & \frac{12EI_y}{(1+b_y)L^3} & \frac{6EI_y}{(1+b_y)L^2} & 0 \\
 & & & & & & \frac{(4+b_y)EI_y}{(1+b_y)L} & 0 \\
 & & & & & & & \frac{(4+b_z)EI_z}{(1+b_z)L}
\end{pmatrix}, \tag{A.2}$$



(a) Displacement degrees of freedom **(b)** Rotation degrees of freedom

Figure A.27: Degrees of freedom of a Timoshenko beam element (coupled degrees of freedom are depicted in the same colours).

where I_y and I_z are the second moments of area about the y and z axis. The two dimensionless factors b_y and b_z take into account the transversal shear and are defined according to $b = 12EI/(\kappa GAL^2)$, with G the shear modulus and κ the shear correction factor. In case of a circular cross section we have $I_y = I_z = I$, $b_y = b_z = b$ and $\kappa = 0.9$. Note that setting $b_y = b_z = 0$ in (A.2) gives the stiffness matrix of an Euler-Bernoulli beam.

Finally, the torsional stiffness matrix corresponding to the rotational degrees of freedom $\theta_x^{(1)}$ and $\theta_x^{(2)}$ is given by

$$\frac{GJ}{L}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \tag{A.3}$$

where J is the torsion constant of the cross section.

The local stiffness matrix $\boldsymbol{K}_i^l$ of an element i is obtained by suitably assembling the three stiffness matrices (A.1), (A.2) and (A.3) into a 12×12 matrix. This matrix in the local coordinate system x_i^l, y_i^l and z_i^l is transformed into a stiffness matrix $\boldsymbol{K}_i$ in the global coordinate system x, y and z according to (11).

Appendix B. Algorithms

The proposed approach essentially relies on the Algorithms 1 and 2 mentioned in Sections 3.2 and 4.1, respectively. The input of Algorithm 1 for skeletonisation consists of the binary voxel model with $\mathcal{V} = \mathcal{V}_s \cup \mathcal{V}_e$ and the tagged non-removable voxels $\mathcal{V}_t$. In each iteration (loop starting line 4) a layer of border voxels in one of the six coordinate directions is removed. Within each iteration first all potentially removable voxels are determined and then removed (loops starting lines 4 and 10 respectively). Only voxels which have an empty neighbour, are not the end point of a chain, are a simple point and are not tagged as non-removable can be potentially removed. According (22) a voxel v is a simple point if its removal does not change the Euler characteristic and the number of connected components of its 26-neighbourhood $\mathcal{N}_{26}(v)$. We determine the change of the Euler characteristic with a look-up table [4, 44, 45] and the number of connected components using the Boost Graph Library [61]. The inner loop starting line 10 is necessary because each voxel is present in 27 26-neighbourhoods and the removal of one voxel can potentially render any one of the neighbouring 26 voxels non-removable.

Algorithm 1: Skeletonise

Input: $\mathcal{V}, \mathcal{V}_s, \mathcal{V}_t$ // binary voxel model and tagged voxels

1 $c = 0$ // auxiliary counter

2 **while** $c \neq 6$ **do**

3 $\quad c \leftarrow 0$

4 $\quad$ **for** $i = 1$ **to** 6 **do** // loop over all coordinate directions

5 $\quad\quad \mathcal{D} = \emptyset$ // potentially deletable voxels

6 $\quad\quad$ **for** $v \in \mathcal{V}$ **do**

7 $\quad\quad\quad$ **if** $\begin{pmatrix} v \in \mathcal{V}_s \text{ and} \\ \text{neighbour}(i, v) \notin V_s \text{ and} \\ \text{isNotEndPoint } (v) \text{ and} \\ \text{isSimplePoint}(v) \text{ and} \\ v \notin \mathcal{V}_t \end{pmatrix}$ **then** $\mathcal{D} \leftarrow \mathcal{D} \cup \{v\}$

8 $\quad\quad$ **end**

9 $\quad\quad n_s = |V_s|$

10 $\quad\quad$ **for** $v \in \mathcal{D}$ **do** // delete solid voxels

11 $\quad\quad\quad$ **if** isSimplePoint(v) **then** $\mathcal{V}_s \leftarrow \mathcal{V}_s \setminus \{v\}$

12 $\quad\quad$ **end**

13 $\quad\quad$ **if** $n_s = |V_s|$ **then** $c \leftarrow c + 1$

14 $\quad$ **end**

15 **end**

Output: $\mathcal{V}_s$ // voxel skeleton

Algorithm 2: ExtractGraph

```
   Input: V_s, V_t                                          // voxel skeleton and tagged voxels
   // initialise graph G with nodes P, edges E
   // and function w mapping edges to weights
 1 G = (P, E, w)
 2 P ← ∅; E ← ∅
 3 for v ∈ V_s do
 4     if jointOrTagged(v) then
 5         P ← P ∪ {v}                                    // edge starting at joint v
 6         for s ∈ N_26(v) do                                  // trace voxel chain
 7             ℓ = 2; s_prev = v
 8             while ( not jointOrTagged(s) ) do
 9                 s_next = N_26(s) \ {s_prev}
10                 s_prev = s; s = s_next
11                 ℓ ← ℓ + 1
12             end
13             P ← P ∪ {s}                                // edge ending at joint s
14             E ← E ∪ {{v, s}}
15             w({v, s}) ← ℓ
16         end
17     end
18 end
   Output: G = (P, E, w)                                   // weighted undirected graph
```

The obtained voxel skeleton is converted with Algorithm 2 into a weighted undirected graph. Each joint voxel with $|\mathcal{N}_{26}(v)| \neq 2$ or tagged voxel $v \in \mathcal{V}_t$ becomes a node of the graph. The edges are obtained by starting from a joint voxel v and marching along all its neighbours $\mathcal{N}_{26}(v)$, in turn, until a second joint or tagged voxel is reached (loop starting line 6). The edge weight is obtained by incrementing the variable ℓ during the marching (line 11). In Algorithm 2 each edge is traced twice which is redundant and could be improved.

References

[1] M. P. Bendsoe, Optimal shape design as a material distribution problem, Structural Optimization 1 (1989) 193–202.
[2] O. Sigmund, A 99 line topology optimization code written in Matlab, Structural and Multidisciplinary Optimization 21 (2001) 120–127.
[3] M. P. Bendsoe, O. Sigmund, Topology optimization: theory, methods and applications, Springer, 2003.
[4] T.-C. Lee, R. L. Kashyap, C.-N. Chu, Building skeleton models via 3-D medial surface axis thinning algorithms, CVGIP: Graphical Models and Image Processing 56 (1994) 462–478.
[5] M. Y. Wang, X. Wang, D. Guo, A level set method for structural topology optimization, Computer Methods in Applied Mechanics and Engineering 192 (2003) 227–246.
[6] G. Allaire, F. Jouve, A.-M. Toader, Structural optimization using sensitivity analysis and a level-set method, Journal of Computational Physics 194 (2004) 363–393.
[7] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Computer Methods in Applied Mechanics and Engineering 71 (1988) 197–224.
[8] O. Sigmund, K. Maute, Topology optimization approaches, Structural and Multidisciplinary Optimization 48 (2013) 1031–1055.
[9] X. Guo, W. Zhang, W. Zhong, Doing topology optimization explicitly and geometrically—a new moving morphable components based framework, Journal of Applied Mechanics 81 (2014) 081009.
[10] J. A. Norato, B. K. Bell, D. A. Tortorelli, A geometry projection method for continuum-based topology optimization with discrete elements, Computer Methods in Applied Mechanics and Engineering 293 (2015) 306–327.
[11] Y. Zhou, W. Zhang, J. Zhu, Z. Xu, Feature-driven topology optimization method with signed distance function, Computer Methods in Applied Mechanics and Engineering 310 (2016) 1–32.
[12] X. Xie, S. Wang, M. Xu, Y. Wang, A new isogeometric topology optimization using moving morphable components based on R-functions and collocation schemes, Computer Methods in Applied Mechanics and Engineering 339 (2018) 61–90.
[13] V. Braibant, C. Fleury, Shape optimal design using b-splines, Computer Methods in Applied Mechanics and Engineering 44 (1984) 247–267.
[14] K.-U. Bletzinger, S. Kimmich, E. Ramm, Efficient modeling in shape optimal design, Computing Systems in Engineering 2 (1991) 483–495.
[15] T. T. Robinson, C. G. Armstrong, H. S. Chua, C. Othmer, T. Grahs, Optimizing parameterized CAD geometries using sensitivities based on adjoint functions, Computer-Aided Design and Applications 9 (2012) 253–268.
[16] X. Han, D. W. Zingg, An adaptive geometry parametrization for aerodynamic shape optimization, Optimization and Engineering 15 (2014) 69–91.
[17] F. Cirak, M. J. Scott, E. K. Antonsson, M. Ortiz, P. Schröder, Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision, Computer-Aided Design 34 (2002) 137–148.
[18] W. A. Wall, M. A. Frenzel, C. Cyron, Isogeometric structural shape optimizisation, Computer Methods in Applied Mechanics and Engineering 197 (2008) 2976–2988.
[19] J. Kiendl, K.-U. Bletzinger, J. Linhard, W. R., Isogeometric shell analysis with Kirchhoff-Love elements, Computer Methods in Applied Mechanics and Engineering 198 (2009) 3902–3914.
[20] K. Bandara, T. Rüberg, F. Cirak, Shape optimisation with multiresolution subdivision surfaces and immersed finite elements, Computer Methods in Applied Mechanics and Engineering 300 (2016) 510–539.
[21] A. J. Herrema, N. M. Wiese, C. N. Darling, B. Ganapathysubramanian, A. Krishnamurthy, M.-C. Hsu, A framework for parametric design optimization using isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 316 (2017) 944–965.
[22] K. Bandara, F. Cirak, Isogeometric shape optimisation of shell structures using multiresolution subdivision surfaces, Computer-Aided Design 95 (2018) 62–71.
[23] K. Maute, E. Ramm, Adaptive topology optimization, Structural Optimization 10 (1995) 100–112.
[24] C.-Y. Lin, L.-S. Chao, Automated image interpretation for integrated topology and shape optimization, Structural and Multidisciplinary Optimization 20 (2000) 125–137.
[25] Y.-L. Hsu, M.-S. Hsu, C.-T. Chen, Interpreting results from topology optimization using density contours, Computers & Structures 79 (2001) 1049–1058.
[26] M. Bremicker, M. Chirehdast, N. Kikuchi, P. Y. Papalambros, Integrated topology and shape optimization in structural design, Journal of Structural Mechanics 19 (1991) 551–587.
[27] J.-C. Cuillière, V. François, A. Nana, Automatic construction of structural CAD models from 3D topology optimization, Computer-Aided Design and Applications 15 (2018) 107–121.
[28] S. Pellegrino, Structural computations with the singular value decomposition of the equilibrium matrix, International Journal of Solids and Structures 30 (1993) 3025–3035.
[29] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, Skeleton extraction by mesh contraction, ACM Transactions on Graphics 27 (2008) 44:1–44:10.
[30] J. Liu, Y. Ma, A survey of manufacturing oriented topology optimization methods, Advances in Engineering Software 100 (2016) 161–175.
[31] N. D. Cornea, D. Silver, P. Min, Curve-skeleton properties, applications, and algorithms, IEEE Transactions on Visualization & Computer Graphics 13 (2007) 530–548.
[32] K. Siddiqi, S. Pizer, Medial representations: mathematics, algorithms and applications, Springer, 2008.
[33] P. K. Saha, G. Borgefors, G. S. di Baja, A survey on skeletonization algorithms and their applications, Pattern Recognition Letters 76 (2016) 3–12.
[34] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, A. Telea, 3d skeletons: A state-of-the-art report, in: Computer Graphics Forum, vol. 35, 573–597, 2016.
[35] P. K. Saha, G. Borgefors, G. S. di Baja, Skeletonization: Theory, methods and applications, Academic Press, 2017.
[36] D. J. Sheehy, C. G. Armstrong, D. J. Robinson, Shape description by medial surface construction, IEEE Transactions on Visualization and Computer Graphics 2 (1996) 62–72.
[37] T. Y. Kong, A. Rosenfeld, Digital topology: Introduction and survey, Computer Vision, Graphics, and Image Processing 48 (1989) 357–393.

[38] R. Klette, A. Rosenfeld, Digital geometry: Geometric methods for digital picture analysis, Elsevier, 2004.
[39] L. Lam, S.-W. Lee, C. Y. Suen, Thinning methodologies-a comprehensive survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992) 869–885.
[40] S. Lobregt, P. W. Verbeek, F. C. A. Groen, Three-dimensional skeletonization: principle and algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (1980) 75–77.
[41] C.-M. Ma, S.-Y. Wan, J.-D. Lee, Three-dimensional topology preserving reduction on the 4-subfields, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 1594–1605.
[42] C. Lohou, G. Bertrand, A 3D 6-subiteration curve thinning algorithm based on P-simple points, Discrete Applied Mathematics 151 (2005) 198–228.
[43] Y. Yan, D. Letscher, T. Ju, Voxel Cores: Efficient, robust, and provably good approximation of 3D medial axes, ACM Transactions on Graphics (TOG) 37 (2018) 1–13.
[44] H. Homann, Implementation of a 3D thinning algorithm, URL http://hdl.handle.net/1926/1292, 2007.
[45] T. Post, C. Gillmann, T. Wischgoll, H. Hagen, Fast 3D Thinning of Medical Image Data based on Local Neighborhood Lookups, in: EuroVis 2016 - Short Papers, 2016.
[46] K. K. Choi, N.-H. Kim, Structural sensitivity analysis and optimization 1: linear systems, vol. 1, Springer, 2006.
[47] A. Hatcher, Algebraic topology, Cambridge University Press, 2009.
[48] M. D. Crossley, Essential topology, Springer, 2006.
[49] S. Even, Graph algorithms, Cambridge University Press, 2011.
[50] K. Museth, J. Lait, J. Johanson, J. Budsberg, R. Henderson, M. Alden, P. Cucka, D. Hill, A. Pearce, OpenVDB: an open-source data structure and toolkit for high-resolution volumes, in: SIGGRAPH 2013 Courses, ACM, 2013.
[51] C. J. Smith, M. Gilbert, I. Todd, F. Derguti, Application of layout optimization to the design of additively manufactured metallic components, Structural and Multidisciplinary Optimization 54 (2016) 1297–1313.
[52] R. Arora, A. Jacobson, T. R. Langlois, Y. Huang, C. Mueller, W. Matusik, A. Shamir, K. Singh, D. I. Levin, Volumetric Michell Trusses for Parametric Design & Fabrication, in: Proceedings of the 3rd ACM Symposium on Computation Fabrication, SCF '19, 2019.
[53] S. G. Johnson, The NLopt nonlinear-optimization package, URL http://github.com/stevengj/nlopt, 2014.
[54] J. R. R. A. Martins, N. M. K. Poon, On Structural Optimization Using Constraint Aggregation, in: Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization, 2005.
[55] X. Xiao, M. Sabin, F. Cirak, Interrogation of spline surfaces with application to isogeometric design and analysis of lattice-skin structures, Computer Methods in Applied Mechanics and Engineering 351 (2019) 928–950.
[56] A. Gupta, G. Allen, J. Rossignac, Quador: Quadric-of-revolution beams for lattices, Computer-Aided Design 102 (2018) 160–170.
[57] A. Gupta, G. Allen, J. Rossignac, Exact Representations and Geometric Queries for Lattice Structures with Quador Beams, Computer-Aided Design 115 (2019) 64–77.
[58] F. Cirak, M. Sabin, Adding quadric fillets to quador lattice structures, Computer-Aided Design 118 (2020) 102754.
[59] E. Bradner, F. Iorio, M. Davis, Parameters tell the design story: ideation and abstraction in design optimization, in: Proceedings of the Symposium on Simulation for Architecture & Urban Design, 26, 2014.
[60] T. H. G. Megson, Aircraft structures for engineering students, Butterworth-Heinemann, 2016.
[61] J. Siek, A. Lumsdaine, L.-Q. Lee, The Boost graph library: user guide and reference manual, Addison-Wesley, 2002.