SGLDBench

A Benchmark Suite for Stress-Guided Lightweight 3D Designs

Wang, Junpeng; Bukenberger, Dennis R.; Niedermayr, Simon; Neuhauser, Christoph; Wu, Jun; Westermann, Rudiger

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

SGLDBench: A Benchmark Suite for Stress-Guided Lightweight 3D Designs

Junpeng Wang [ID], Dennis R. Bukenberger, Simon Niedermayr, Christoph Neuhauser [ID], Jun Wu [ID],
and Rüdiger Westermann [ID]

Abstract—We introduce the Stress-Guided Lightweight Design Benchmark (SGLDBench), a comprehensive benchmark suite for applying and evaluating material layout strategies to generate stiff, lightweight designs in 3D domains. SGLDBench provides a seamlessly integrated simulation and analysis framework, including six reference strategies and a scalable multigrid elasticity solver to efficiently execute these strategies and validate the stiffness of their results. This facilitates the systematic analysis and comparison of design strategies based on the mechanical properties they achieve. SGLDBench enables the evaluation of diverse load conditions and, through the tight integration of the solver, supports high-resolution designs and stiffness analysis. Additionally, SGLDBench emphasizes visual analysis to explore the relationship between the geometric structure of a design and the distribution of stresses, offering insights into the specific properties and behaviors of different design strategies. SGLDBench's specific features are highlighted through several experiments, comparing the results of reference strategies with respect to geometric and mechanical properties.

Index Terms—Topology optimization, lattice infill, lightweight design, simulation design.

I. INTRODUCTION

TOPOLOGY optimization (TO) and functionally graded lattice infill are primary strategies for designing mechanically sound, lightweight structures, i.e., structures with high stiffness (corresponding to a low compliance, or degree of deformability) under applied loads. TO determines the optimal material distribution within a given design domain to achieve a desired structural performance, such as maximizing stiffness, while satisfying constraints like material use [1], [2]. Functionally graded lattice infill refers to a design approach in which the density, size, shape, or material properties of the lattice structure vary spatially within a 3D object to meet specific performance requirements [3]. For beam-based lattices, the final design can be represented by a graph or grid composed of polyhedral cells, each constructed from individual edges.

TO, in its basic form, does not consider the geometric structure of the resulting material layout but aims to achieve the highest possible stiffness. Lattice infill design strategies, in principle, share this goal, by tailoring the lattice layout based on the stress distribution. Stress is a measure of the internal forces that develop within a material when it is subjected to external loads and quantifies the intensity of these forces at a specific point in the material. The structural rigidity of an infill increases when the material aligns with the orthogonal principal stress directions of the object under load [4]. These directions, corresponding to the eigenvectors of the 3x3 stress tensor, indicate the normal stresses acting on specific planes within where shear stresses are zero. At the limit of material volume, considering these directions for the lattice layout results in microstructures resembling quads or hexahedra [5], [6].

Each lattice infill design strategy, however, involves additional considerations that may compromise stiffness. These include achieving geometric properties such as regularity (i.e., variation in element type), uniformity (i.e., variation in element size), or space-fillingness to enhance robustness, as well as purely aesthetic features [3], [7], [8], [9].

Moreover, generating 3D domain-filling lattice structures that align with major stress directions presents significant challenges. This difficulty arises from the existence of degenerate points [10] (or degenerate regions in 3D domains [11], [12], [13]) where the stress tensor has repeating eigenvalues, making the principal stress directions indeterminate. As a result, integrability conditions are violated, and consistent domain parameterizations cannot be computed [14].

To assist users in selecting the right 3D lightweight design strategy for various use cases, and to help researchers identify open research questions, a benchmark suite for generating, analyzing, and comparing the results of different strategies is essential.

A few benchmark papers have addressed issues such as special solvers for TO [15], benchmarks for 2D TO in specific load cases [16], practices that should be considered when performing TO [17], as well as the mechanical soundness of simple lattice infills such as orthogonal grids and shells [18]. Different architectures for multidisciplinary design optimization have been reviewed and compared [19], and the design and structural optimization of lightweight design has been discussed, especially in the context of additive manufacturing [7]. A review

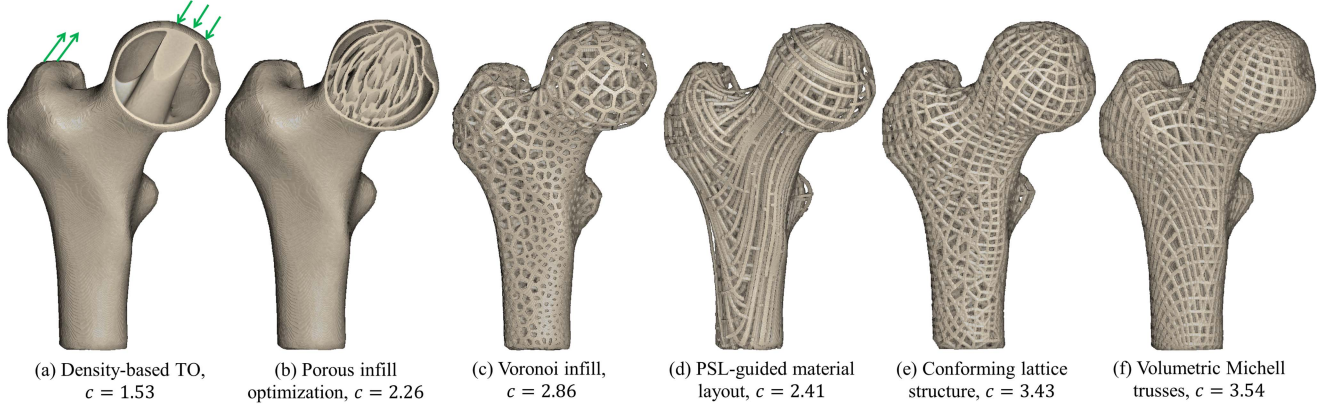| (a) Density-based TO, $c = 1.53$ | (b) Porous infill optimization, $c = 2.26$ | (c) Voronoi infill, $c = 2.86$ | (d) PSL-guided material layout, $c = 2.41$ | (e) Conforming lattice structure, $c = 3.43$ | (f) Volumetric Michell trusses, $c = 3.54$ |

Fig. 1. Infill designs computed using the strategies provided by SGLDBench. All designs consume roughly the same amount of material and are subject to the same boundary conditions: the bottom of the design domain is fixed, and the loads are indicated by green arrows in (a). Below each design, its compliance c is provided. For visualization, isosurface volume rendering is used. All designs are coated with fully solid boundary elements of consistent thickness. In (c) to (f), these elements are peeled away to reveal the infills.

of uniform and non-uniform lattice structures such as foams and honeycombs sheds light on their properties and methods for designing and optimizing such structures [20], [21]. Unit cell lattices comprising structures made of a single type of cells have been researched [22], and the properties of certain types of lattice infills regarding 3D printing processes have been discussed [23]. The combination of TO and micro element-based lattice infills have resulted in structures exhibiting anisotropic mechanical properties [24], [25]. There is no benchmark that allows researchers and users to efficiently compute 3D designs with different strategies and to effectively compare the results with respect to their mechanical and structural properties.

We introduce *SGLDBench* to address this gap. It provides a comprehensive investigation of the combination of boundary shapes and conditions with lightweight design strategies in 3D domains. This is facilitated by a seamlessly integrated, MATLAB-based simulation and analysis framework offering the following key features:

- *Selection of Lightweight 3D Design Strategies:* The benchmark includes various strategies, enabling comparisons between TO and lattice infill, as well as studies of new scenarios and designs. For a 3D human femur under load, Fig. 1 shows visualizations of the infill designs computed by SGLDBench.
- *Material Layout Generation:* A central feature of the benchmark is the voxelization of complex infills into a Cartesian simulation grid. This ensures consistent comparisons of lightweight designs—whether represented as a material field, mesh, or edge graph—with respect to their mechanical properties.
- *Simulation Suite:* SGLDBench provides a MATLAB-interfaced simulation framework with an efficient multigrid solver for generating high-resolution stress fields and assessing the stiffness of a design efficiently.
- *Visual Design Analysis:* A fast volume visualization module accompanies the benchmark, offering visual feedback on a design's shape, stress distribution, and material alignment before and after layout optimization.

We chose MATLAB as the working environment due to its widespread use in computational design. SGLDBench leverages MATLAB's simulation and visualization capabilities for design generation and analysis. New design strategies can be integrated either through MATLAB programs or by using MATLAB's functionality to call executables or Python programs from other codebases via inline calls. All SGLDBench-specific operations have been implemented in MATLAB or rely on publicly available MATLAB, C++, or Python programs. SGLDBench also uses external libraries for core operations such as meshing and voxelization. The visualization module is implemented in WebGL, allowing it to function either as a standalone viewer in a web browser or via inline calls to MATLAB's viewing functionality. The entire codebase for SGLDBench can be accessed through the link https://github.com/PSLer/SGLDBench.

II. SGLDBENCH'S FUNCTIONAL STRUCTURE

We begin by introducing the key functionality of SGLD-Bench. SGLDBench provides an interface to enable users to specify the **boundary conditions**, which include the boundary of a domain, the applied loads, and the fixed boundary regions. The domain is then voxelized, meaning it is discretized into a Cartesian grid, with per-voxel properties assigned based on the boundary conditions and material properties. We refer to this configuration as a **preset**.

Using the selected preset, SGLDBench **simulates the object's internal stress field** via a multigrid finite-element elasticity solver [26], [27], [28], [29]. While some design strategies require repeated solver executions to iteratively optimize the material layout, others generate an infill structure guided by the principal stresses in the initial field.

Users can choose from **six layout strategies** for computing an **infill design**: density-based TO [30], porous infill optimization [31], Voronoi infill [32], stress-line-guided material layout [33], conforming lattice structures [34], and volumetric Michell trusses [35].
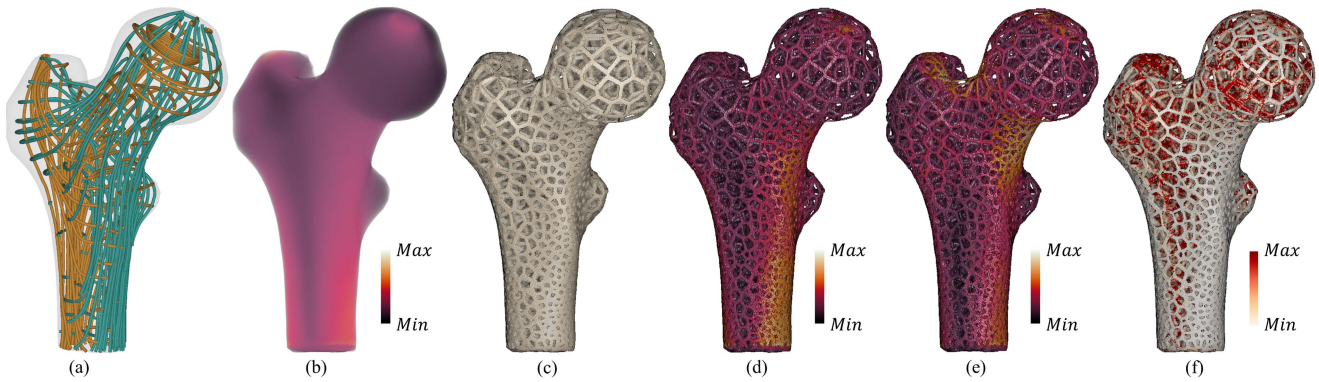
Fig. 2. SGLDBench's visual analysis options. (a) Major (brown) and minor (green) PSLs according to boundary conditions from Fig. 1. (b) Direct volume rendering of scalar von Mises stresses in the solid domain. (c) Voronoi infill (c = 2.86). (d) Infill with color-coded von Mises stresses. (e) Same as (d), but different forces apply and von Mises stresses change (c = 4.14). (f) Comparative visual stress analysis showing the misalignment between the major stress directions in the solid and the infill under the same loads.

Our selection has not been made with the intention to favor any of these methods, but to reveal the specific characteristics of 3D design strategies following different objectives. The methods span the spectrum from purely stiffness-based optimization to geometry-aware infill generation. We select **density-based TO** as a representative of various TO approaches. It serves as a reference for the stiffness that can be achieved. **Porous infill optimization**, while not explicitly using the principal stress directions, results in wall-like structures that largely agree with two of these directions. **Voronoi infill** considers only the stress magnitude but not its principal directions. In contrast, material layouts guided by the **Principal Stress Lines (PSLs)** follow exactly the mutually orthogonal principal stress trajectories in the initial solid domain. PSL-guided infills serve as a reference conveying these directions, even though the final designs are not connected in general. **Conforming lattice structures** and **volumetric Michell trusses** aim at finding a balance between stress alignment and geometric regularity of the designs. While the first approach favors stress alignment and, therefore, needs to resort to an edge-graph structure, the latter approach strives for a pure hexahedral mesh and, therefore, needs to sacrifice stress alignment.

While TO and porous infill generate a material field, other methods compute a lattice structure composed of edges and nodes. SGLDBench **voxelizes** these structures into a material field on a Cartesian grid with the same resolution as the initial preset. Using the material field and boundary conditions, the elasticity solver computes the **compliance of the design**. When using iterative optimization methods, the compliance history is recorded and can be visualized.

SGLDBench supports different **visualization options** to inspect a 3D design. The principal stress directions in a stress field are visualized using **PSL-guided trajectory visualization** implemented in MATLAB programs [36]. These programs are accessible through SGLDBench's interface and allow users to customize the number and appearance of the visualized trajectories (see Fig. 2(a)).

From the principal stresses the **scalar von Mises stresses** are computed. The von Mises stress is commonly used in engineering and materials science to predict yielding in ductile materials under load. It provides a single value that reflects the combined effect of all stress components acting on a material. This scalar field can then be visualized with SGLDBench's **WebGL-based visualization module** for enhanced rendering performance (Fig. 2(b)). An infill structure is rendered as an iso-surface in the material field (Fig. 2(c)), and it can be color-coded with the von Mises stress to reveal local stress concentrations (Fig. 2(d)).

Additionally, a **variable load structural analysis** can be performed, by loading a design with forces different from those for which it was initially optimized. This functionality enables users to evaluate the robustness of a design under different loading conditions. The optimized design can be color-coded with the von Mises stresses occurring under the new load conditions (Fig. 2(e)).

Furthermore, SGLDBench offers tailored visualization options to examine how the mechanical properties of the initial solid and the generated infill design have changed. Direct volume rendering is used with a predefined color transfer function to visualize the **per-voxel stress deviations** in the final design relative to those in the initial solid body (see Fig. 2(f)).

III. COMPONENTS OF SGLDBENCH

We describe here the most important features and operations of SGLDBench, including descriptions of the supported TO and lattice infill methods. The use of SGLDBench is demonstrated in the accompanying videos.

A. Domain Specification

SGLDBench simulates a stress field in the design domain using Finite Element Analysis (FEA). This requires discretizing the domain into finite elements and specifying boundary conditions. SGLDBench uses a hexahedral finite-element representation to facilitate the use of scalable geometric multigrid solvers. Therefore, SGLDBench first converts an initial object representation to a hexahedral simulation grid.
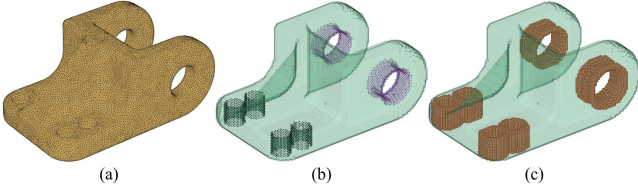
Fig. 3. (a) A triangle mesh defines the domain boundary. (b) After voxelizing the domain, the user specifies boundary forces and fixes grid vertices. Black and violet dots indicate fixed vertices and vertices subjected to an external force, respectively. (c) Passive elements are shown in brown.

Fig. 4. Element-based computation layout. M is the number of hexahedral simulation elements, K_{e0} is the generic element stiffness matrix and U_e is the element-wise displacement matrix. Different colors represent the 8 node-based displacement vectors per element. With each column l_i in U_e, MATLAB computes the product $K_{e0} \cdot l_i$.

Voxelization: The simulation grid is created by voxelizing the simulation domain. The user provides the domain boundary as a closed triangular mesh. SGLDBench utilizes MATLAB's voxelization capabilities [37] with a user-defined voxel resolution to compute a solid voxelization. For complex-shaped simulation domains, the voxels are classified as solid or void, depending on the centroid of the voxel. Alternatively, users can upload a 3D voxel grid that discretizes the domain and marks each voxel as solid or void. Void elements are excluded from the finite element analysis.

Voxels with at least one of their 26 neighboring voxels classified as void are designated as boundary voxels. SGLDBench applies a dilation operation to expand the boundary by assigning any voxel adjacent to an initial boundary voxel as a new boundary voxel. This voxelized object serves as the foundation for all subsequent operations in SGLDBench.

Boundary Conditions: The boundary conditions define where the object is fixed and how loads are applied. Fixations and forces are assigned to the nodes of the boundary elements. The user specifies the extent and position of an axis-aligned box or sphere, and SGLDBench automatically fixes or applies the specified loads to all boundary nodes within this region. Similarly, the nodes to be reset can be selected in the same manner. When all nodes of a finite element are fixed, the element becomes rigid and does not respond to any loads.

Passive Elements: Passive elements are used to preserve specific geometric features, such as object boundaries or notches for mounting connections. Passive elements remain solid throughout the optimization process and contribute to the stiffness of the structure. SGLDBench supports two general methods for specifying passive elements: Setting all boundary elements as passive, which is common in infill design problems, and setting elements with loaded or fixed nodes as passive to preserve geometric features during optimization. The dilation operation can also be used to enlarge passive structures as needed. Fig. 3 illustrates the transformation from a boundary mesh to a voxel model, including boundary conditions and passive elements.

B. Stress Simulation

At the core of TO and lattice infill methods is the numerical simulation of a stress tensor field using the selected boundary conditions and material properties. SGLDBench provides a MATLAB-interfaced C++ implementation of a multigrid elasticity solver to efficiently simulate the stress field.
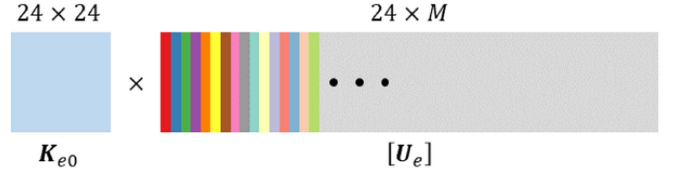
The implementation employs a geometric multigrid solver as a preconditioner for a conjugate gradient method to solve a sparse linear system of equations, i.e., $KU = F$. The global stiffness matrix K is assembled from the element stiffness matrices under the assumption of a linear material law. The computation of the element stiffness matrices accounts for the stiffness tensor, which reflects material properties, and the strain matrix, which expresses the strain-stress relationship. U represents the static displacement vector in response to the external loads F. Several prior works have addressed the efficient assembly of the system matrix K and the specific adaptations of numerical solvers for linear elasticity simulations in TO [26], [27], [28], [29], [38].

SGLDBench's multigrid implementation is primarily based on the work of Wu et al. [27], utilizing on-the-fly numerical stencil assembly and multigrid interpolation and restriction across multiple levels simultaneously. However, the implementation has been adapted for MATLAB running on a CPU, resulting in changes to the internal data and computation layouts.

First, SGLDBench uses MATLAB's built-in Cholesky solver rather than the TAUCS library's Cholesky solver for solving the linear system on the coarsest multigrid level. Second, it transitions from a matrix-free node-based computation layout to a matrix-free element-based layout to take advantage of MATLAB's efficient matrix-vector operations. Instead of assembling stencils per grid vertex on-the-fly using indexed memory access operations, SGLDBench constructs a generic element matrix and utilizes MATLAB to compute the products of this matrix with the 8-node displacement vectors of each element. Since the stiffness matrix of an element with density ρ is obtained by correspondingly scaling the generic stiffness matrix, the final results only need to be scaled accordingly. For each element, the 8 displacement vectors at the vertices are organized into columns of an element-wise displacement matrix, as illustrated in Fig. 4. MATLAB then computes all matrix-vector products between the generic element matrix and each column in the displacement matrix efficiently.

C. Infill Computation

For each of the six lightweight design methods provided by SGLDBench, the user selects specific parameters and lets SGLDBench compute the material layout. SGLDBench provides MATLAB code for density-based TO, porous infill optimization, PSL-guided material layout and volumetric Michell
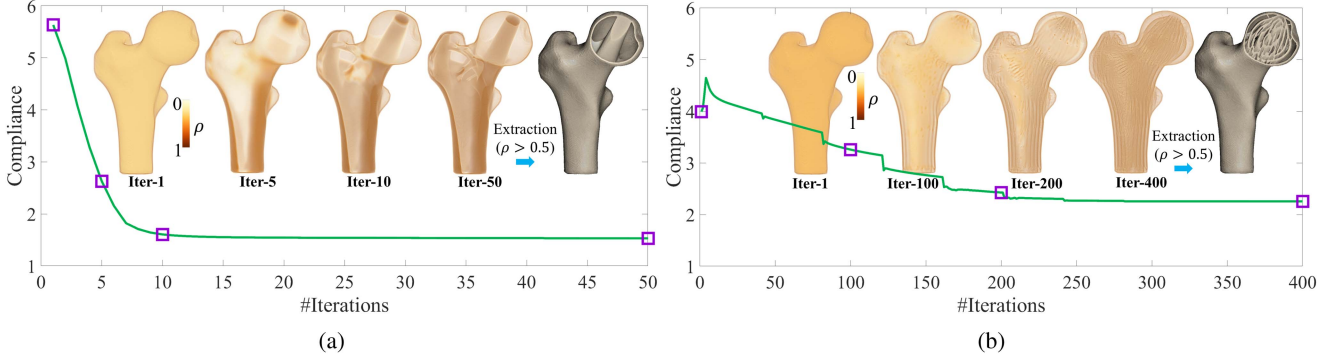
Fig. 5. Material layout optimization with density-based TO (a) and porous infill optimization (b). Compliance history is depicted by the green curves, with violet squares indicating shown states.

trusses. To generate a Voronoi infill, MATLAB calls a Python script including precompiled libraries. Conforming lattice structures are generated by compiling code from a publically available repository and running the executable from MATLAB with the required parameters.

1) Topology Optimization: TO minimizes the compliance of a material layout under the constraint of applied forces and a prescribed global material consumption. SGLDBench implements the density-based TO approach [30], [39]. To formulate the minimization problem over a discrete set of elements e with densities ρ_e, a hexahedral finite element discretization of a linear elastic solid material is generated from the voxelized geometry. The object's compliance c is computed by summing the strain energy over all material elements, i.e.,

$$c = U^{\mathrm{T}} K U. \qquad (1)$$

The lower the compliance, the higher the object's stiffness.

With selected measure of a material's ability to deform under an applied stress, i.e., the Young's modulus E_0 of the solid ($\rho_e = 1$), and the linear material law, TO proceeds in three steps: 1) A large linear system is solved using the MATLAB multigrid implementation to compute the force-induced displacements of the hexahedral vertices. 2) The derivatives of the total strain energy c and the total volume V with respect to the elements' densities ρ_e are computed and used to guide the material distribution to maximize stiffness. 3) The design is updated according to the computed sensitivities. These steps are repeated until the change in material distribution is below a threshold or the number of iterations reaches the prescribed maximum iterations. The computational pipeline for density-based TO is mainly written in MATLAB, using our optimized C++ code for solving the linear system and updating the design variables.

Density-based TO takes the available material budget V_0 as the constraint, known as the global volume constraint. Thus, the problem-specific constraint function for a material layout ϕ using n hexahedral elements is

$$g(\phi) = \sum \rho_e - nV_0 \le 0 \qquad (2)$$

We use the so-called Solid Isotropic Material with Penalization (SIMP) model, where a non-zero constant minimum value

represents the background stiffness of the void region. In each optimization iteration, the design variables are updated within a prescribed step size through a gradient-based optimizer using the optimality criteria method [30]. After optimization, the design variables shall converge to a (near-)binary layout that indicates the spatial material distribution.

It's worth mentioning that several auxiliary processes are also introduced in practical TO for good design quality. For instance, the density-based filtering to counteract numerical instabilities and the Heaviside projection to promote the generation of a binary design, where the proxy density value of each voxel is encoded by the projected value of the filtered value of the design variable [40].

Fig. 5(a) shows the optimized shapes after different optimization iterations. The optimization produces a mono-scale design comprising mainly a thick resistant strand along the maximum stress directions.

2) Porous Infill Optimization: Porous infill optimization is an extension of density-based TO that generates porous substructures distributed across the design domain. This is achieved by replacing the global volume constraint with local volume constraints, which prevent material from accumulating and forming dense, solid regions.

The global volume constraint restricts the total material consumption within the entire simulation domain. The local volume constraint imposes an upper bound V_e on the percentage of solid voxels within a prescribed neighborhood of voxel e. These local constraints ensure a more evenly distributed material layout, promoting porosity and lightweight design. Beyond this adjustment, the optimization process largely follows the approach used in density-based TO.

For the material around each voxel e, the local volume constraint leads to the constraint function

$$g(\phi_e) = \frac{\sum_{i \in N_e} \rho_i}{\sum_{i \in N_e} 1} - V_e \le 0. \qquad (3)$$

N_e defines the voxel neighborhood that is considered, i.e.,

$$N_e = \{i \mid \; \| \, x_i - x_e \, \|_2 \le R_e\}, \;\; \forall e. \qquad (4)$$

R_e is the radius of a spherical region centered at a voxel's center, It defines the area within which local material accumulation is

measured. SGLDBench's implementation of density-based TO in MATLAB has been extended to include the specific constraint function for porous infill optimization. The optimization process uses the Method of Moving Asymptotes (MMA) [41] as the optimizer to iteratively update the design variables.

Fig. 5(b) illustrates the optimization process of porous infill optimization. Unlike standard TO, porous infill optimization generates a space-filling, multi-scale design. These designs generally exhibit lower stiffness compared to those produced by TO with a global volume constraint, as some material is deposited in regions that do not significantly contribute to overall stiffness. However, such designs are typically more robust under varying load conditions and localized damage [31], [33]. Additionally, porous infill optimization in 3D tends to form wall-like structures aligned with the major and minor principal stress directions, as shown in Fig. 1(b).

3) Voronoi Infill: 3D Voronoi infills are generated by computing an initial Delaunay tetrahedralization, based on a set of samples (S) following a stress-based distribution density. Therefore, more samples are generated in regions of higher stress, whereas the sampling density is lower in less stress-critical regions [32]. The Voronoi mesh follows as the dual of the Delaunay complex. Procedural infill optimization techniques building upon similar concepts have been proposed for additive manufacturing [42], [43].

Graded Sampling: In SGLDBench, S is initialized with a small set of auxiliary samples, equally distributed on a sphere, fully enclosing the input object. Further initial samples are added from the set of vertices of the input object's hull. Then, the input tetrahedral mesh is used as sampling domain, where S is iteratively updated in a progressive Poisson disk sampling scheme until no further samples can be added. For improved performance, this is realized using batches of n samples per iteration and organizing S in a kd-tree. Radii for Poisson disks are interpolated at their sample positions based on the von Mises stress field σ_{vm}, using the mapping

$$R = \text{icdf}(\sigma_{vm}) \cdot (\hat{r}\rho - \hat{r}) + \hat{r} \tag{5}$$

where $\hat{r}$ the size of the largest radius (determined as a fraction of the objects bounding box diagonal length) and $\rho \in (0, 1]$ gives the ratio of smallest to largest radii. As the von Mises stress has an arbitrary range from smallest to largest values with spatially varying concentrated extremes, we normalize and homogenize the field using an *inverse cumulative distribution function* (icdf).

Restricted Delaunay/Voronoi: The Delaunay complex or Voronoi diagram resulting from the generated samples are not natively limited to the design domain, i.e., the object's outer boundary. As S includes vertices from the object's hull as samples, the Delaunay complex is restricted to the object's shape by simply excluding Delaunay simplices outside of the object using robust winding numbers [44]. Due to the dual nature of the Voronoi diagram and the Delaunay complex, the Voronoi cells of such hull vertex samples always transcend the object's outer boundary. Therefore, Voronoi cells crossing the outer hull are cut and clipped [45], [46] such that only their inner part remains, cells fully outside are omitted.
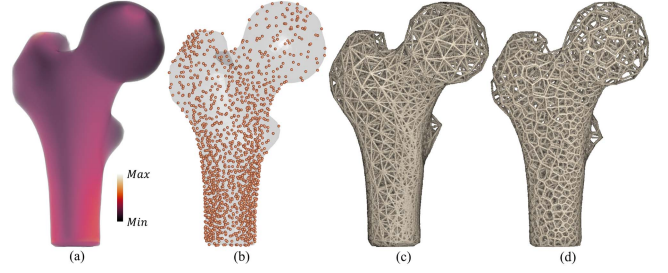

Fig. 6. Stress-aware Voronoi infill. (a) Scalar von Mises stress field. (b) Point cloud with stress-aware density. (c) Tetrahedral Delaunay mesh from (b). (d) Voronoi infill from (c).

(a) Original PSLs (b) Conforming lattice structure (c) Smoothed PSLs (d) Volumetric Michell trusses
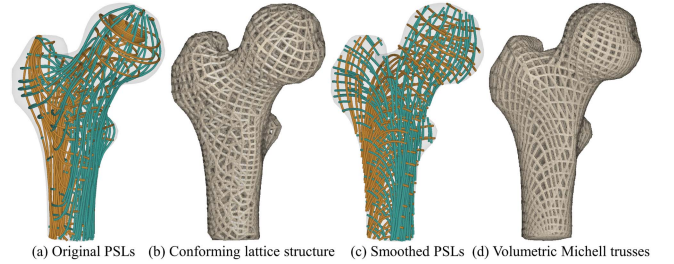
Fig. 7. Major (brown) and minor (green) PSLs in the solid object under load (a) and in the smoothed stress field (c), used to generate the designs in (b) and (d), respectively.

The edges of a Voronoi infill do not follow the stress directions in the initial solid object. Whereas the Delaunay criterion guarantees the most regular simplices when applied on the available sampling points, there is no trivial control for edge directions in the Voronoi graph, for instance, to construct Voronoi infills with controlled elasticity [42] or constraint alignment of the Voronoi edges [47].

SGLDBench provides Voronoi infill generation via Python due to the easy accessibility of required functionality. The SciPy [48] library is used to generate the Voronoi and Delaunay graphs using Qhull [49] and further provides the kd-tree acceleration structure for the Poisson disk sampling. Our code includes fallback methods required for restricting the graph structure to the object domain. The pipeline is illustrated in Fig. 6.

4) PSL-Guided Lattice Infill: In the seminal work by Michell [5] on stiffness-optimal lightweight design, it was conjectured that a stiffness-optimal structure should bear only normal stresses. This means that the sub-structures of such a design align with the principal stress directions. This is known as *Michell's Theory*, which has been considered since then in various lightweight design methods.

The most straightforward approach to create an infill that considers the principal stress directions in the loaded solid domain is to deposit material along the PSLs. When using line seeding strategies to obtain an as uniform as possible and domain filling distribution of PSLs [33], [50], PSL-guided infills show surprisingly good mechanical properties in 2D domains [51]. In 3D domains, however, many PSLs do not significantly contribute to the infill's stiffness, and PSLs might travel through space over a long distance before they intersect with any other PSL or attach to the boundary (see Fig. 7(a)).

On the other hand, when depositing material by voxelizing lines with a selected thickness, the thicker the PSLs are, the more connections are generated. This increases significantly the mechanical performance, and it often results in infills that show superior mechanical properties. In addition, since the stress field needs to be computed only once in the solid domain, the computational complexity is significantly reduced.

SGLDBench uses the publically available MATLAB backend of 3D-TSV [36] to generate PSLs in a 3D stress tensor field. It generates a domain-filling and evenly-spaced set of PSLs. The method starts from a set of domain-filling seed points and iteratively creates PSL from these seeds. All remaining seeds in a certain distance to the PSL are removed to control the sparseness of the resulting PSL distribution. The thickness of PSLs is selected by the user, and for a selected thickness the density of seeded PSLs is iteratively increased until the given volume budget is roughly reached. A PSL might enter into a region where the three principal stress directions are not uniquely defined and exchange their orientation, i.e., around so-called degenerate points [10]. Tracing a PSL stops if the resulting directional change exceeds a given limit, and the PSL is removed to avoid wasting material.

5) Conforming Lattice Structure: Conforming lattice structure originates from the geometry-based structural dehomogenization presented by Wu et al. [6]. Structural dehomogenization [52], [53], [54] diverges from the fine-resolution simulation and optimization used by density-based TO and porous infill optimization by adopting a multi-scale strategy:

Homogenization-based TO: During the initial optimization, a coarse-scale representation is tuned to approach the optimal distribution of material across a structure. This structure doesn't represent the exact material layout but rather provides a set of specifications to guide the optimal material layout.

Dehomogenization: Once the optimized specifications are found, dehomogenization is the process of converting coarse, homogenized results into detailed, fine-scaled structures. This involves creating actual geometric elements (trusses, lattices, microstructures) realizing the material properties and orientations suggested by the homogenized result.

Homogenization-based TO provides an orthotropic direction field as the optimized specifications. Dehomogenization extracts a conforming lattice structure with edges aligning with the direction field, and aspect ratios or sizes conveying the associated properties of the corresponding directions. The optimal directions are given by the principal stress directions of the homogenization-based structure layout.

SGLDBench directly feds the stress field in the solid domain to the dehomogenization stage, where it is used to generate a stress-aligned lattice structure. Fig. 7(a) and (b) show the initial stress directions in the solid domain and the conforming lattice structure where the cell sizes have been further adapted to the local von Mises stresses.

The conforming lattice structure addresses the intersection issue found in PSL-guided infills by relaxing the requirement to strictly follow the principal stress directions. This approach builds upon the field-aligned hex-dominant meshing method by Gao et al. [34], which employs an orthogonal frame field to align the edges of a hexahedral mesh with this field. In the conforming lattice structure, the frame field is replaced with the field of principal stress directions, and mechanical anisotropy is incorporated into the field-aligned parameterization process.

Due to the presence of degenerate points and regions in a stress field, it is generally impossible to compute a conforming hexahedral mesh for all but the simplest fields. The conforming lattice structure addresses this limitation by employing a local smoothing strategy. The key idea is to leverage the rotational symmetry of principal stress directions to generate a smoothed direction field, which enables the construction of a conforming hexahedral mesh.

From this smoothed direction field, a position field is computed, ensuring that its gradient aligns with the adjusted stress directions. By combining the smoothed direction field and the position field, the method constructs a graph structure and subsequently extracts the final mesh. To preserve the divergence and convergence properties of the underlying stress field, irregular vertex connections are introduced in the final structure, resulting in an edge-graph representation.

6) Volumetric Michell Trusses: Volumetric Michell Trusses compute a stress-aligned hexahedral lattice using a parametric approach to align truss elements with the principal directions of the stress field. This method involves two key steps:

Frame Field Smoothing: The algorithm first applies an FEA to compute the stress tensor field, followed by a frame field generation aligned with the principal stress directions. However, to achieve a global parametric structure, this step smooths out tensor field singularities, sacrificing local alignment near degenerate points for overall global consistency. To address this, the method uses Loubignac iterations [55] to smooth out the discontinuous stress field. This iterative method adjusts the stress field to ensure that it becomes continuous across element boundaries, thereby allowing for smoother and more uniform alignment in subsequent steps. Therefore, a smoothness energy function that penalizes sharp changes in frame directions is minimized. This optimization enables a globally smooth frame field that approximates the original stress directions.

Tracing Integer Isolines: After smoothing, integer isolines of the volumetric texture parameterization are traced. This step maps the truss nodes to integer points of the parameterization, yielding the geometry for the extracted graph structure. Its connectivity follows from the nodes' adjacency in the grid. To ensure flexibility, the method allows scaling of the parameterization via a user-defined resolution parameter ρ, which controls the density of the truss structure.

Due to the applied smoothing of the initial stress field, volumetric Michell trusses produce a regular hexahedral lattice structure with improved continuity of load transmissions, as demonstrated in Fig. 7(c), Fig. (d). However, the smoothing process can substantially alter the initial stress field, leading to significant deviations in the resulting design's stiffness from the optimal value.

D. Infill Voxelization

While TO and porous infill compute a binary material field on a 3D voxel grid, the other methods compute a 3D lattice structure composed of edges and nodes. To enable a
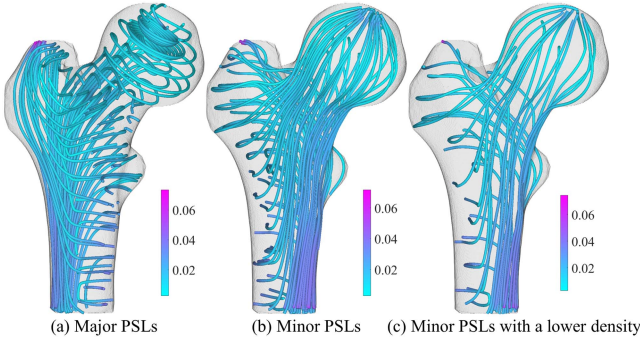
Fig. 8. PSL visualization with cylindrical elements and color-coded von Mises stresses using MATLAB visualization programs.

meaningful comparison of the structural properties of all approaches, SGLDBench voxelizes these structures into a voxel grid. The grid resolution is selected automatically to represent edges with a minimum required voxels. SGLDBench computes for each edge the intersected voxels via the DDA line drawing algorithm [56]. These voxels are set to solid. Edges are thickened by setting for all these voxels their 26 neighboring voxels to solid.

For all edge-based infill strategies, the material budget and the targeted edge thickness are prescribed. The methods are then conducted in a dichotomy manner to find the settings that match closely the design specifications, i.e., the design process is run multiple times to find the design that matches the material consumption under the edge thickness constraint.

E. Visualization and Layout Analysis

Once a Voronoi infill, PSL-guided infill, conforming lattice structures, or volumetric Michell trusses has been computed, users can view the meshes and graph structures using MATLAB's mesh viewing operations, rendering edges as cylinders with a specified width. Thus, voxelizing the infill into a 3D material field is not required.

To compute the compliance of a design, lattice infill designs must first be voxelized into a 3D material field. SGLDBench then uses its linear elasticity solver to perform an FEA and simulate the stress field, from which the compliance is computed. For density-based TO and porous infill optimization, the compliance history throughout the optimization process is recorded and can be visualized via a curve plot, as demonstrated in Fig. 5.

To visualize a stress field, an evenly spaced set of PSLs covering the domain as uniformly as possible is computed using MATLAB programs. Users can control the density of seeded PSLs and select scalar stress measures, such as the von Mises norm, to map onto the lines' colors. PSLs can be computed for the initial stress field in the solid domain. Examples of PSL-guided visualizations using MATLAB are shown in Fig. 8.

For realtime visualization of even high resolution designs, SGLDBench provides an advanced WebGL-based volume visualization module. It performs isosurface and direct volume rendering, and assists users in a stress-based comparative design analysis. Isosurfaces in a 3D material field are rendered using

GPU ray-casting, with screen-space ambient occlusion to enhance depth perception, as demonstrated in Fig. 1. A view-space parallel clip plane can be moved back and forth to expose otherwise occluded structures.

Direct volume rendering is particularly used to visualize the scalar von Mises stress field, enabling the evaluation of whether a material will permanently deform under the given stress state. High magnitudes of the von Mises stress indicate a risk of fracture under the applied loads, and direct volume rendering effectively highlights the spatial regions where this danger is significant. This provides a powerful tool for structural and mechanical analysis. SGLDBench's WebGL interface allows users to color a ray-traced infill surface based on the von Mises stress.

In addition, SGLDBench's visualization module highlights the differences between the principal stress directions in the solid design and the computed infill design. SGLDBench computes the stress field of the infill using the initial boundary conditions and generates an auxiliary grid where each voxel stores a single deviation measure. This measure indicates the deviation of the stress directions corresponding to the principal stresses with the maximum absolute value, based on the ordering of the absolute values of the principal stresses in the initial solid and the infill.

The resulting scalar field is visualized through direct volume rendering. SGLDBench employs a transfer function that maps directional deviations linearly to colors, ranging from white (low deviation) to light brown (medium deviation) and dark brown (high deviation). Opacity is initially set to one but can be adjusted by the user to smoothly fade out regions with low or high deviation.

Given an optimized infill structure, SGLDBench can also be used to apply new boundary conditions to it, allowing users to probe conditions different from those for which the structure was initially optimized. This functionality is inspired by worst-case structural analysis [57], [58], a method used to evaluate the performance and reliability of a structure under its most unfavorable conditions. While SGLDBench is designed for a completely different use case and cannot perform such analysis, it provides the tools to explore similar scenarios.

Specifically, SGLDBench includes an interface to modify the initial boundary conditions by changing the direction of forces and re-computing the compliance and von Mises stress under the new conditions. A visualization of the structure with stress-based color coding highlights the mechanical strengths and weaknesses of different parts, enabling a deeper understanding of its behavior under varied conditions.

IV. EXPERIMENTS

We demonstrate the use of SGLDBench to generate and analyze lightweight designs for various models and boundary conditions. The models include a human femur (*Bone*), a machine part commonly seen in engineering applications (*Part*), and *Cantilever*, a widely used benchmark model in TO. All models are initially provided as triangle meshes. We showcase the usability of SGLDBench with additional datasets in the supplementary material.

(a) Density-based TO,
$c = 1.02, V = 0.10$

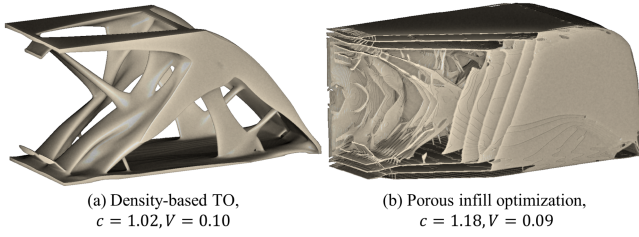(b) Porous infill optimization,
$c = 1.18, V = 0.09$

Fig. 9. Material fields generated with TO (a) and porous infill optimization (b) for *Cantilever* using a $800 \times 400 \times 400$ voxel grid. The left face of the cubic design domain is fixed, and a downward force acts along the bottom-right edge.

All experiments are conducted on a desktop computer equipped with an Intel 6-core Xeon W2235 CPU, 64 GB of RAM, and an NVIDIA RTX 2070 GPU with 8 GB of video memory. We intentionally select a mid-range architecture to demonstrate SGLDBench's capabilities on affordable hardware. The main memory limits the maximum number of simulation elements to approximately 160 million for simulating a 3D stress field.

A. Performance Evaluation

Solver: For the *Cantilever* model at a voxel grid resolution of $860 \times 430 \times 430$ (159 million elements), SGLDBench solves the FEA linear system on the CPU in roughly 30 minutes, achieving convergence within 41 solver iterations at a threshold of 1.0×10^{-3}

A one-to-one CUDA implementation of the solver by Wu et al. [27] on the RTX 2070 GPU can simulate up to 45 million elements before running out of GPU memory. For this number of elements, the GPU implementation needs 91 seconds to solve the FEA linear system. SGLDBench requires 610 seconds for the same setting, showing roughly a 7x reduction of the performance compared to the optimized GPU solver. For the used GPU this is a reasonable reduction, and slightly better than commonly reported when GPU and CPU implementations of similar problems are compared.

Iterative Optimization: TO and porous infill optimization require additional memory for updating the material distribution during numerical optimization. SGLDBench performs these optimizations with grids of up to 130 million simulation elements, corresponding to a $800 \times 400 \times 400$ simulation grid with 386 million degrees of freedom. For these cases, SGLDBench completes each optimization iteration in approximately 45 minutes for TO and 67 minutes for porous infill optimization. The results shown in Fig. 9 are generated using 30 iterations of TO and 280 iterations of porous infill optimization.

Compared to an optimized OpenMP CPU implementation of TO [38], SGLDBench is only about 1.6 times slower when reproducing the same *Cantilever* model at a resolution of $640 \times 320 \times 320$ on a similar computing architecture (a 48-core Xeon CPU). This demonstrates the efficiency of the MATLAB computing environment in combination with SGLDBench's element-wise computation structure.

All designs for *Bone* in Fig. 1 are generated with a $384 \times 256 \times 512$ voxel grid, corresponding to 30 million degrees of

freedom. TO and porous infill optimization, respectively, generate the infill in 1.5 hours and 15.1 hours. Porous infill requires a significantly higher number of optimization iterations and additional computations to enforce the local volume constraint. Due to the high geometric complexity of porous infills, generating these infill also requires a higher number of iterations for solving the linear FEA system in each optimization iteration.

Lattice Infill Optimization: To compute the various types of lattice infills, we utilize state-of-the-art implementations currently available, which are provided as either C/C++ codes (PSL-guided layouts, Voronoi, and conforming lattice infills) or MATLAB programs (Michell trusses). Executing these implementations through SGLDBench does not result in performance penalties.

SGLDBench requires approximately 10 minutes to generate the Voronoi infill and the PSL-guided infill, with about half of this time spent simulating the 3D stress field in the initial solid domain. Using this stress field, the external codes for generating the conforming lattice infill take around 5 minutes, while the MATLAB code for generating the volumetric Michell truss requires approximately 4.5 hours. For the latter, slightly less than half of the execution time is allocated to smoothing the original stress field.

B. Material Use as a Modelling Parameter

To evaluate the response of different design methods to changes in material consumption, we repeat all experiments shown in Fig. 1 with a reduced volume fraction. Instead of the original volume fraction of 0.4, SGLDBench now uses a lower volume fraction of approximately 0.2, enforcing finer and less dense support structures. For methods that generate graph structures, the thickness of the voxelized edges remains constant.

The results, shown in Fig. 10, reveal an increased sparseness in all designs, accompanied by reduced stiffness and significantly varied topologies. TO and porous infill methods require no changes to their simulation parameters but must rerun the entire optimization process to produce the results. In contrast, other approaches can utilize the stress field from the initial solid domain and need to rerun only the steps that generate the graph structure from it.

With a lower material budget, the PSL-guided infill demonstrates surprisingly good relative performance, as it effectively utilizes the material to generate support structures along the most dominant stress directions.

One possible reason for this reversal is that the resolution of the conforming lattice is too low, leading to the misalignment of many edges in the graph structure with the dominant stress directions. Furthermore, adaptive porosity also plays a critical role in reducing compliance. Specifically, in regions of high stress, more material should be allocated to resist strain effectively. This could explain the relatively moderate stiffness of the Voronoi infill, which is unable to concentrate more material in high-stress regions due to the low material budget. Moreover, by design, the Voronoi infill does not align with the dominant stress directions, further contributing to its reduced mechanical performance.
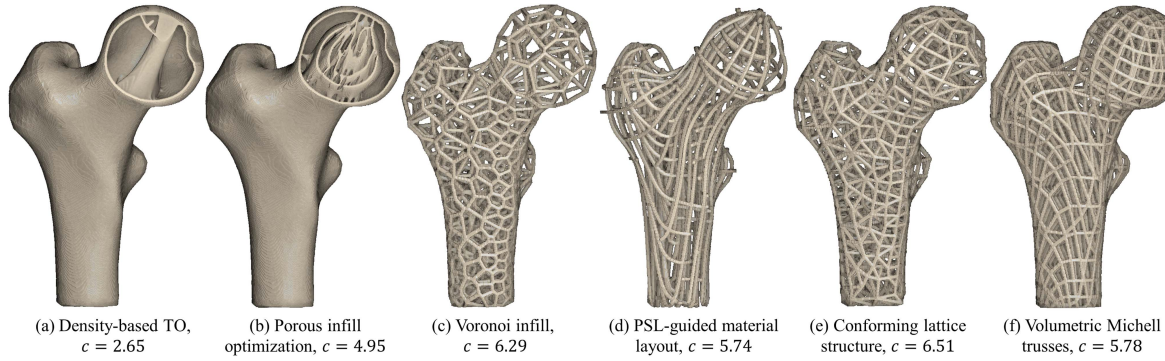
(a) Density-based TO, $c = 2.65$ (b) Porous infill optimization, $c = 4.95$ (c) Voronoi infill, $c = 6.29$ (d) PSL-guided material layout, $c = 5.74$ (e) Conforming lattice structure, $c = 6.51$ (f) Volumetric Michell trusses, $c = 5.78$

Fig. 10.   Same designs as in Fig. 1 but with lower material budget of roughly 0.22.

(a) Conforming lattice structure, $c = 3.38$ (b) Voronoi infill, $c = 2.68$
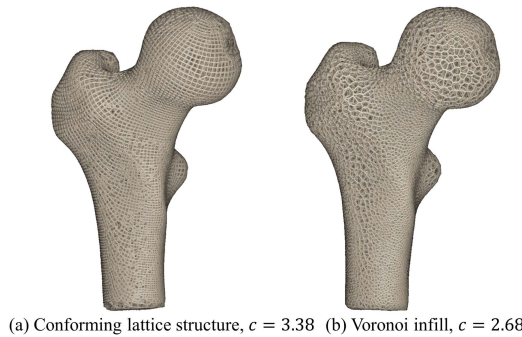
Fig. 11.   High resolution results using the same boundary conditions and material budget as in Fig. 1.

(a) PSL-guided material layout   (b) Conforming lattice structure   (c) Volumetric Michell structures
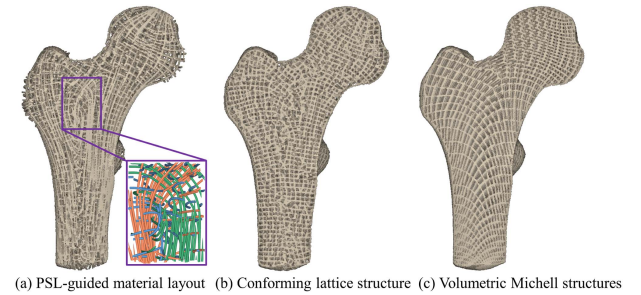
Fig. 12.   A clip plane reveals the interior structure of different designs. The closeup view shows the PSLs in the selected region, with the major, medium, and minor PSLs, respectively, shown in ocher, green, and blue.

The stress field in *Bone* is relatively simple and contains few degeneracies, allowing the Michell trusses to preserve most features effectively. This behavior, however, changes with *Part* (see the first row of Fig. 13). In this case, the smoothed stress field exhibits significant differences from the initial field, causing many infill edges to deviate substantially from the original stress directions. As a result, the Michell trusses demonstrate lower stiffness compared to the other alternatives.

C. Infill Resolution and Compliance

To explore the relationship between compliance, volume fraction, and geometric infill resolution, SGLDBench is used to evaluate the stiffness of high-resolution lattice infills for *Bone*. The volume fraction, edge thickness, and voxel grid resolution are user-defined parameters, and SGLDBench computes the compliance of the infill using a single FEA iteration.

In Fig. 11, a voxelized conforming lattice infill and Voronoi infill with 136,877 and 396,458 edges, respectively, are shown, using the same boundary conditions as in Figs 1 and 10. SGLDBench generates a $848 \times 576 \times 1200$ voxel grid and constructs a finite element model from it. The voxelized versions of the infills are visualized using SGLDBench's WebGL visualization module, employing isosurface ray-casting and ambient occlusion for enhanced depth perception.

To compute the compliance of the voxel models, approximately 380 million degrees of freedom are solved, which SGLDBench completes in roughly 45 minutes. Interestingly, as shown

in Fig. 1, the stiffness appears to be independent of the geometric details of the infills when using the same material budget.

D. Infill Geometry

Given the varying compliance of different types of infills for the same object and load conditions, we use SGLDBench to investigate the major causes of these differences. We compare the interior structures of graph-based infillS, as illustrated with *Bone* in Fig. 12. A clip plane is used to expose the interior regions of the PSL-guided infill, the conforming lattice structure, and the volumetric Michell trusses.

The PSL-guided infill serves as a reference, illustrating the primary load transmission pathways. Overall, the edges of the conforming lattice structure align with the PSLs. However, in regions where the curvature of the PSLs changes significantly (see the inset in Fig. 12), the lattice loses its geometric regularity. These irregularities appear to be associated with degeneracies in the stress field, where the principal stress directions flip.

In contrast, the volumetric Michell trusses exhibit a highly regular geometric structure but demonstrate less alignment with the original stress directions. This discrepancy arises because the original stress field is smoothed, which alters the stress field globally and particularly eliminates unwanted degeneracies.

The visualization of the 3D conforming lattice structure reveals several stitching edges in the interior that do not align with any of the dominant stress directions. Additionally, the
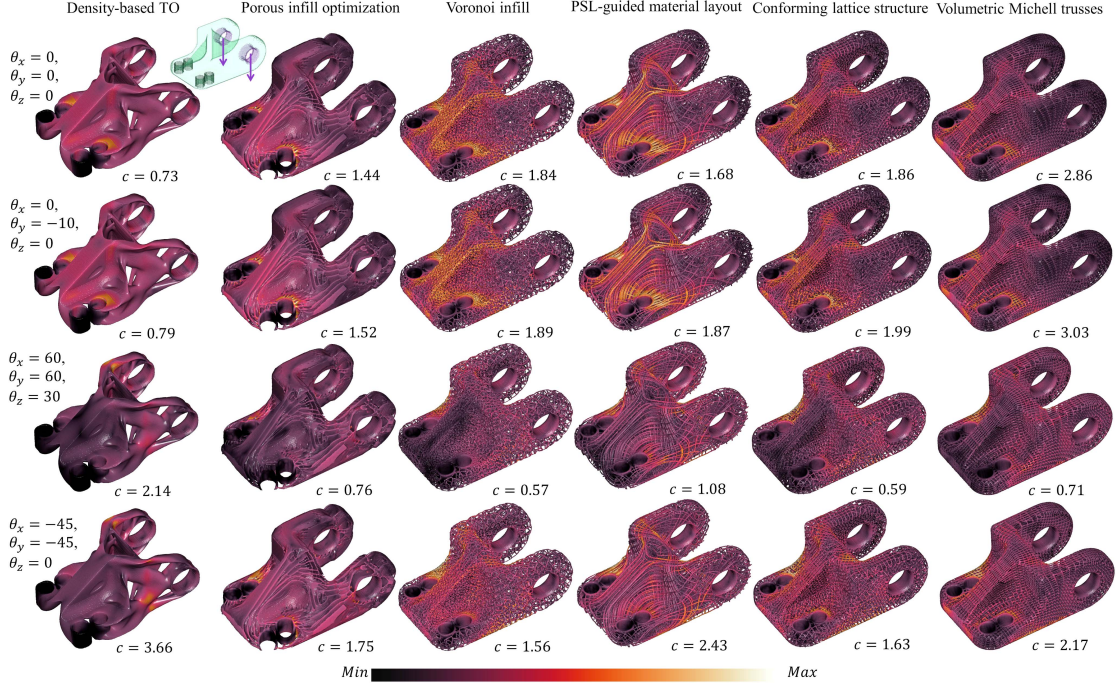
Fig. 13. Variable load structural analysis. Top row: All infills are optimized for the same boundary conditions (see inset with downward forces). Rows 2–4: Varying forces (Euler angles $\theta_x, \theta_y, \theta_z$ indicate angular deviation from the downward direction) are applied to the optimized infills. All designs are shown with their compliance c and are color-coded according to the von Mises stress.
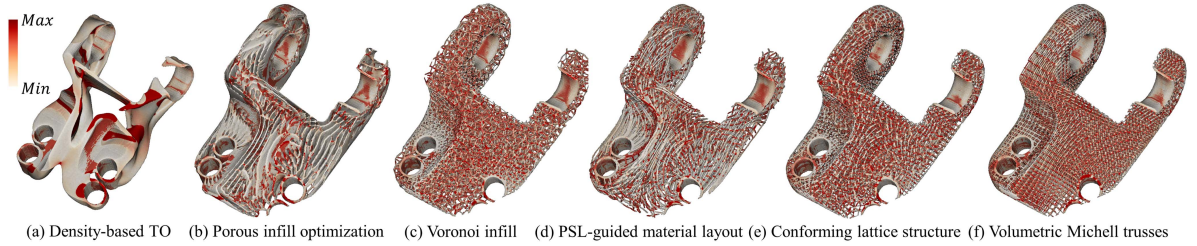

Fig. 14. Stress-to-stress alignment. High (white) to low (brown) alignment between stresses in the solid and the optimized infill with respect to the same boundary conditions is shown. A view plane aligned clip plane reveals interior parts.

number of longer edge sequences that consistently follow one of the PSLs is lower compared to the PSL-guided infill and the volumetric Michell trusses. This behavior might be attributed to the approach allocating too much material to maintain a consistent edge graph rather than prioritizing stiffness. Oscillating load paths, represented by inconsistent edge sequences, may lead to less effective load transitions and, consequently, reduced stiffness.

E. Variable Load Structural Analysis

To shed light on how well a design optimized for a specific load case can resist forces applied from a different direction, SGLDBench is used in the following way: First, all six infill design methods are applied with the same boundary condition to compute six different designs, and the von Mises stress field in the optimized material field is computed. Then, the applied forces are changed, and the von Mises stress field is recomputed

for all designs. The designs, color-coded with the von Mises stress under the varied forces, are shown in Fig. 13.

Density-based TO consistently shows higher compliance under the new force conditions. While the compliance increases only slightly with a slight change of the force direction, larger directional changes result in a significant loss in stiffness. The coloring with the von Mises stress shows where the critical structures occur under the new forces and have the potential to break. A significant re-distribution of the von Mises stresses indicates a significant loss in stiffness.

All other designs show the same slight change in stiffness as porous infill when the force direction changes only slightly. For more significant changes in the force directions, however, all other designs perform significantly better than the one optimized with density-based TO. Interestingly, even the PSL-guided material layout, which is specifically aligned with the major stress directions occurring with the initial force setting, can significantly better resist the new force directions.

Designs that show a more uniform distribution of material throughout the domain, such as the Voronoi infill, the conforming lattice structures, and the volumetric Michell trusses, are much better at resisting varying forces compared to topology-optimized designs. When the force directions are varied significantly from their initial values, in some cases the stiffness becomes even higher. These are situations where the new forces are along some of the principal stress directions that have occurred in the initial stress field, and along which some of the structures have grown, so that the structure is bearing more normal stresses than shear stresses.

F. Reproducing Stresses in the Solid Domain

Using *Part* shown in Fig. 13, SGLDBench is applied to analyze how well different designs reproduce the stress directions in the initial solid configuration. This analysis allows users to examine, for instance, whether force transmission through tension and compression occurs along the same load paths as in the initial solid. Since an infill design can, in principle, transmit forces along the most efficient load paths in the solid design or diverge to alternative paths, understanding the relationship between compliance and the reproduction of stress directions can provide valuable insights for improving infill designs.

Fig. 14 shows visualizations of the alignment fields computed for each infill. Iso-surface rendering with the color transfer function described in Section III-D is used to emphasize regions with high deviation in brown. TO produces an infill that, in many regions, aligns well with the major stress directions in the solid under load. However, significant deviations are observed in certain areas, particularly in thin structures and regions near the fixed elements where stress concentrations are highest.

Similarly, porous infill exhibits more pronounced deviations, which are distributed throughout the entire domain. This behavior arises from its space-filling material distribution, which emphasizes uniformity rather than alignment with the stress directions.

The stress deviations are highest in the Voronoi infill, as the initial stress directions are not considered in its edge-graph layout. Only when edges align with boundary elements, which are incorporated into the stress analysis of both the solid object and the infill, stress deviations are lower.

The material in the PSL-guided infill is distributed along the principal stress directions, resulting in good alignment with the initial stress field. However, as shown in Fig. 14, alignment accuracy decreases at the boundaries of the voxelized edges. This reduction in accuracy stems from discretization-induced inaccuracies in the stress simulation—a limitation shared by all graph-based approaches when voxelized structures are used for compliance analysis.

Additionally, all graph-based infills contain numerous edges aligned with the two principal stress directions not corresponding to the direction of the maximum absolute stress value. This can lead to high directional deviations along these edges.

The deviation maps of the conforming lattice structure and the Michell trusses are very similar, showing the same low deviation patterns in the boundary region as the Voronoi infill. In both infills, there is still a considerable number of voxels with significantly different dominant stress directions than the solid object.

V. Conclusion and Outlook

SGLDBench is a benchmark suite designed for the simulation and analysis of stiff, lightweight designs, with a special emphasis on high-resolution 3D models. It supports the computation of six distinct design types, employing methods that range from purely stiffness-based optimization to geometry-aware infill generation.

SGLDBench enables users to create designs at varying resolutions and material consumption levels while assessing their mechanical and geometric properties. Various visualization options provide additional insights into governing stress states and a design's geometric structure. The resistance of a design to new force situations can be assessed via the color coding of designs with scalar stress measures. Additionally, SGLDBench visualizes deviations between stress directions in the initial solid object and the generated design, providing insights into relationships between stiffness, geometric properties, and stress replication.

The suite allows users to compute individual designs with specific boundary conditions and offers flexibility for integrating new design strategies into its publicly available code base. Moreover, SGLDBench supports the creation of novel design types by combining existing strategies. For instance, the material field of an optimized design can be computed via TO and downloaded to extract its boundary as an isosurface. This surface can then be uploaded to SGLDBench to compute an infill restricted to the interior of the surface, as demonstrated in the supplemental material.

By leveraging SGLDBench's capabilities, several intriguing properties of existing design strategies have been uncovered, paving the way for new research directions in lightweight design. For example, why does the mechanical performance of conforming lattice structures fall below expectations in 3D domains, what role does adaptive porosity play in achieving high stiffness, what is the interplay between truss thickness, cell size, and stiffness in lattice infills, how can a tensor field be optimally smoothed to minimize stiffness deviation while maintaining efficient load transmission paths, or how would a Voronoi infill perform if additional constraints on edge stress reproduction were applied? SGLDBench provides researchers with tools to investigate these questions and develop improved design strategies tailored to diverse objectives.

acknowledge the generosity of the authors in making their code publicly available.

REFERENCES

[1] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Struct. Optim.*, vol. 1, no. 4, pp. 193–202, 1989.

[2] G. Allaire, F. Jouve, and A.-M. Toader, "Structural optimization using sensitivity analysis and a level-set method," *J. Comput. Phys.*, vol. 194, no. 1, pp. 363–393, 2004.

[3] J. Wu, O. Sigmund, and J. P. Groen, "Topology optimization of multi-scale structures: A review," *Struct. Multidisciplinary Optim.*, vol. 63, pp. 1455–1480, 2021.

[4] P. Pedersen, "On optimal orientation of orthotropic materials," *Struct. Optim.*, vol. 1, no. 2, pp. 101–106, 1989.

[5] A. G. M. Michell, "Lviii, the limits of economy of material in frame-structures," *London, Edinburgh, Dublin Philos. Mag. J. Sci.*, vol. 8, no. 47, pp. 589–597, 1904.

[6] J. Wu, W. Wang, and X. Gao, "Design and optimization of conforming lattice structures," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 1, pp. 43–56, Jan. 2021.

[7] J. Plocher and A. Panesar, "Review on design and structural optimisation in additive manufacturing: Towards next-generation lightweight structures," *Mater. Des.*, vol. 183, 2019, Art. no. 108164.

[8] F. Gil-Ureta, N. Pietroni, and D. Zorin, "Reinforcement of general shell structures," *ACM Trans. Graph.*, vol. 39, no. 5, pp. 1–19, 2020.

[9] A. Bacciaglia, A. Ceruti, and A. Liverani, "A systematic review of voxelization method in additive manufacturing," *Mechan. Ind.*, vol. 20, no. 6, 2019, Art. no. 630.

[10] T. Delmarcelle and L. Hesselink, "The topology of symmetric, second-order tensor fields," in *Proc. IEEE Vis.*, 1994, pp. 140–147.

[11] J. Wang, J. Wu, and R. Westermann, "A globally conforming lattice structure for 2D stress tensor visualization," *Comput. Graph. Forum*, vol. 39, no. 3, pp. 417–427, 2020.

[12] C. Hergl et al., "Visualization of tensor fields in mechanics," in *Computer Graphics Forum*, vol. 40. Hoboken, NJ, USA: Wiley, 2021, pp. 135–161.

[13] S.-H. Hung, Y. Zhang, and E. Zhang, "Global topology of 3D symmetric tensor fields," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 1, pp. 1282–1291, Jan. 2024.

[14] F. C. Stutz, J. P. Groen, O. Sigmund, and J. A. Bærentzen, "Singularity aware de-homogenization for high-resolution topology optimized structures," *Struct. Multidisciplinary Optim.*, vol. 62, pp. 2279–2295, 2020.

[15] S. Rojas-Labanda and M. Stolpe, "Benchmarking optimization solvers for structural topology optimization," *Struct. Multidisciplinary Optim.*, vol. 52, no. 3, pp. 527–547, 2015.

[16] S. I. Valdez, S. Botello, M. A. Ochoa, J. L. Marroquín, and V. Cardoso, "Topology optimization benchmarks in 2D: Results for minimum compliance and minimum volume in planar stress problems," *Arch. Comput. Methods Eng.*, vol. 24, pp. 803–839, 2017.

[17] O. Sigmund, "On benchmarking and good scientific practise in topology optimization," *Struct. Multidisciplinary Optim.*, vol. 65, no. 11, 2022, Art. no. 315.

[18] B. Pernet, J. K. Nagel, and H. Zhang, "Compressive strength assessment of 3D printing infill patterns," *Procedia CIRP*, vol. 105, pp. 682–687, 2022, Leuven, Belgium. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827122001159

[19] J. R. Martins and A. B. Lambe, "Multidisciplinary design optimization: A survey of architectures," *AIAA J.*, vol. 51, no. 9, pp. 2049–2075, 2013.

[20] F. Tamburrino, S. Graziosi, and M. Bordegoni, "The design process of additively manufactured mesoscale lattice structures: A review," *J. Comput. Inf. Sci. Eng.*, vol. 18, no. 4, 2018, Art. no. 040801.

[21] C. Pan, Y. Han, and J. Lu, "Design and optimization of lattice structures: A review," *Appl. Sci.*, vol. 10, no. 18, 2020, Art. no. 6374.

[22] A. Seharing, A. H. Azman, and S. Abdullah, "A review on integration of lightweight gradient lattice structures in additive manufacturing parts," in *Proc. Adv. Mech. Eng.*, 2020, Art. no. 1687814020916951.

[23] T. Maconachie et al., "SLM lattice structures: Properties, performance, applications and challenges," *Mater. Des.*, vol. 183, 2019, Art. no. 108137.

[24] C. Zhang et al., "A novel lattice structure topology optimization method with extreme anisotropic lattice properties," *J. Comput. Des. Eng.*, vol. 8, no. 5, pp. 1367–1390, 2021.

[25] B. Zhu, M. Skouras, D. Chen, and W. Matusik, "Two-scale topology optimization with microstructures," *ACM Trans. Graph.*, vol. 36, no. 4, 2017, Art. no. 1.

[26] C. Dick, J. Georgii, and R. Westermann, "A real-time multigrid finite hexahedra method for elasticity simulation using cuda," *Simul. Modelling Pract. Theory*, vol. 19, no. 2, pp. 801–816, 2011.

[27] J. Wu, C. Dick, and R. Westermann, "A system for high-resolution topology optimization," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 3, pp. 1195–1208, Mar. 2016.

[28] H. Liu, Y. Hu, B. Zhu, W. Matusik, and E. Sifakis, "Narrow-band topology optimization on a sparsely populated grid," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–14, 2018.

[29] N. Aage, E. Andreassen, B. S. Lazarov, and O. Sigmund, "Giga-voxel computational morphogenesis for structural design," *Nature*, vol. 550, no. 7674, pp. 84–86, 2017.

[30] O. Sigmund, "A 99 line topology optimization code written in Matlab," *Struct. Multidisciplinary Optim.*, vol. 21, no. 2, pp. 120–127, 2001.

[31] J. Wu, N. Aage, R. Westermann, and O. Sigmund, "Infill optimization for additive manufacturing–approaching bone-like porous structures," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 2, pp. 1127–1140, Feb. 2018.

[32] L. Lu et al., "Build-to-last: Strength to weight 3D printed objects," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–10, 2014.

[33] J. Wang, J. Wu, and R. Westermann, "Stress trajectory guided structural design and topology optimization," in *Proc. Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, 2022, Art. no. 1.

[34] X. Gao, W. Jakob, M. Tarini, and D. Panozzo, "Robust hex-dominant mesh generation using field-guided polyhedral agglomeration," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, 2017.

[35] R. Arora et al., "Volumetric michell trusses for parametric design & fabrication," in *Proc. ACM Symp. Comput. Fabr.*, 2019, pp. 1–13.

[36] J. Wang, C. Neuhauser, J. Wu, X. Gao, and R. Westermann, "3D-TSV: The 3D trajectory-based stress visualizer," in *Proc. Adv. Eng. Softw.*, 2022, Art. no. 103144.

[37] A. H. Aitkenhead, "Mesh voxelisation," 2013. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/27390-mesh-voxelisation

[38] E. A. Träff, A. Rydahl, S. Karlsson, O. Sigmund, and N. Aage, "Simple and efficient GPU accelerated topology optimisation: Codes and applications," *Comput. Methods Appl. Mechan. Eng.*, vol. 410, 2023, Art. no. 116043.

[39] K. Liu and A. Tovar, "An efficient 3D topology optimization code written in matlab," *Struct. Multidisciplinary Optim.*, vol. 50, pp. 1175–1196, 2014.

[40] F. Wang, B. S. Lazarov, and O. Sigmund, "On projection methods, convergence and robust formulations in topology optimization," *Struct. Multidisciplinary Optim.*, vol. 43, no. 6, pp. 767–784, 2011.

[41] K. Svanberg, "The method of moving asymptotes—a new method for structural optimization," *Int. J. Numer. Methods Eng.*, vol. 24, no. 2, pp. 359–373, 1987.

[42] J. Martínez, J. Dumas, and S. Lefebvre, "Procedural voronoi foams for additive manufacturing," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–12, 2016.

[43] J. Martínez, H. Song, J. Dumas, and S. Lefebvre, "Orthotropic k-nearest foams for additive manufacturing," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–12, Jul. 2017, doi: 10.1145/3072959.3073638.

[44] A. Jacobson, L. Kavan, and O. Sorkine-Hornung, "Robust inside-outside segmentation using generalized winding numbers," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–12, 2013.

[45] I. E. Sutherland and G. W. Hodgman, "Reentrant polygon clipping," *Commun. ACM*, vol. 17, no. 1, pp. 32–42, 1974.

[46] H. Edelsbrunner and N. R. Shah, "Triangulating topological spaces," in *Proc. 10th Annu. Symp. Comput. Geometry*, 1994, pp. 285–292.

[47] J. Martínez, S. Hornus, H. Song, and S. Lefebvre, "Polyhedral voronoi diagrams for additive manufacturing," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–15, 2018.

[48] P. Virtanen et al., "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[49] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, 1996.

[50] T.-H. Kwok, Y. Li, and Y. Chen, "A structural topology design method based on principal stress line," *Comput.-Aided Des.*, vol. 80, pp. 19–31, 2016.

[51] J. Wang, "Trajectory-based stress tensor visualization and its application in structural design and optimization," PhD dissertation, Technische Universität München, 2023.

[52] O. Pantz and K. Trabelsi, "A post-treatment of the homogenization method for shape optimization," *SIAM J. Control Optim.*, vol. 47, no. 3, pp. 1380–1398, 2008.

[53] J. P. Groen and O. Sigmund, "Homogenization-based topology optimiza-tion for high-resolution manufacturable microstructures," *Int. J. Numer. Methods Eng.*, vol. 113, no. 8, pp. 1148–1163, 2018.

[54] G. Allaire, P. Geoffroy-Donders, and O. Pantz, "Topology optimization of modulated and oriented periodic microstructures by the homogenization method," *Comput. Math. Appl.*, vol. 78, no. 7, pp. 2197–2229, 2019.

[55] G. Loubignac, G. Cantin, and G. Touzot, "Continuous stress fields in finite element analysis," *AIAA J.*, vol. 15, no. 11, pp. 1645–1647, 1977.

[56] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," *Eurographics*, vol. 87, no. 3, pp. 3–10, 1987.

[57] Q. Zhou, J. Panetta, and D. Zorin, "Worst-case structural analysis," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 137–1, 2013.

[58] D. Zhang, X. Zhai, X.-M. Fu, H. Wang, and L. Liu, "Large-scale worst-case topology optimization," in *Computer Graphics Forum*, vol. 41. Hoboken, NJ, USA: Wiley, 2022, pp. 529–540.

[59] S. Hang, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw*, vol. 41, no. 2, 2015, Art. no. 11.

[60] Y. Hu, T. Schneider, B. Wang, D. Zorin, and D. Panozzo, "Fast tetrahedral meshing in the wild," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 117–1, 2020.

[61] N. Aage and B. S. Lazarov, "Parallel framework for topology optimization using the method of moving asymptotes," *Struct. Multidisciplinary Optim.*, vol. 47, no. 4, pp. 493–505, 2013.
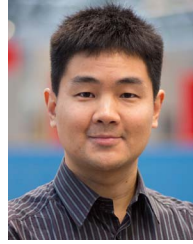
Simon Niedermayr received the master's degree in computer science from TUM, in 2022. He is currently working toward the PhD degree with the Computer Graphics and Visualization Group, Technical University of Munich (TUM). His research interests include differentiable rendering, real-time rendering and high performance GPGPU programming.

Christoph Neuhauser received the bachelor's and master's degrees in computer science from the Technical University of Munich (TUM), in 2019 and 2020, respectively, and the PhD degree from the Computer Graphics and Visualization Group, TUM, in 2025. He is a GPU Software Optimization Engineer with Intel. Major interests in research comprise real-time rendering, GPU computing and scientific visualization.

Junpeng Wang received the PhD degree from the same group, in 2023. He is a post-doc in the Computer Graphics and Visualization Group, Technical University of Munich, Germany. His research interests range from scientific visualization to computational design, with a special focus on lightweight structure design and optimization.

Jun Wu received the PhD degree in computer science from TUM, in 2015 and the PhD degree in mechanical engineering from Beihang University, in 2012, Beijing, China. He is an associate professor with the Department of Sustainable Design Engineering, Delft University of Technology. Before this, he was a Marie Curie postdoc fellow with the Department of Mechanical Engineering, Technical University of Denmark. His research is focused on computational design and digital fabrication, with an emphasis on topology optimization.

Dennis R. Bukenberger received the PhD degree from the University of Tübingen, in 2021. He is a postdoc in the Computer Graphics and Visualization Group, Technical University of Munich, Germany. Before this, he was a postdoc with the Technical University of Dortmund. His research interests include meshing algorithms, Voronoi diagrams, geometry processing & sustainable design engineering.

Rüdiger Westermann is a full professor for computer science in the TUM School of Computation, Information and Technology. He is the head of Computer Graphics and Visualization group. His research interests include scalable data visualization and simulation algorithms, GPU computing, TO, neural representations for computer graphics and visualization, as well as real-time and photorealistic rendering.