Narrow-Band Topology Optimization on a Sparsely Populated Grid

HAIXIANG LIU*, University of Wisconsin-Madison
YUANMING HU*, MIT CSAIL
BO ZHU, MIT CSAIL and Dartmouth College
WOJCIECH MATUSIK, MIT CSAIL
EFTYCHIOS SIFAKIS, University of Wisconsin-Madison

A variety of structures in nature exhibit sparse, thin, and intricate features. It is challenging to investigate these structural characteristics using conventional numerical approaches since such features require highly refined spatial resolution to capture and therefore they incur a prohibitively high computational cost. We present a novel computational framework for high-resolution topology optimization that delivers leaps in simulation capabilities, by two orders of magnitude, from the state-of-the-art approaches. Our technique accommodates computational domains with over one billion grid voxels on a single shared-memory multiprocessor platform, allowing automated emergence of structures with both rich geometric features and exceptional mechanical performance. To achieve this, we track the evolution of thin structures and simulate its elastic deformation in a dynamic narrow-band region around high-density sites to avoid wasted computational effort on large void regions. We have also designed a mixed-precision multigrid-preconditioned iterative solver that keeps the memory footprint of the simulation to a compact size while maintaining double-precision accuracy. We have demonstrated the efficacy of the algorithm through optimizing a variety of complex structures from both natural and engineering systems.

CCS Concepts: • **Computing methodologies → Massively parallel and high-performance simulations**;

Additional Key Words and Phrases: topology optimization, sparsely populated grid, multigrid solver

ACM Reference Format:
Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-Band Topology Optimization on a Sparsely Populated Grid. *ACM Trans. Graph.* 37, 6, Article 251 (November 2018), 14 pages. https://doi.org/10.1145/3272127.3275012

1 INTRODUCTION

Mechanical structures in nature exhibit a broad spectrum of thin features. Such examples include interior structures supporting insect wings, animal bones, bird beaks, honeycombs, etc. These codimensional structures have negligible volume compared to the size of geometries they support. Under the influence of natural selection,

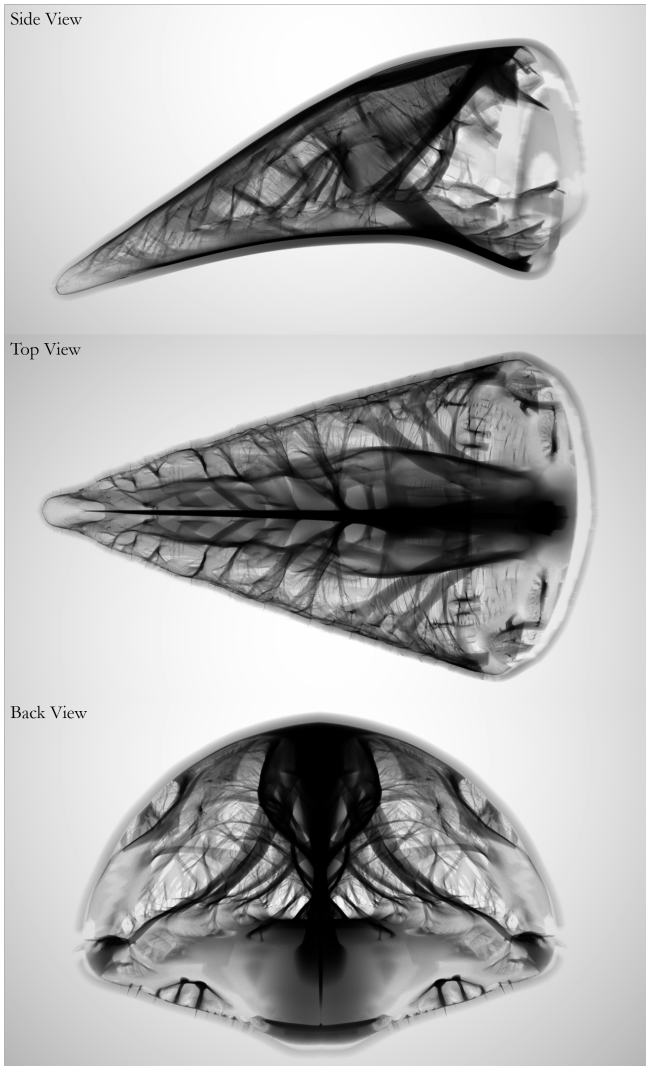

Fig. 1. *The interior structures supporting the shell of a bird beak generated using our narrow-band topology optimization algorithm with* 1, 040, 875, 347 *(1.04 billion) active FEM voxels. The resolution of the background grid is* $3000 \times 2400 \times 1600$. *The three figures show the volumetric rendering of the same structure from different views.*

*Indicates equal contribution.

Authors' addresses: Haixiang Liu, University of Wisconsin-Madison, hliu253@wisc.edu; Yuanming Hu, MIT CSAIL, yuanming@csail.mit.edu; Bo Zhu, MIT CSAIL and Dartmouth College, bo.zhu@dartmouth.edu; Wojciech Matusik, MIT CSAIL, wojciech@csail.mit.edu; Eftychios Sifakis, University of Wisconsin-Madison, sifakis@cs.wisc.edu.

0730-0301/2018/11-ART251 $15.00
https://doi.org/10.1145/3272127.3275012

these thin supportive structures enable many light, agile, and efficient systems with outstanding physical or biological properties. Computational modeling and investigation of these thin structures are crucial for understanding the hidden mechanisms and patterns of natural designs. Further, high-fidelity numerical studies of these mechanisms enable automated design, optimization, and manufacturing of a variety of human-made mechanical systems resembling their natural analogues. For example, the mathematical modeling and numerical simulation of dragonfly wings, bird beaks, and animal bones have enabled the automated design of microrobotic wings [Dileo and Deng 2009], aircraft wings [Aage et al. 2017], and artificial bone structures [Wu et al. 2017].

Topology optimization has demonstrated its efficacy in creating mechanical designs with complex structures and extreme properties in various engineering problems (see [Rozvany 2009], [Sigmund and Maute 2013], [Deaton and Grandhi 2014] for surveys). Starting from a volumetric domain that is uniformly filled with material, a standard topology optimization algorithm iteratively removes and redistributes material to develop a structure that minimizes a design objective (e.g., structural compliance), given the prescribed target volume and boundary conditions. However, due to the limitations of the previous computational frameworks, it is challenging to perform a standard topology optimization algorithm to emerge and evolve these thin and sparse features. For example, to model the evolution of a thin sheet at the length scale of ten micrometers within a centimeter cube, it requires an FEM solver discretized on a 1000^3 Cartesian grid. This grid resolution amounts to the order of magnitude of one billion active elements.

Recent advances in supercomputing provide some solutions that overcome this challenge. For example, Aage et al. [2017] ran a parallel topology optimization program on a cluster with 8000 cores for days and obtained the structure of an airplane wing with the computational domain filled with 1.1 billion voxels. A variety of novel, intricate, and multi-scale structures naturally emerge from the super-resolution computations. This result is impressive, yet the limited accessibility and high cost of the supercomputing resources impede the use of such numerical approaches in a broader range of research and engineering applications. New computational tools, which are easy to access, efficient to run, and able to solve super-resolution systems, are needed in the scientific community.

To this end, we propose a new computation framework that combines a sparse representation in conjunction with a highly optimized elastic multigrid solver for topology optimization, which delivers a significant leap in solving super-resolution problems from the prior state-of-the-art results. Our framework has enabled the simulation of a sparsely populated computational domain on the level of billion active grid voxels on a single workstation. By examining the simulation results at such scale, we identify and analyze new challenges that have not been addressed, or even observed, in the previous research on elastic deformation simulation and topology optimization, such as poor asymptotic convergence of standard Galerkin-coarsened multigrid algorithm and the algorithmic limits of floating-point precision.

To address these scale-dependent challenges, we combined design practices and algorithmic interventions on data structures, numerical methods, and parallel solver implementations. First, our framework is centered around a sparsely populated grid data structure, which allows the dynamic allocation of the degrees of freedom within a narrow band around the structure. This data structure combines the benefits of sparse storage, implicit topology representation, and the performance potential of high-throughput stream processing. We have observed and demonstrated numerically that the elements with high structural sensitivities, which are essential for developing the structure towards its optimal topology, tend to bundle within a narrow band around the structure during evolution. In contrast, the vast bulk of void regions far from the structure contribute little to the total structural compliance, but they are the primary source of the computational cost in a dense discretization. Therefore, the computation can be dramatically reduced by using a dynamically allocated sparse discretization, i.e the narrow-band representation.

In addition to the narrow-band representation, we developed a novel mixed-precision multigrid solver that is capable of solving FEM discretized linear elasticity in the order of billion elements on a single workstation. By proposing an SPGrid-optimized matrix-free formulation for data storage and a novel mixed-precision computation, the solver can meet the memory storage and bandwidth demands of a multiprocessor workstation while maintaining high accuracy. Also, our vectorized multigrid solver takes advantage of the AVX512 instruction set on the Intel Skylake-X/SP architecture in order to further enhance performance.

We summarize the main contributions of our work as follows:

- We demonstrate an ensemble of data structures, numerical schemes, and implementation best practices to perform topology optimization with the highest resolution (over one billion voxels) on a single shared-memory multiprocessor.
- We propose a sparse, adaptive topology optimization framework where simulation of elastic deformation is restricted to a narrow band surrounding the high-density region.
- We demonstrate the capacity of our framework to obtain a variety of complex thin and codimensional features, such as thin films and beams, that previous approaches might suppress.

2 RELATED WORK

Uniform and adaptive grids. Among the various kinds of data structures that have been developed by researchers in computational science and computer graphics, uniform grids play a central role in the vast majority of topology optimization applications for their multiple advantages in computation. These advantages include cache-coherent memory access, regular subdivisions for parallelization, simple data layout, and, in particular, the existence of efficient numerical PDE solvers. A multigrid FEM solver discretized on a uniform grid has been established as one of the standard solutions for elastic solids [Zhu et al. 2010], character skinning [McAdams et al. 2011], and topology optimization (e.g. [Sigmund and Torquato 1999]). One of the main challenges of uniform grids lies in their lack of adaptivity. Due to their uniformly distributed and axis-aligned grid lines, it is difficult to dynamically allocate fine grid cells around the regions of interest while still preserving all the computational

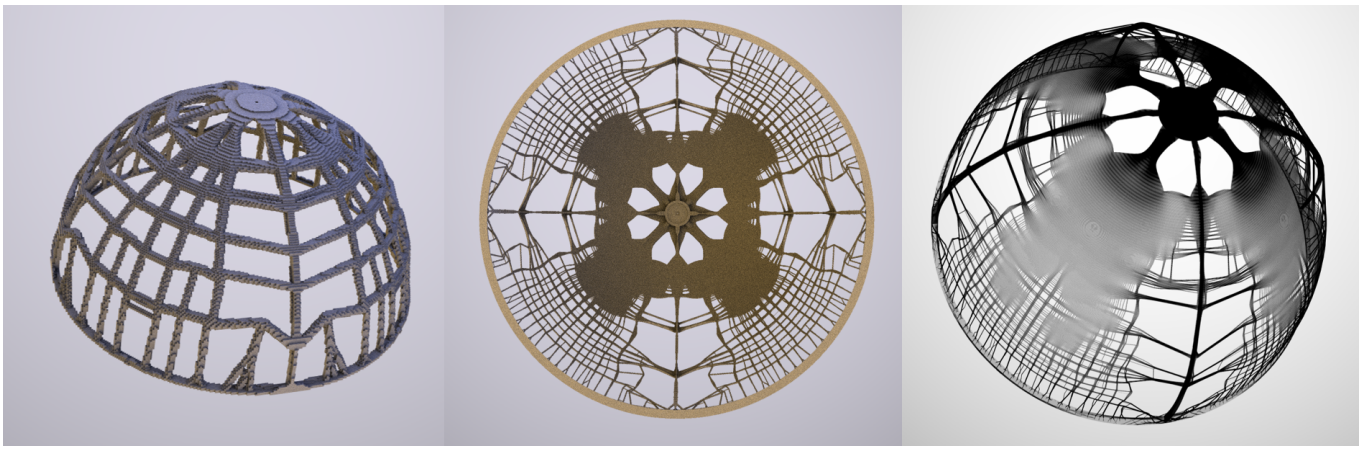

Fig. 2. *Structures with codimensional thin features are obtained by running the narrow-band algorithm on a thin hemispherical shell. The boundary conditions include a fixed boundary on the top and an external load applied on the cross surface on the bottom. As the grid resolution increases from to* 512^3 *(left) to* 2048^3 *(right), thin-shell structures are observed to dominate, matching the tendency studied in [Sigmund et al. 2016].*

advantages. To overcome this limitation, researchers have been devoting efforts to create multiple levels of resolutions by dynamically refining grid cells, which lead to the invention of a variety of hierarchical data structures, e.g., the octree grid [Losasso et al. 2004] and the AMR grid [Donea et al. 1982]. These adaptive data structures have been integrated with high-performance solvers for computing large-scale systems. For example, Ferstl et al. [2014] combined an octree grid with a multigrid-preconditioned conjugate gradients (MGPCG) solver for Poisson problems in large-scale liquid simulation. In addition, new dynamic structures have been proposed to obtain adaptivity without breaking the topology of a uniform grid, e.g., by overlapping grids with different resolutions [English et al. 2013] or anisotropically stretching grid cells [Zhu et al. 2013].

Lagrangian approaches. An unstructured Lagrangian mesh inherently exhibits adaptivity. It is natural to allocate degrees of freedom on a Lagrangian mesh adaptively, according to some local meshing criteria, e.g., the distance to structures of interest. Further, Lagrangian approaches are capable of tracking the structure interface by explicitly maintaining the boundary mesh. These two main advantages allow Lagrangian meshes to be widely used in topology optimization (see [Eschenauer et al. 1994], [Sokolowski and Zochowski 1999], [Christiansen et al. 2014], [Christiansen et al. 2015] for examples). The main limitation of Lagrangian approaches lies in their inefficiency in solving large-scale linear systems. The unstructured nature of a Lagrangian mesh makes it difficult to build an efficient preconditioner for iterative solvers.

Sparse representations. In contrast to uniform grids, sparse data structures reduce the computational cost of a volumetric discretization by maintaining active elements selectively. The key idea for sparse data structures is to establish a mapping from a grid cell index in the real, sparse space to an index in the virtual, compact storage, enabling the allocation of computational resources only to grid cells occupied by or near the real structures. This mapping can be implemented by a standard hash table (e.g., local level set [Brun et al. 2012]), an octree (e.g., adaptive distance field [Frisken et al. 2000], OpenVDB [Museth 2013]), or via a Virtual Memory Page Table and the Translation Lookaside Buffer (TLB) (e.g., SPGrid [Setaluri et al. 2014]) for fast access to grid cells. In addition to using a single type of discretization, researchers have also invented a variety of hybrid data structures to model thin phenomena embedded in a high-dimensional space. One example is the particle level set [Enright et al. 2002], which maintains a narrow band of particles around an implicit interface discretized on a background grid. The coupling of the two representations enables an accurate computation for the signed distance function. Similar concepts can be seen in the narrow-band FLIP method [Ferstl et al. 2016] and the hybrid grid-mesh approach [Zheng et al. 2015], where a thin layer of Lagrangian elements are hybridized with a Eulerian discretization to efficiently capture the features around the interface.

Hardware acceleration. At the heart of a high-resolution topology optimization algorithm is a highly efficient numerical solver for FEM discretized linear elasticity. Hardware acceleration is essential to boost the performance of these solvers. GPU-based approaches have been widely used in speeding up topology optimization algorithms. For example, Wu et al. [2016a] have proposed a high-performance multigrid FEM solver with deep integration of GPU hardware to achieve good convergence, low memory consumption, and reduced bandwidth requirements. More related work on GPU-based topology optimization can be seen in [Wadbro and Berggren 2009] [Schmidt and Schulz 2011] [Challis et al. 2014] [Yadav and Suresh 2014]. The power of supercomputing has also been explored. In [Aage et al. 2015], a parallel topology optimization framework was proposed using the Portable and Extendable Toolkit for Scientific Computing (PETSc). A variety of mechanical designs with intricate thin features was obtained by running high-resolution simulations (with up to 83 million elements) on a supercomputer. This computational framework was further used in [Aage et al. 2017] to explore the optimal design of airplane wings at the level of one billion voxels. Besides GPU and supercomputer, multi-accelerator heterogeneous

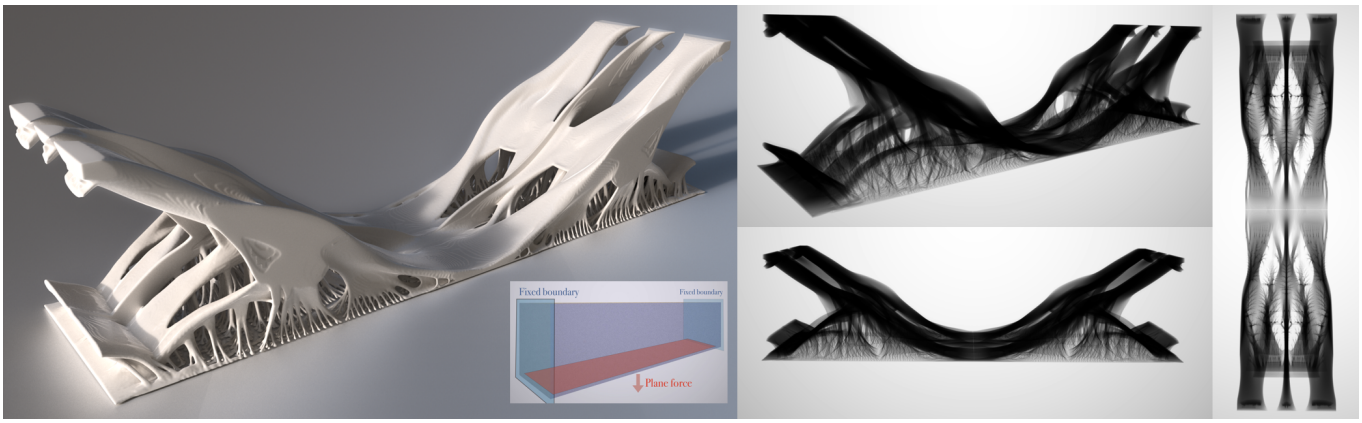

Fig. 3. *An optimized bridge structure is obtained on a* $271 \times 540 \times 1081$ *grid (157 M voxels) with mirror boundary conditions applied on the X and Z axes. The left figure shows a solid render, and the right three figures show the visualization of the density field from different camera views. The middle sub-image illustrates the boundary conditions, including fixed boundaries on the sides and a load applied on the bottom plane.*

computing has also been explored. For example, a scalable parallel solver on a workstation fitted with GPUs or many-core accelerators has been proposed in [Liu et al. 2016] to solve systems in the order of a billion degrees of freedom. The proposed Schur-complement numerical technique allows to benefit from the high memory and compute bandwidth of GPUs for large-scale problems.

3 METHOD OVERVIEW

Our sparse topology optimization framework consists of three key components: a sparse grid structure, a high-resolution multigrid FEM solver, and a narrow-band and unbounded structure optimizer. First, we briefly introduce the sparse paged structure (SPGrid) [Setaluri et al. 2014] as the base data structure to track the optimizing structures (Section 4). Next, we discuss our multigrid FEM solver for computing large-scale elastic systems on SPGrid in Section 5. We then discuss our topology optimization algorithms to obtain structures with thin features based on the sparse representation in Section 6. In Section 7, we provide method validations with respect to both the multigrid solver and the topology optimizer, with a particular focus on the comparison with [Wu et al. 2016a] to highlight the capability of our solver in handling large-scale systems. In addition, we demonstrate a variety of examples to optimize structures that exhibit complex thin features.

4 SPARSELY POPULATED GRID STRUCTURE

The Sparsely Populated Grid (SPGrid) data structure [Setaluri et al. 2014], in comparison to other sparse data structures, leverages the virtual memory system to allocate a very large virtual memory address span, corresponding to a sparsely populated background grid, while only materializing in physical memory the parts of this grid that are active. The allocation unit in SPGrid is a *block*, a rectangular region of the Cartesian grid that is made contiguous in memory address space by virtue of a space-filling traversal scheme; The size of SPGrid blocks is chosen to be a multiple of a 4KB, i.e. the size of a physical memory page. SPGrid stores a number of data channels, which have similar sparsity pattern and corresponding indexing, in the same allocation block. Using the SPGrid data structure enables us to compute effectively on a sparsely populated domain with effective bandwidth comparable to a cache-optimized, dense uniform grid. In our multigrid FEM solver, we utilized the flexibility of SPGrid's block size, and chose a block size of $4 \times 4 \times 8$, tailored for vectorization as described in Section 5.

5 MULTIGRID SOLVER

Our topology optimization framework utilizes a numerical solver for a lattice-based Finite Element Method (FEM) discretization of linear elasticity, with spatially varying material parameters. In order to accommodate the large resolutions targeted by our method, we employ a solver based on the Multigrid-Preconditioned Conjugate Gradients (MGPCG) algorithm. Algebraically, our methodology is similar to prior formulations of multigrid-preconditioned solvers [Dick et al. 2011; Wu et al. 2016b], in employing a hexahedral discretization of the linear elasticity operator, leveraging Galerkin coarsening to generate coarse level operators, and using a symmetric V-cycle as a preconditioner for Conjugate Gradients. We deviate from standard practices as documented in prior work by way of design choices, data structures, and parallelization practices including:

- A matrix-free implementation of the finest-level elasticity operator tailored for the SPGrid sparse storage structure.
- A bandwidth-saving construction of the Galerkin coarsened operator at each level, which avoids streaming through explicitly-built matrices as input any only relies on material properties at the finest level.
- Storage of the explicitly formed coarse grid operators in a novel, SPGrid-specific banded sparse matrix format.
- A modified eight-color Gauss-Seidel smoother designed to optimize memory bandwidth utilization and SIMD efficiency.
- A mixed-precision implementation of MGPCG, which combines the accuracy of double-precision arithmetic with the storage saving of single-precision representations.

Matrix-free design of finest level operator. Due to the size of the simulation domains we target, economy in memory footprint is essential to our approach. An explicit matrix storage at any level of the discretization would necessitate 243 scalar coefficients per lattice node, consisting of 27 spokes of 3×3 matrices. This number excludes any compaction afforded by storing only the symmetric half of the operator, but also does not account for any additional storage of matrix indices (e.g. in compressed sparse row format), or explicit topology storage of the computational domain (e.g. explicit hexahedral mesh).

The SPGrid data structure provides the abstraction of a sparsely populated grid, without the need to explicitly encode the connectivity of hexahedral simulation elements (e.g. as in an explicit mesh structure), as it is implied by the background uniform grid topology. Although our computational domain is embedded in a large background regular grid (up to the resolution of $3000 \times 2400 \times 1600$), its active cells in our simulation are only a sparse subset (up to 1.04 billion active cells, in our tests). The SPGrid structure allows us to directly store these active cells, their material parameters as well as their nodal forces and displacements in a sparsely populated grid.

At the finest level, we implemented a numerical kernel that computes the elastic forces resulting from the elasticity operator, for all nodes of a given $4 \times 4 \times 8$ block of the SPGrid structure; this kernel is designed to be free of write-dependencies, hence all blocks can be processed in parallel on different threads.

Consider a single grid cell with nodal displacements $\{\mathbf{u}_i\}_{i=1}^8$. We denote by $\{\mathbf{f}_i\}_{i=1}^8$ the nodal forces produced by the elastic response of the same cell, which can be expressed as $\mathbf{f}_i = \sum_{j=1}^8 \mathbf{K}_{ij} \cdot \mathbf{u}_j$. Each $\mathbf{K}_{ij}$ in this expression is a 3×3 matrix; we store these coefficients in a $8\times8\times3\times3$ tensor $\mathcal{K}$, whose elements are given by $\mathcal{K}_{ijvw} = [\mathbf{K}_{ij}]_{vw}$. This tensor is given as a linear combination of two canonical tensors $\mathcal{K}^\mu$ and $\mathcal{K}^\lambda$ based on the Lamé coefficients, corresponding to the stiffness matrices computed for $(\mu, \lambda) = (1, 0)$ and $(\mu, \lambda) = (0, 1)$ respectively. They can be computed either by analytic integration of the linear elastic trilinear element, or an eight-point Gauss quadrature rule. Ultimately, the elemental stiffness tensor is expressed as $\mathcal{K} = \mu\mathcal{K}^\mu + \lambda\mathcal{K}^\lambda$. We use the notation $\mathcal{C}(i)$ for the set of eight cells incident to node i, while $\mathcal{V}(c)$ denotes the eight vertices at the corners of cell c. We can then express the total force on node i as:

$$\mathbf{f}_i = \sum_{c \in \mathcal{C}(i)} \sum_{j \in \mathcal{V}(c)} (\mu^{(c)} \cdot \mathbf{K}^\mu + \lambda^{(c)} \cdot \mathbf{K}^\lambda)_{i^{(c)} j^{(c)}} \cdot \mathbf{u}_j$$

where $\mu^{(c)}, \lambda^{(c)}$ are the parameters of cell c, and $1 \le i^{(c)}, j^{(c)} \le 8$ are the local indices of nodes i, j as vertices of cell c. This operation can be trivially vectorized; let $\underline{\mathbf{f}} := (f_{(p,q,r)}, f_{(p,q,r+1)}, \ldots, f_{(p,q,r+7)})$ be a SIMD vector of eight forces on nodes that are sequential along the z-axis, while $\underline{\mu}^{(c)}, \underline{\lambda}^{(c)}$ are the parameters of a cell neighbor for each of the eight sequential grid indices, and finally, $\underline{\mathbf{u}}_j$ the set of eight sequential nodal neighbors at a specific offset direction. The previous operation is then expressed as:

$$\underline{\mathbf{f}} = \sum_{c \in \mathcal{C}(i)} \sum_{j \in \mathcal{V}(c)} (\underline{\mu}^{(c)} \cdot \mathbf{K}^\mu + \underline{\lambda}^{(c)} \cdot \mathbf{K}^\lambda)_{i^{(c)} j^{(c)}} \cdot \underline{\mathbf{u}}_j. \qquad (1)$$

We note that a SIMD line need not be structured strictly along a sequence of nodes aligned along the z-axis (or any other single axis); a rectangular arrangement, e.g. eight nodes straddling a 2×4 rectangle along the y- and z-axes would work in exactly the same fashion. From a programming standpoint, equation (1) indicates that SIMD vectors of each Lamé parameter are scaled with a constant coefficient (a single multiply instruction, with embedded broadcast, in the AVX2 and AVX512 instruction sets), and multiply the respective neighboring displacements $\underline{\mathbf{u}}_j$ (a fused multiply-add operation) to compute a term contributing to the nodal force $\underline{\mathbf{f}}$. With the increased number (32) of available registers in AVX512, we have verified that the entire stencil application can be executed with 2×24^2 multiply and 2×24^2 fused multiply-add instructions (FMA) without any register spilling, and enough distance between operation dependencies to allow the full throughput of two FMA instructions per cycle in Skylake-X/SP[1] (approximately 1200 cycles for the stencil application on one 16-wide SIMD line).

Modifications to the SPGrid structure. In order to accommodate the SIMD-heavy stencil computations in our elasticity operator, we enacted two crucial modifications/enhancements of the baseline implementation (http://www.cs.wisc.edu/~sifakis/SPGrid.html) of the SPGrid structure of Setaluri et al. [2014]. The first of those, is a vectorized load routine, with a compile-time stencil offset, as in:

```
  template <int di,int dj,int dk,class SPG_array>
__m512 VectorGet<di,dj,dk>(SPG_Array a,int64_t offset)
```

where a 16-entry SIMD width in single-precision has been used, as an AVX512 example. The offset given as argument is presumed aligned at SIMD-width granularity, while a stencil offset `(di,dj,dk)` is given as a compile-time argument. Using these semantics, grid data at a given stencil "spoke" (from an aligned baseline vector address) can be loaded for an entire SIMD line, as illustrated in Figure4. Even though the stencil shift might cause data to originate

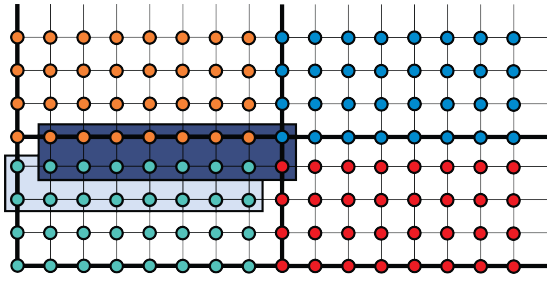

Fig. 4. *The* `offset` *passed to* `VectorGet` *points to an sequence of SPGrid entries with SIMD-width alignment (light blue box). Using a stencil offset, e.g., shifts the region to be loaded by* `VectorGet`*, as shown in the dark blue box. The data of this shifted box may originate in several distinct SPGrid blocks (indicated by different node colors).*

from different SPGrid blocks, the fact that such shift is known at compile time allows significant optimizations that avoid expensive gather operations and minimize address translations.

[1]Intel Xeon Gold 6140 processor (18 cores at 2.30 GHz) with 192 GB memory.

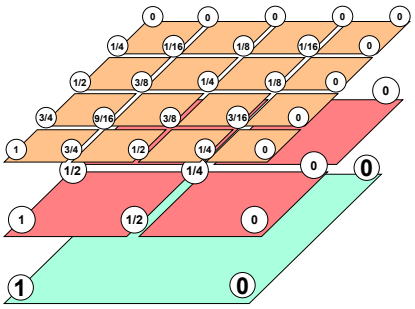

Fig. 5. *Two successive prolongation operations on a Kronecker delta function during Galerkin coarsening of a coarse cell, illustrated in 2D.*

The second SPGrid modification was the relaxation of the design restriction in Setaluri et al. [2014] that each SPGrid block be sized to exactly 4KB (the size of a virtual memory page). Our solver used a total of 128 bytes for all variables stored at each grid index, leaving the block size to just 2×4×4 when a 4KB block size is used. We found it more effective to be able to use a larger block size, namely 4×4×8 in order to (a) minimize the number of SIMD stencil accesses that straddle multiple blocks, and simplify implementation of `VectorGet`, and (b) allow an adequate number of nodes to be present per-block, to ensure that even after eight-coloring (as required by our smoother, described later in this section), the nodes on each color are a multiple of the SIMD width, even on AVX512 systems. We include source code for both proposed SPGrid modifications, as a supplement to our paper. As an indication of performance, we have achieved an effective bandwidth of 17.45 GB/s for our multiply kernel.

Efficient Galerkin Coarsening. In the construction of the multigrid hierarchy, we used the Galerkin coarsening method, computing the coarse grid operator as $\mathbf{K}^c = \mathbf{P}^T \cdot \mathbf{K}^f \cdot \mathbf{P}$, where $\mathbf{P}$ is the prolongation matrix and $\mathbf{K}^f$ the fine grid operator. At all but the two finest levels, we can afford to store the coarsened operator explicitly, as the reduced dimensionality of the coarse grid allows us to do so at one-eighth of the memory footprint that such matrix would have occupied at the finer level. Our construction of $\mathbf{K}^c$ is tailored around the following implementation objectives:

- Neither $\mathbf{K}^f$ or $\mathbf{P}$ are presumed available in matrix form.
- The rows of $\mathbf{K}^c$ should be computed independently, to avoid write hazards. †
- We seek the flexibility to compute $\mathbf{K}^c$ at *any* coarse level directly from the material parameters at the finest level, without depending on operators at intermediate levels.

Let us consider the specific example of constructing the operator $\mathbf{K}^{4h}$ at two levels coarser from the finest grid :

$$\mathbf{K}^{4h} = \mathbf{P}^T_{4h\to 2h}\mathbf{P}^T_{2h\to h}\mathbf{K}^h\mathbf{P}_{2h\to h}\mathbf{P}_{4h\to 2h} \tag{2}$$

The coefficients of the i-th row of this matrix (or equivalently, the i-th column, due to symmetry) are given by the action $\mathbf{K}^{4h}\mathbf{e}_i$ of this operator on the basis vector $\mathbf{e}_i$. Equation (2) suggests that this action can be computed by successively prolongating $\mathbf{e}_i$ to the finest level, applying the fine-grid operator $\mathbf{K}^h$, and restricting the result back to the coarse grid. We perform this operation separately on each of the eight cells (at level $4h$) incident on node i, as illustrated in Figure 5. The input to this process is a discrete Kronecker delta, shown as the input coefficients to the coarsest level. We can use an eight-wide SIMD register to store all eight nodal values of this coarse cell. We have implemented a routine ProlongateCell() that interpolates this eight-value SIMD register into eight more eight-wide registers corresponding to the nodal values of the child cells at the immediate finer level. This routine is called recursively to prolongate all the way to the finest level. At that point, a routine CellMultiply() is used to compute the force response of each individual fine cell to these prolongated nodal displacements (using the material properties at the finest level). A routine RestrictCell() implements the adjoint of the prolongation operation by collecting force contributions from fine child cells to their coarser parent. Since these routines are called recursively, all SIMD vectors are stack-allocated and can be effectively cached. As an indicator of performance, the construction of the entire operator hierarchy in our 1.04 billion-voxel example (Figure 1) requires 113.9 seconds using AVX512 instructions, which is a very small fraction of the MGPCG cost at this resolution.

Sparse Matrix Storage. The storage of the Galerkin-coarsened matrix needs to be handled as to exploit sparsity, facilitate the application of the smoother routine, and allow a direct solver (in our case, Intel MKL PARDISO) to be used for solving the problem at the coarsest level of the hierarchy. Given that the topology of the background is a regular Cartesian lattice, we use a band-storage approach, where the 243 nonzero coefficients associated with the stencil each node (a 3×3 matrix for every spoke of a 27-connected $3 \times 3 \times 3$ stencil) are stored into a secondary SPGrid structure. We supplement these 243 scalars with a bit field, indicating whether each stencil spoke is structurally present in our discretization. This representation allows straightforward implementation of the smoother routine, and can be easily converted to compressed sparse row (CSR) format for usage in direct solvers like PARDISO. In this conversion, the only serial operation is the calculation of linearized indices, and the necessary allocated length of each compressed row; the data transfer into the CSR coefficient buffer is performed in parallel over SPGrid blocks.

Optimization of the relaxation routine. In order to balance convergence efficiency with parallelization potential, we employ an eight-color Gauss-Seidel (GS) routine, as other authors have similarly adopted in prior work [Wu et al. 2016b], with the slight modification that we collectively update all three collocated degrees of freedom at each grid node, by inverting the 3×3 diagonal block of the stiffness matrix corresponding to that node. A drawback of combining the eight-color GS smoother with a SIMD implementation is the suboptimal utilization of memory bandwidth, as each SIMD vector will require data that is consistently discontinuous in memory (Figure 6, left). Such scattered memory assess is particularly wasteful for modern hardware which always performs load operation from memory at cache-line granularity. In order to circumvent the need for such scattered data access, we preemptively transpose the data in each SPGrid block as to reorder the indices of each color to be consecutive in memory (Figure 6, right/bottom). We observe that applying the same stencil offset to the nodes of one color always leads to nodes of a different yet consistent color (e.g., in Figure 6, applying a (+1,+1) offset to a yellow node always leads

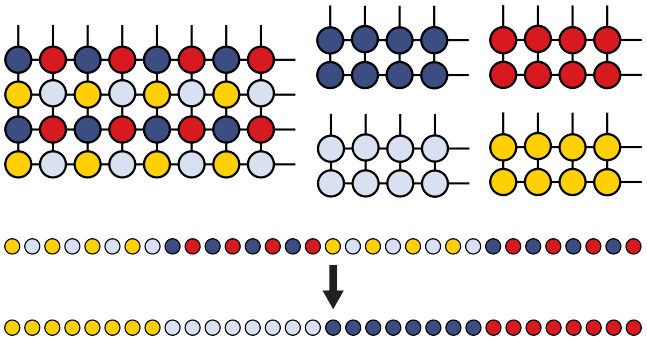

Fig. 6. *Left: a SPGrid block of 4x8 in which the nodes are ordered lexicographically, Right: the transposed colored storage enables efficient vectorization for Gauss-Seidel iteration. Bottom: Transposition operation of a SPGrid block illustrated in a linear memory layout.*

to a red node). As a consequence, after the described transposition, applying the colored GS smoother on each individual color can be performed by a straightforward application of the `VectorGet` routine to the transposed data. Each SPGrid block is transposed back to the original ordering at the end of the application of the relaxation routine. We have observed an effective bandwidth of up to 68 GB/s (out of maximum 128 GB/s) on a Skylake-SP platform for the eight-color GS routine.

A mixed-precision MGPCG solver. The linear systems arising from the equations of elasticity in our large-scale topology optimization tasks impose a unique set of challenges to the numerical algorithms used. Due to both the sheer size of the computational domains we seek to accommodate, and the large contrast of material stiffness values used in different regions of the simulated domain, we often encounter situations where an MGPCG solver using single-precision (32-bit) floating-point arithmetic cannot sustain satisfactory convergence, or even instances where the solver will plainly diverge. There are also scenarios where single precision will have catastrophic consequences on our solver, when Galerkin-coarsened matrices will be reported as effectively "singular" by direct solvers (i.e. MKL PARDISO) if constructed to single precision. We note that the frequency of incidence of such issues was dramatically increased, in our experience, when dealing with domains in excess of 10^8 voxels, while lower-resolution problems would be significantly more resilient.

An MGPCG solver implemented natively in double precision was fully effective for all the examples in our paper. However, using double precision would double our memory footprint, which was a significant concession given our pursuit of exceptionally high-resolution domains. We thus designed a variant of such solver that used a carefully crafted mix of single- and double-precision arithmetic, which we have found to produce results of effectively identical accuracy as a native double-precision solver. Consider the main loop of a preconditioned conjugate gradients algorithm, as captured in the following pseudocode:

$$
\begin{aligned}
&\textbf{for } k = 1 : N \\
&\quad \mathbf{q}_k \leftarrow \mathbf{A}\mathbf{p}_k && (3)\\
&\quad \alpha_k \leftarrow \mathbf{r}_k{}^T\mathbf{z}_k / \mathbf{p}_k{}^T\mathbf{q}_k \\
&\quad \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k\mathbf{p}_k && (4)\\
&\quad \mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \alpha_k\mathbf{q}_k \\
&\quad \mathbf{z}_k \leftarrow \mathbf{M}^{-1}\mathbf{r}_k && (5)\\
&\quad \beta_k \leftarrow \mathbf{z}_{k+1}{}^T\mathbf{r}_{k+1} / \mathbf{z}_k{}^T\mathbf{r}_k \\
&\quad \mathbf{p}_{k+1} \leftarrow \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k
\end{aligned}
$$

Our modifications which yield a mixed-precision implementation are summarized as follows:

- Vectors $\mathbf{r}, \mathbf{q}, \mathbf{z}$ and $\mathbf{p}$ (colored blue, above) are persistently stored in single-precision floating-point variables.
- The solution $\mathbf{x}$ is stored in double precision. The accumulation operation in (4) is also performed in double precision.
- The operator application in line (3) is performed in double precision (the single-precision input $\mathbf{p}$ is up-cast to double precision prior to the multiplication). The result of the operator application is then truncated to single precision and stored into $\mathbf{q}$.
- The multigrid V-cycle used as the preconditioner $\mathbf{M}^{-1}$ in line (5) is modified as follows: The smoother at the finest level uses double precision for the application of the operator, just as in line (3), although inputs and outputs are stored in single-precision. Every level of the V-Cycle other than the finest uses double-precision arithmetic entirely.

Using this mixed-precision approach, the memory footprint of our solver is further reduced, providing a significant boost in the maximum resolution we can accommodate for a given amount of physical memory (128 bytes/node suffice to store all variables necessary for the MGPCG solver as well as the minimum compliance optimizer). Our results and validation section provides experimental evidence indicating that this mixed-precision approach yields almost identical accuracy of final results in all our tests.

6 NARROW-BAND TOPOLOGY OPTIMIZATION

We discretize the design domain using a density field $\boldsymbol{\rho}$ on cell centers. For each cell, ρ can take value from 0 to 1 to represent materials ranging from void to solid. The sparsity of the domain is specified on the granularity of SPGrid blocks: a block is marked as *void* if all of its cells have density smaller than a threshold (1×10^{-6} in our case); otherwise, the block is marked as *active* (see Figure 7). We then define the narrow band of a structure as the group of all the void blocks within the one-ring neighborhoods (26-neighbours) of the active blocks, including both the interface (narrow band) and the interior. All the blocks in the domain are marked as active initially and then updated according to the new cell densities throughout the topology optimization iterations. Only the grid cells in active blocks and the narrow band are used by the FEM solver. The information on grid cells in other blocks is not maintained on SPGrid.

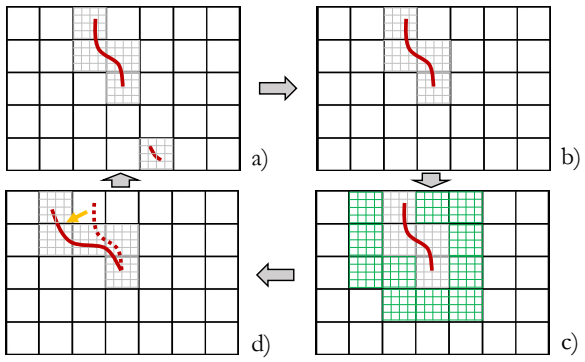

Fig. 7. *The evolution of SPGrid block activation for a single topology optimization iteration.* ***a)*** *The thin page map consists of blocks that are "hard" (i.e., with maximum voxel density above the threshold).* ***b)*** *The filtered thin page map excludes small disconnected blocks to avoid FEM solve singularity.* ***c)*** *The fat page map, where FEM solve and topology evolution happens, is an expanded version of the thin page map.* ***d)*** *The new thin page map is an eroded version of the fat page map with only hard blocks considered. Following this cycle, active (gray) blocks and narrow-band blocks (green) are updated in each iteration along with structure evolution (red lines).*

The full-domain initialization allows structures to emerge everywhere in the design space. Though such strategy requires more storage initially, numerical convergence is rapid due to the uniformity of stiffness at the beginning. The true performance bottleneck emerges in later optimization iterations, when the irregular, thinned-out domain requires more solver iterations for convergence. Restricting the cost of these later iterations to the narrow-band is a key source of the runtime savings, by omitting blocks that are away from the stiff region.

Topology optimization. We solve a classic topology optimization problem by evolving the density field on the SPGrid and updating the active blocks and the narrow band accordingly. The objective is to minimize the elastic energy of the structure constrained by the linear elastic FEM equation and the target volume $\hat{V}$. The topology optimization problem can be formulated as:

$$\begin{aligned} \min: \quad & S(\boldsymbol{\rho}, \mathbf{u}) = \mathbf{u}^T \mathbf{K}(\boldsymbol{\rho})\mathbf{u} \\ s.t.: \quad & \mathcal{F}(\boldsymbol{\rho}, \mathbf{u}) = \mathbf{K}(\boldsymbol{\rho})\mathbf{u} - \mathbf{f} = 0 \\ : \quad & V(\boldsymbol{\rho}) - \hat{V} \leq 0, \end{aligned} \tag{6}$$

where $\mathbf{K}$ is the global stiffness matrix of the entire system, $\mathbf{f}$ is the vector of external forces, and $V(\boldsymbol{\rho})$ represents the sum of ρ of all grid cells weighted by the volume of each cell. In order to avoid singularity of the stiffness matrix, we define the relationship between ρ and the stiffness matrix of each element as: $\mathbf{K}_e(\rho) = [\rho^k(1-\epsilon)+\epsilon]\mathbf{K}_0$, where $k = 3$, $\epsilon = 10^{-6}$ or 10^{-9}, and $\mathbf{K}_0$ is the stiffness matrix of an element fully filled with materials. The global stiffness matrix is assembled by summing the stiffness matrices of all the grid cells.

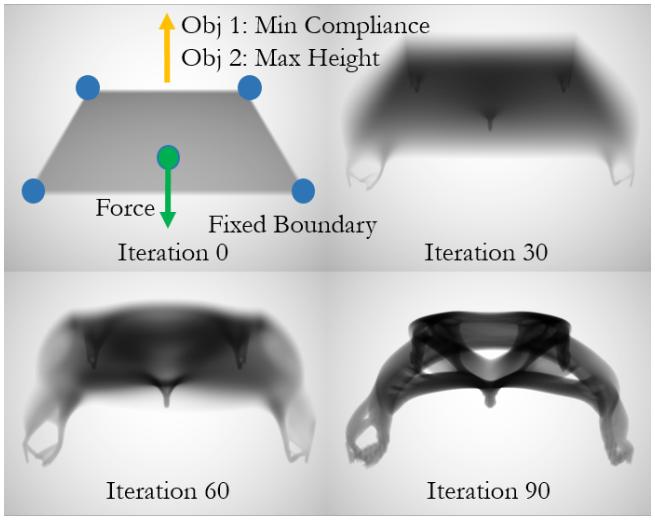

Fig. 8. *Unbounded topology optimization for a quadpod structure. A thin membrane is initialized (a), evolved upwards (b), and converge to a clear structure (c and d), guided by a combined objective of both minimum compliance and maximum height.*

We follow the standard optimality criterion scheme (e.g., see [Sigmund 2001] [Bendsøe and Sigmund 2009]) to iteratively evolve the density field on the grid. In each iteration, we first solve the FEM equation to update the nodal displacements given the current density distribution. Next, we compute the sensitivity of each active cell as $C_e = -\mathbf{u_e}^T(\partial \mathbf{K_e}/\partial \rho_e)\mathbf{u_e}$ with the updated nodal displacements. A heuristic scheme is then applied to each cell to update its density according to the new sensitivity. Notice that all the updates are performed in the active and the narrow band blocks. We refer readers to [Sigmund 2001] for more implementation details for each step.

Narrow band evolution. We use page maps to track the SPGrid block activation. Each page map is implemented as a bitmap, using one bit for each block for economical storage and fast query. For each block, either void or active, we store one bit in the page map at the cost of 32 KB per GB ambient grid storage. We maintain two page maps: a *thin* page map that tracks only active blocks and a secondary *fat* page map to track the blocks that are either active or in the narrow band. Notice that the *fat* page map is an expanded version of the thin page map. The two page maps are updated in an intertwined fashion in each topology optimization iteration (see Figure 7). First, the *thin* page map is re-populated from the *fat* page map by deactivating blocks with only low density. We then perform a connectivity filtering operation on the *thin* page map to eliminate small isolated active blocks. A connected component is considered to be isolated if its volume is less than 30% of the total active volume. This step is crucial in avoiding singularity in the FEM stiffness matrix. After the connectivity filtering step, the *fat* page map is passively updated according to the new densities at the end of the topology optimization iteration (or active blocks). This allows inactive blocks to become active again, similar to [Bruns and Tortorelli 2003]. The complete algorithm coupling the steps of FEM solve

and page map update are summarized in Algorithm 1. Most of the topology optimization algorithms require a fixed boundary for the computation domain. Our narrow-band evolution approach enables unbounded topology optimization to bypass this fixed boundary restriction. Thanks to the dynamic and sparse peculiarity of our grid structure, we can initialize active voxels within a small region and then update the computation domain following the evolving structure. The objective gradients in the active blocks within the narrow band guides the evolution of the structure. This dynamic-tracking capability opens up possibilities for many applications where the design domain is not known as a priori. As in Figure 8, we optimize the evolution of a thin membrane under a combined objective of both minimum compliance and maximum height. We assume no prior knowledge of the computation grid. All the voxels are activated automatically as the membrane grows upwards, and the structure naturally emerges from the background.

7 VALIDATIONS AND EXAMPLES

7.1 Multigrid Solver Validation

Algorithmically, our implementation of the solver is a standard multi-linearly interpolated Garlerkin-coarsened Multigrid preconditioned conjugate gradient method. At early topology optimization iterations or small domains, such method proves to have fast asymptotic convergence. But in our examples, especially the bird beak Figure 1 and the wing Figure 15, the high variation of material spatial distribution has challenged the convergence of the multigrid solver as shown by Figure 9 (left). Thiner features resulted from Higher resolution, can significantly impact the convergence of multigrid as indicated in Figure 9 (right).

Besides convergence, the high resolution has also pushed the numerical limited of floating point. To validate our mixed precision scheme, we have conducted the following tests, both on analytical structures and the structures naturally emerged from topology optimization: 1. The last iteration of the bird beak example; 2. The last iteration of the wing example; 3-8. Homogeneous and isotropic cantilever beams of different length with one side fixed and the other side loaded with a uniform downward force.

Table 1 shows the final residual of five different precision schemes: 1. Full double precision; 2. Only solution vector and computation

Algorithm 1 The narrow-band topology optimization algorithm.

```
function NarrowBandTopologyOptimization()
    Initialize density field and boundary conditions.
    while not converged do
        NarrowBandEvolution (Section 6)
        for each objective do
            FEM Solve (Section 5)
            Sensitivity analysis
        end for
        Update density field using OC
    end while
end function
```

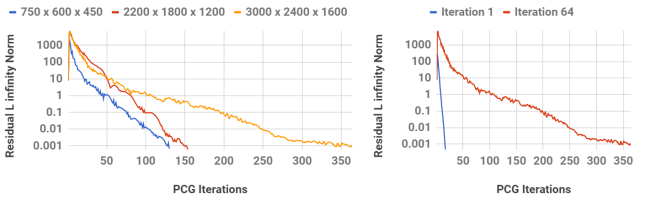

Fig. 9. *(Left) Convergence comparison of the bird beak example of different resolutions at the last topology optimization iteration. (Right) Convergence comparison of the one-billion-voxel bird beak example at different topology optimization iterations. All residuals reported in this work are L_∞ norm across all active cells, normalized relative to the L_∞ norm of the load.*

are in double precision, the mix precision scheme we used in our topology optimization; 3. Only solution vector is in double precision; 4. Only computation is in double precision; 5. All in float precision. The results shows that under all examples our proposed mix precision scheme(column 2) can reduce the residual to an order of magnitude close to the double precision. Due to the fact that at high resolutions, cells that are far from Dirichlet boundaries have large displacements but yet only small strains. This loss of precision leaves it insufficient to use single precision for the solution vector. As shown in (column 4 and 5), using single precision for solution can result in inaccurate computation of strain and final residual. Similarly, using single-precision multiply will not be able to compute search direction to sufficient accuracy, conjugate gradient, in this case, will halt due to a detected singularity (column 3 and 5).

7.2 Topology Optimization Validations

Narrow Band v.s. Full Grid. We validate our narrow-band scheme[2] by comparing it to the standard dense scheme concerning both accuracy and efficiency. We reuse the simulation parameters for the bridge (Figure 3) and the thin hemispherical shell (Figure 2) to set up our experiments. For each example, we run topology optimization with both schemes, and then we record the evolution of objectives as well as the number of active blocks during the optimization iterations. As shown in Figure 10 left, the compliance curves for the two schemes closely match each other. As shown in Figure 10 right, the narrow-band scheme uses significantly fewer active blocks than the dense one. For the bridge, the number of active blocks converges within 20 iterations to 21.64% of the total number of blocks; for the thin shell, this ratio converges to 23.46% within 25 iterations. This block reduction leads to a 2.5× and a 3.1× acceleration in computation respectively. Initially, the advantage of narrow-band is less significant compared with later topology optimization iterations, yet the FEM solver converges relatively fast on these iterations thanks to the almost uniform density field (Figure 9 right).

Figure 11 visualizes the distributions of the active blocks as well as the optimized voxels for the hemispherical shell optimization using both schemes. The difference between the two structures is imperceptible (Figure 11 bottom), though complicated filament

[2]Implemented based on the **Taichi** computer graphics library [Hu 2018].

Table 1. *The final residuals of different precision scheme after the same number of iterations. Test cases include bird beak, plane wing, and cantilever beam (CB) with different resolutions. The final residuals are recomputed based on solution vectors. All tests are scaled to initial residual infinity norm of 1. From the left to right, the 5 schemes are: 1. Full double precision; 2. Only solution vector and computation are in double precision; 3. Only solution vector is in double precision; 4. Only computation is in double precision; 5. All in float precision. SINGULAR indicates conjugate gradients have halted due to detected singularity.*

Example	Double Precision	Mix Precision (double multiply)	Mix Precision (single multiply)	Single Precision (double multiply)	Single Precision (single multiply)
Bird beak	9.192e-5	9.063e-5	1.969e-4	6.846e+0	7.611e+0
Wing	8.968e-5	1.815e-4	SINGULAR	1.850e+1	SINGULAR
CB (32x32x32)	7.942e-6	7.942e-6	5.827e-5	2.259e-5	5.827e-5
CB (32x32x64)	8.217e-6	8.218e-6	5.773e-5	4.905e-5	6.019e-5
CB (32x32x128)	8.207e-6	8.208e-6	1.041e-3	1.018e-4	1.041e-3
CB (32x32x256)	9.322e-6	9.321e-6	6.295e-3	1.934e-4	6.295e-3
CB (32x32x512)	4.506e-6	4.508e-6	SINGULAR	3.990e-4	SINGULAR
CB (32x32x1024)	5.237e-6	5.242e-6	SINGULAR	7.043e-4	SINGULAR

Table 2. *Statistics of the topology optimization examples. We report the volume fraction, the number of active voxels, the total number of iterations, and the average time spent for each topology optimization iteration. Notice that the active voxels include both voxels occupied by materials and voxels in the narrow band. We run all examples on a workstation equipped with four Intel E7-4830v4 processors (14 cores each at 2.0 GHz) and 512GB of memory. Though highly optimized, FEM solve is still the computation bottleneck, taking over 90% run time in all examples.*

Example	Ambient Grid	Volume Fraction	Max Voxels (M)	Min Voxels (M)	Iterations	Time per iteration
(Fig. 1) Bird beak	$2737 \times 1484 \times 910$	10%	1040.88	296.41	49	2.31h
(Fig. 2) Hemisphere	$2048 \times 2048 \times 2048$	4%	518.99	71.21	36	0.40h
(Fig. 15) Plane wing	$1696 \times 342 \times 1971$	20%	401.53	155.82	43	0.83h
(Fig. 3) Bridge	$271 \times 540 \times 1081$	8%	158.19	32.71	63	128.0s
(Fig. 14) Bicycle wheels	$984 \times 984 \times 204$	1%	53.56	9.56	65	659.0s
(Fig. 13) Shear plate	$640 \times 640 \times 640$	8%	262.14	71.21	30	0.70h
(Fig. 8) Quadpod	$300 \times \infty \times 300$	–	11.11	0.98	96	17.4s

and thin shell structures are developed based on very different computational discretizations (Figure 11 top). It is noteworthy that the distribution of active blocks gets not only sparser but also thinner (see the region pointed by the yellow arrow in the top middle image of Figure 11) to tightly bound the voxel evolution in all directions. In summary, the narrow-band scheme not only delivers almost identical results to the full-grid scheme, but also runs more efficiently thanks to accelerated narrow-band FEM solves, especially in later optimization iterations.

Compliance and Feature Evaluation. We explore the relationship between the complex features emerged on a high-resolution grid and the mechanical performance of the optimized structure. We start from a $20 \times 20 \times 80$ coarse grid and progressively refine it by a factor of two in all axes. We run topology optimization for a bridge on each grid and measure the compliance of the optimized structure. To this end, we first up-sample the density fields on different grids to the same finest grid ($160 \times 160 \times 640$). Then, each voxel in the density field is discretized to 0 or 1, and a FEM simulation is performed taking the discretized density as the input to measure the compliance of structure. For each up-sampled density field, we also compute the sum of L_2 norm of the density's Laplacian to approximate the complexity of the structure's fine-scale features. As shown in Figure 12, we observe from the two curves that the structures obtained from higher resolution computation exhibit finer-scale features and lower compliance. This observation is consistent with the conclusions made in a line of previous work such as [Wu et al. 2016a] and [Aage et al. 2017], necessitating the use of high-resolution computational domain to obtain designs with extraordinary mechanical properties.

7.3 Examples

Mechanical Designs. We demonstrate the efficiency of our approach by solving large-scale topology optimization problems targeting at different applications, ranging from classical mechanical designs to engineering structural optimizations and biomimetic structure explorations. These problems all exhibit sparse characteristics, with the active volume fraction ranging from 1% to 20% of the entire volume. The statistics for these examples are reported in Table 2.

We first demonstrate the capability of our approach in generating codimensional structures mixing filaments and thin films by optimizing a thin hemispherical shell with a target volume fraction of 4% (see Figure 2 and Figure 11). We observe that structures characterized by the thin film, instead of filament, are dominating the entire design domain as the grid gets refined. This observation is consistent with the recent discovery made in [Sigmund et al. 2016] that a twisted Michell sphere tends to converge to a vanishingly thin shell instead of a wire-frame when the computation resolution is high enough.

We then optimize a bridge structure to minimize its compliance under a uniformly distributed load at the bottom (see Figure 3 and 12). After 63 iterations, a rich diversity of multi-scale structures

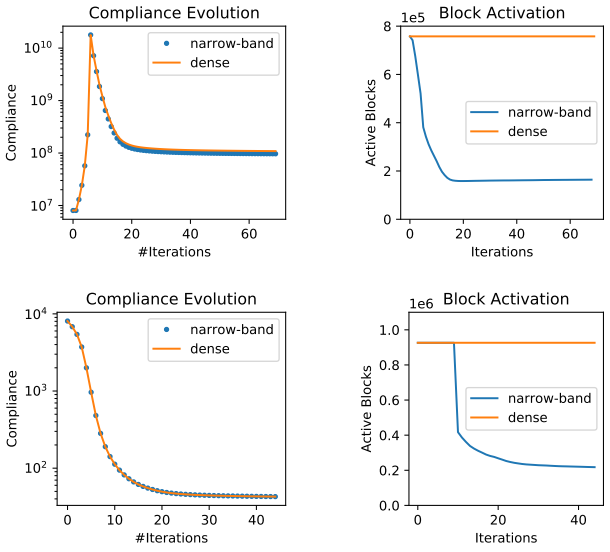

Fig. 10. *Comparison of the compliance evolution (left column) and the number of active blocks (right column) for the narrow-band and the standard dense topology optimization algorithms on the bridge (top row) and the hemispherical thin shell (bottom row). Note that the increase of the compliance in the first few iterations of the bridge example is due to progressively decreasing volume fraction to accelerate convergence.*

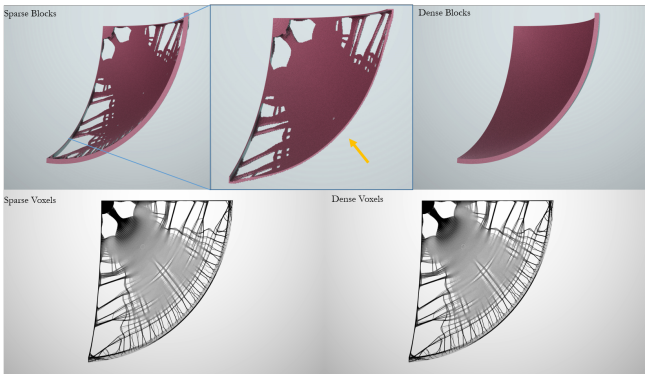

Fig. 11. *Comparison of the active blocks (top) and the optimized voxels (bottom) using the narrow-band scheme and the dense scheme. We observe negligible difference between the two optimized structures optimized by the narrow-band and dense-grid schemes.*

emerge, exhibiting very different characteristics from a standard bridge structure on a coarse grid. We then push the resolution to 262 million elements, by optimizing the supporting structures of a flat platform resisting horizontal shear forces (see Figure 13). Bunch of complicated filamentary structures emerges in the volume after 30 iterations, demonstrating the efficacy of our approach in generating structural features that were infeasible for a low-resolution algorithm.

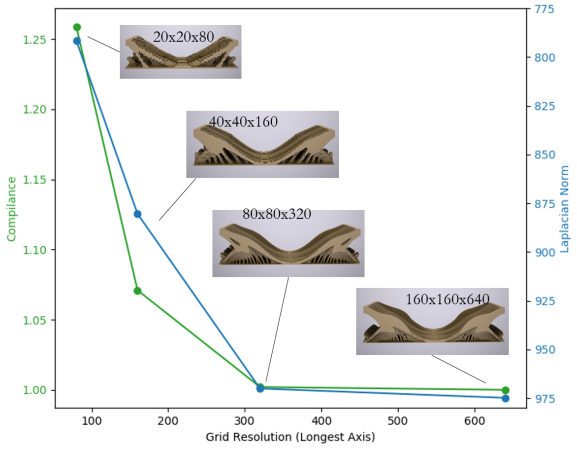

Fig. 12. *Relation between the grid resolution, the optimized structural compliance, and the level of details (measured by the sum of L_2 norm of the density's Laplacian). The plot shows that the high-resolution simulation enables detailed structures with low compliance.*

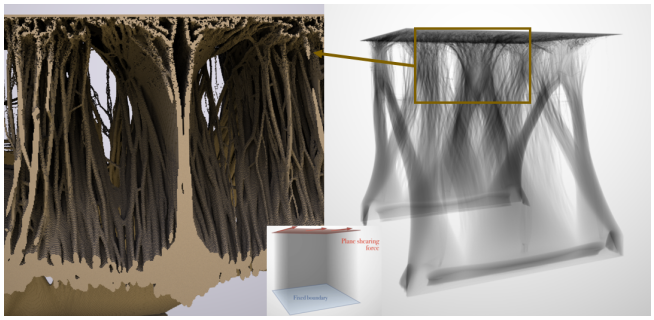

Fig. 13. *Optimized structures supporting a plate to resist horizontal shear forces are generated at a resolution of* $640 \times 640 \times 640$. *The left figure is a cross-section of the structure exhibiting various filamentary features. The right picture shows the volumetric rendering. The sub-image in the middle shows the boundary conditions, including a fixed boundary and a shear force on the bottom and top plane in separate.*

We next optimize a wheel structure to demonstrate multi-objective optimization. Different boundary conditions are enforced to obtain sparse inner supporting structures under in-plane and out-of-plane external loads (see Figure 14 a, b). We then combine the two objectives to optimize structure exhibiting outstanding mechanical properties in both cases (see Figure 14 c).

Natural Structures and Biomimetic Designs. We show the optimization of the interior supporting structure for a bird beak as in Figure 1. The dominating pressure forces are applied on both the upper and lower layer of the beak, and the fixed boundary condition is used on the left side. The target volume fraction is set to be 10% of the entire computation domain. We push the resolution to 1, 040, 875, 347 (1.04 billion) voxels, enabling the emergence of a collection of web-like

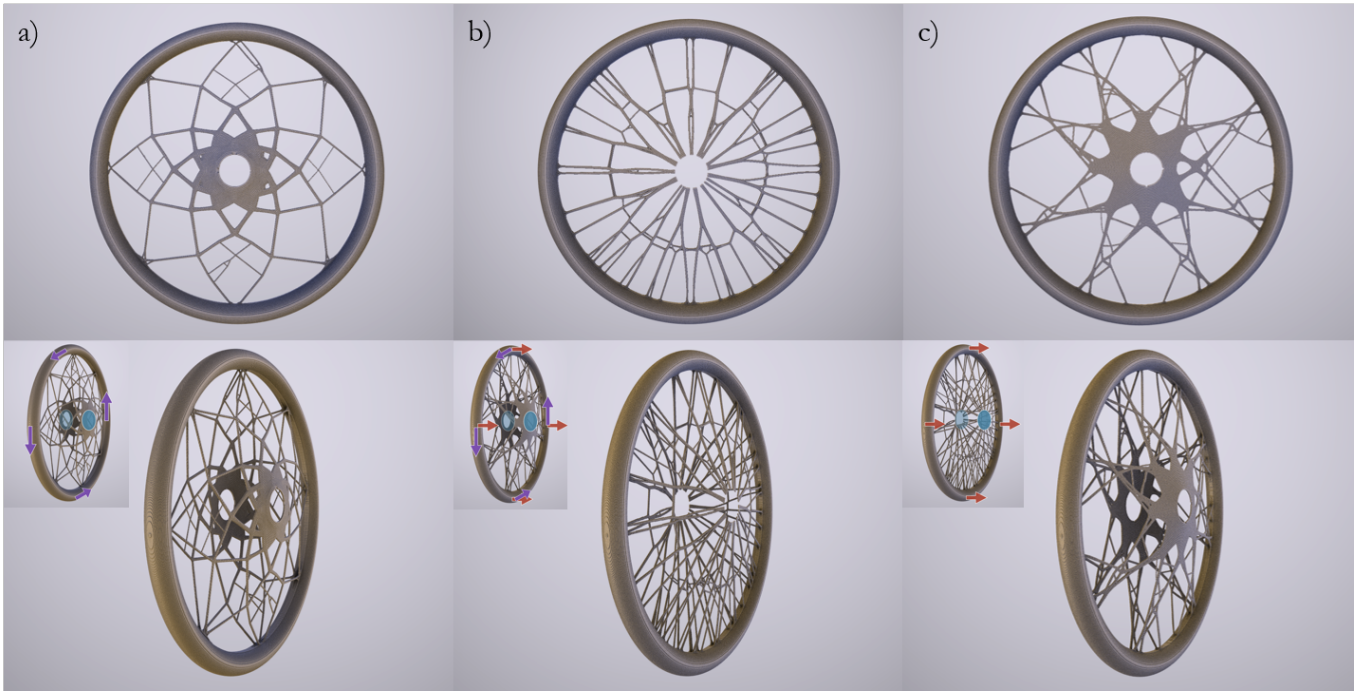

Fig. 14. *Optimized wheel structures are generated using our algorithm. Structure (a) and (b) are obtained with in-plane tangential force boundary and out-of-plane normal force boundary. Structure (c) is optimized with a linear combination of objectives in both (a) and (b).*

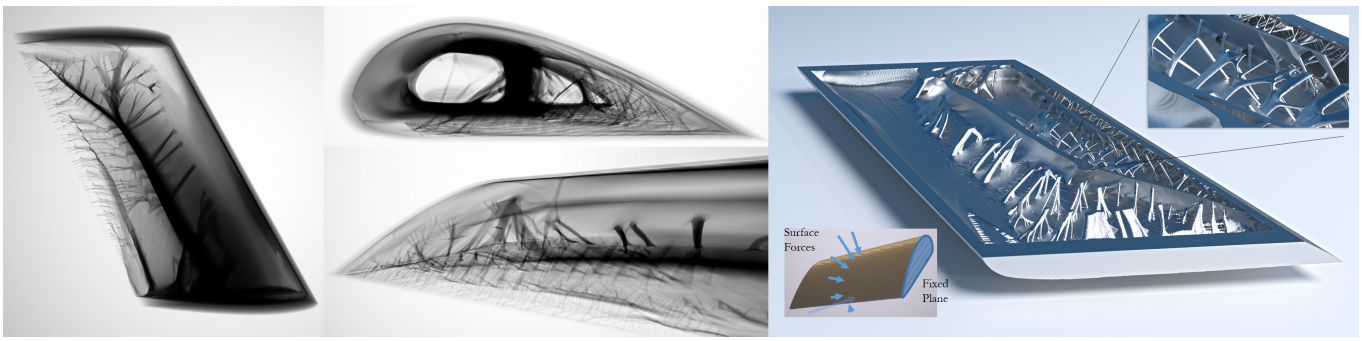

Fig. 15. *An optimized interior supporting structure of part of a wing is generated using our algorithm on a* $1696 \times 342 \times 1971$ *grid (*402 *M active voxels).*

codimensional thin structures supporting the shell of the beak while maintaining a low weight. The structural complexity of this result qualitatively resembles that of a real Hornbill bird beak as shown in [Aage et al. 2017], opening up further possibilities in exploring such natural systems using numerical topology optimization.

We further apply our approach to reproduce the design of a wing structure similar to [Aage et al. 2017]. As shown in Figure 15, the internal structures supporting the wing is optimized using the similar boundary conditions to the bird beak example, i.e., the pressure force on the wing surface along the local normal direction of the wing geometry. After 40 iterations, a variety of design details, such as the curved spars and multi-scale truss, have automatically emerged on the grid. We show the 3D prints for both the bird beak and the wing structures in Figure 16.

8 LIMITATIONS AND FUTURE WORK

Limitations. One of the main limitations of our approach is the dense representation of the domain in the first few iterations. Though the number of active blocks decreases after clear structures emerge in the domain, which significantly speeds up the solver, the full-space computation at the beginning of the optimization limits the capability of our algorithm in handling larger-scale examples. One possible solution is to initialize the domain with some narrow-band

Fig. 16. *3D prints of the slices of the bird beak and the wing structures.*

structures or coarsened discretization, which will guarantee a small number of active elements throughout the process.

For the numerical solver, increasing the resolution worsens the conditioning of the global stiffness matrix. We have observed that with higher resolution, both convergence rate and convergence limit are significantly reduced. The high-resolution filamentary structures can substantially impair the convergence of a multi-linear interpolated Galerkin-coarsened multigrid solver. One of the possible solutions for restoring the multigrid convergence is to investigate using a stencil-aware coarsening technique to better represent the high contrast stiffness ratio at the coarse levels. Finally, due to intrinsic limitations on the virtual memory subsystem, as exploited by SPGrid, we are restricted to a maximum background grid resolution of 8196^3 voxels (such resolution would require reserving a 64 TB virtual memory span at 128 bytes/node, independent of the sparsity of the computational domain; 128 TB is the total virtual memory available to user-space processes).

Future Work. There are many exciting avenues for future work combining our high-resolution numerical approach to investigate complex natural systems. Most of the real-world structures we are aiming to investigate have received little attention in the topology optimization community due to their inherent complexities and prohibitively high computational cost. Aage et al. [2017] have performed some pioneering study on supercomputers to unveil the connection between the structure of bird beaks and the design of airplane wings using topology optimization. The computational tool we have proposed can run on a single workstation, which makes the high-resolution numerical study of these structures accessible to a broader range of research teams and individuals. It might prove fruitful to collaborate with those researchers as the capability of modeling and developing these thin structures numerically enables a wide variety of computational studies that may be challenging or even infeasible before.

9 CONCLUSION

We presented a novel approach for topology optimization of structures exhibiting sparse and thin features at extremely high resolution. In particular, our computational framework makes it possible, for the first time, to computationally synthesize thin structures at the level of billion active elements on a single workstation. We have shown a variety of complex structures generated by our computational framework that share similarities with structures in nature. These intricate and multi-scale features can be seamlessly transitioned to the design of engineering systems within the same computational setting. Surveying the literature, it appears that some of these complex structures are only vaguely understood, due to the lack of computational tools for both simulation and optimization, in contrast to those thoroughly investigated engineering structures where volumetric simulation approaches are applicable. Our computational tool opens up new possibilities for scientists, engineers, and designers to explore the limits of these complex structures to develop a better understanding of their underpinning mechanisms.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their valuable suggestions and comments. Yuanming Hu is supported by the Edwin S. Webster graduate fellowship. This work was supported in part by NSF grants CMMI-1644558, CCF-1533753, CCF-1533885, IIS-1763638, CCF-1812944.

REFERENCES

Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. 2015. Topology optimization using PETSc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization* 51, 3 (2015), 565–572.

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

Martin P Bendsøe and Ole Sigmund. 2009. Topology Optimization. (2009).

Emmanuel Brun, Arthur Guittet, and Frédéric Gibou. 2012. A local level-set method using a hash table data structure. *J. Comput. Phys.* 231, 6 (2012), 2528–2536.

Tyler E Bruns and Daniel A Tortorelli. 2003. An element removal and reintroduction strategy for the topology optimization of structures and compliant mechanisms. *International journal for numerical methods in engineering* 57, 10 (2003), 1413–1430.

Vivien J Challis, Anthony P Roberts, and Joseph F Grotowski. 2014. High resolution topology optimization using graphics processing units (GPUs). *Structural and Multidisciplinary Optimization* 49, 2 (2014), 315–325.

Asger Nyman Christiansen, J Andreas Bærentzen, Morten Nobel-Jørgensen, Niels Aage, and Ole Sigmund. 2015. Combined shape and topology optimization of 3D structures. *Computers & Graphics* 46 (2015), 25–35.

Asger Nyman Christiansen, Morten Nobel-Jørgensen, Niels Aage, Ole Sigmund, and Jakob Andreas Bærentzen. 2014. Topology optimization using an explicit interface representation. *Structural and Multidisciplinary Optimization* 49, 3 (2014), 387–399.

Joshua D Deaton and Ramana V Grandhi. 2014. A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization* 49, 1 (2014), 1–38.

Christian Dick, Joachim Georgii, and Rüdiger Westermann. 2011. A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. *Simulation Modelling Practice and Theory* 19, 2 (2011).

Christopher Dileo and Xinyan Deng. 2009. Design of and experiments on a dragonfly-inspired robot. *Advanced Robotics* 23, 7-8 (2009), 1003–1021.

Jean Donea, S Giuliani, and Jean-Pierre Halleux. 1982. An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. *Computer methods in applied mechanics and engineering* 33, 1-3 (1982), 689–723.

R Elliot English, Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2013. Chimera grids for water simulation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 85–94.

Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. 2002. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys.* 183, 1 (2002), 83–116.

H. A. Eschenauer, V. V. Kobelev, and A. Schumacher. 1994. Bubble method for topology and shape optimization of structures. *Structural optimization* 8, 1 (1994).

Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. 2016. Narrow band FLIP for liquid simulations. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 225–232.

Florian Ferstl, Rüdiger Westermann, and Christian Dick. 2014. Large-scale liquid simulation on adaptive hexahedral grids. *IEEE transactions on visualization and computer graphics* 20, 10 (2014), 1405–1417.

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. 2000. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer

Graphics. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. 249–254.

Yuanming Hu. 2018. Taichi: An Open-Source Computer Graphics Library. *arXiv preprint arXiv:1804.09293* (2018).

Haixiang Liu, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. 2016. A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 201.

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 457–462.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 37.

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 27.

George IN Rozvany. 2009. A critical review of established methods of structural topology optimization. *Structural and multidisciplinary optimization* 37, 3 (2009), 217–237.

Stephan Schmidt and Volker Schulz. 2011. A 2589 line topology optimization code written for the graphics card. *Computing and Visualization in Science* (2011), 1–8.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 205.

Ole Sigmund. 2001. A 99 line topology optimization code written in Matlab. *Structural and Multidisciplinary Optimization* 21, 2 (2001), 120–127.

Ole Sigmund, Niels Aage, and Erik Andreassen. 2016. On the (non-) optimality of Michell structures. *Structural and Multidisciplinary Optimization* 54, 2 (2016), 361–373.

Ole Sigmund and Kurt Maute. 2013. Topology optimization approaches. *Structural and Multidisciplinary Optimization* 48, 6 (2013), 1031–1055.

Ole Sigmund and S Torquato. 1999. Design of smart composite materials using topology optimization. *Smart Materials and Structures* 8, 3 (1999), 365.

J. Sokolowski and A. Zochowski. 1999. On the Topological Derivative in Shape Optimization. *SIAM Journal on Control and Optimization* 37, 4 (1999).

Eddie Wadbro and Martin Berggren. 2009. Megapixel topology optimization on a graphics processing unit. *SIAM review* 51, 4 (2009), 707–721.

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

Jun Wu, Christian Dick, and Rüdiger Westermann. 2016a. A system for high-resolution topology optimization. *IEEE transactions on visualization and computer graphics* 22, 3 (2016), 1195–1208.

Jun Wu, Christian Dick, and Rüdiger Westermann. 2016b. A System for High-Resolution Topology Optimization. *IEEE Trans. on Visualization and Computer Graphics* 22, 3 (2016).

Praveen Yadav and Krishnan Suresh. 2014. Large scale finite element analysis via assembly-free deflated conjugate gradient. *Journal of Computing and Information Science in Engineering* 14, 4 (2014), 041008.

Wen Zheng, Bo Zhu, Byungmoon Kim, and Ronald Fedkiw. 2015. A new incompressibility discretization for a hybrid particle MAC grid representation with surface tension. *J. Comput. Phys.* 280 (2015), 96–142.

Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. 2013. A new grid structure for domain extension. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 63.

Y Zhu, E Sifakis, J Teran, and A Brandt. 2010. An efficient and parallelizable multigrid framework for the simulation of elastic solids. *ACM Trans. Graph* 29, 16 (2010), 1–16.