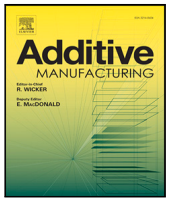
Additive Manufacturing 79 (2024) 103912

Contents lists available at ScienceDirect

Additive Manufacturing

journal homepage: www.elsevier.com/locate/addma



Research paper

OpenVCAD: An open source volumetric multi-material geometry compiler

Charles Wade, Graham Williams, Sean Connelly, Braden Kopec, Robert MacCurdy *

University of Colorado – Boulder, Boulder, CO, 80309-0020, USA

ARTICLE INFO

Dataset link: https://matterassembly.org/openvcad

Keywords:
Volumetric design
Multi-material additive manufacturing
Meta-materials
Lattice-structures
InkJet 3D printing
Functional grading

ABSTRACT

Modern additive manufacturing has made significant advancements in multi-material fabrication techniques that allow for position-specific control of material deposition. With these advancements, design tools have fallen behind machine capabilities in specifying volumetric information. Traditionally, design and fabrication workflows have expressed multi-material objects as several single-material bodies. By storing only the information about the surfaces of the geometries, information about the volumetric composition of the solids is unrepresented. The intense interest in compliant mechanisms and meta-materials demands a new design method that can support architecting material distribution throughout an object. To address these needs, we present OpenVCAD, an open-source volumetric design compiler with multi-material capabilities. OpenVCAD provides a scriptable suite of geometric and material design methods that enable efficient representation of objects with complex geometry and material distributions. OpenVCAD allows functional specification of multi-material volumes that are parameterized on spatial locations, yielding complex multi-material distributions that would be impossible to describe using alternative methods. This paper will present the OpenVCAD pipeline, compare it to related work, and demonstrate its use through the design and manufacturing of functionally graded multi-material components.

1. Introduction

Enabled by advancements in additive manufacturing (AM), geometries that would be time-consuming, costly, or impossible to fabricate using alternative methods are now relatively easy, fast, and inexpensive to realize. Notably, multi-material AM capabilities are now practical for several distinct deposition technologies. However, these developments have not been accompanied by corresponding innovations in multi-material computer-aided design (CAD) tools. Designers and engineers are increasingly in need of a methodology that allows for precise placement of multiple materials distributed throughout the full volume of a design.

The de facto standards for 3D object files in additive manufacturing workflows are the STL (Standard Triangle Language) and 3MF (3D Manufacturing Format). Both of these standards use a surface or boundary representation (b-rep) to model objects. Converting from implicit CAD formats, such as a STEP file, requires approximating the boundary surface as a triangulated mesh. The conversion from implicit to explicit representation is necessitated by slicing software. Slicers commonly rely on extracting closed, cross-sectional polygons of 3D triangulated meshes to form the layers of 3D-printed objects [1]. Consequently, engineers wishing to construct objects using multiple materials must export each component of their assembly as discrete bodies and manually assign materials in slicing software. This is known as homogeneous multi-material design. This workflow fundamentally limits multi-material objects to simple designs with discrete material transitions. For designs with several materials graded or blurred together, the existing boundary-driven workflow is ineffective. Similarly, boundary representations are incapable of expressing how material distribution changes throughout a volume. Although the 3MF format allows for a single file to contain material information, it is still limited to expressing a body as a single material. These design approaches and representation standards are incompatible with recently emerged multi-material AM methods that can deposit varying materials at high resolution within a design. To address this limitation, there have been multiple efforts to create a design method for heterogeneous objects, however they lack the operations and exchange formats necessary for modern voxel-level control on AM systems. Similarly, these methods are limited by a trade off between an object's geometric and multi-material complexities [2]. To address this gap, we propose OpenVCAD, a volumetric design compiler. We aim to provide modern AM designers and engineers a framework for efficient multi-material volumetric description of complex shapes comprised of multiple materials.

In this paper, we highlight the increasing relevance of volumetric modeling, describe existing related design tools, and demonstrate

* Corresponding author.
E-mail addresses: charles.wade@colorado.edu (C. Wade), maccurdy@colorado.edu (R. MacCurdy).

https://doi.org/10.1016/j.addma.2023.103912
Received 29 August 2023; Received in revised form 27 November 2023; Accepted 4 December 2023
Available online 5 December 2023

the unique capabilities that OpenVCAD brings to multi-material design. We first explore recent advancements in multi-material additive manufacturing hardware and discuss existing applications of the technology. Next, we present related work on computational design methods employed in rendering and AM workflows and document their limitations in the application domains. Then, we detail our proposal of a multi-material design compiler, OpenVCAD. We demonstrate the performance of OpenVCAD as a workflow for multi-material additive manufacturing design and explore how it scales on modern large-volume and high resolution 3D printing systems. Next, we introduce a suite of objects designed and compiled using OpenVCAD to highlight how it addresses the demands of modern multi-material applications. Finally, we conclude with a discussion of potential improvements and give suggestions for applications of OpenVCAD.

2. Related work

OpenVCAD builds on the foundations of both single-material homogeneous and multi-material heterogeneous design to provide a framework for multi-material AM. We focus on creating a cohesive 3D-printing workflow to fully leverage recent advancements in multi-material hardware and applications. In this section, we discuss critical advancements in additive manufacturing hardware that have created a greater need for multi-material design tools and make OpenVCAD timely and necessary. We will present several application areas where advances in multi-material AM design are vital for enabling future work. Finally, we will discuss related methods for doing both homogeneous and heterogeneous design for additive manufacturing as well as related work in computer graphics and rendering.

2.1. Multi-material additive manufacturing technology

The widespread adoption of 3D-printing (3DP) necessitates a paradigm shift in the way engineers design objects. Without constraints of traditional subtractive fabrication, additive manufacturing enables greater flexibility in the geometry of engineered objects. Consequently, there has been extensive development of deposition modalities and their expansion to multiple materials. Common deposition modalities that allow multiple materials include Fused Filament Fabrication (FFF), Vat Photopolymerization or Stereolithography (SLA), Direct Ink Write, Inkjet, Binder Jetting, and Powder Bed Fusion [3]. Multi-material machines have seen improvement in two technologies: printers with multiple simultaneous deposition points, and diverse build materials with unique properties (e.g. flexibility, conductivity, strength, etc.). The ability to interleave multiple materials has extended 3DP toward the production of full-color objects, and parts with graded material properties [4]. Researchers have also used multi-material 3DP to simulate biological tissue [5] and to create low-cost, open source prosthetics [6]. Recent innovations in multi-material AM methods for FFF, SLA, and Inket motivate the need for a comprehensive design workflow. We provide a brief overview of recent advances in the multi-material capabilities of each of these processes to highlight the gap between fabrication capability and design methods.

Fused Filament Fabrication and Direct Ink Write have seen many innovations with multi-material 3DP in the form of multiple extrusion points and filament multiplexers [7–9]. Both innovations allow for the alternating deposition of different materials within the same layer of a print. There are many commercially available two channel FFF printers on the market. Many of these machines use a single printer head with two material channels. On each layer the channels will take turns depositing material in particular regions of the print. Recent advancements have created systems that extend this method by using multiple discrete print heads that are interchanged to add greater numbers of material channels [7]. Similarly, multiplexing systems like Prusa Research's MMU3 allows for the material in a single deposition channel printer to be automatically changed during a print [9]. However, these setups are limited to a small number of material types, typically less than 6, and the overhead involved in switching material or print heads within a layer can dramatically increase print time, and decrease resolution.

Similar to FFF, Stereolithography (SLA) has seen advancements in its capability to produce multi-material designs. With the development of multi-vat printers, there has been recent work in the functional grading of parts [10,11]. This method has fabricated robots with piezoelectric actuation and multi-material hydrogels [12,13], as well as electromagnetic devices consisting of materials with differing dielectric constants [14].

Material jetted 3D-printing, also known by its commercial trade names Polyjet [15] or MultiJet Modeling [16], uses inkjet technology to deposit micro-scale droplets of material. By using multiple fluid channels, it also offers rapid transitions between materials during a print and higher combined throughput and precision compared to other AM approaches, typically providing a large (>30 liters) build volume with feature sizes below 150 μm [17]. Inkjetting offers the unique capability of enabling materials with widely varying properties to form composites with visual and mechanical gradients [18], active parts including semiconductors and electrical circuits [19–21], and functional "print-in-place" objects requiring no additional assembly [22–24]. This paper will focus its methods on inkjet printing because of the superior deposition control it currently provides, however a section detailing applications to other AM practices is provided at the end.

2.2. Applications of multi-material additive manufacturing

Multi-material AM methods suggest a path toward integrated manufacturing of entire systems in which the precise material distribution determines the functionality. Recent work in "material intelligence" has attempted to imbue macroscopic materials with functional properties. For example, multi-material AM techniques were employed to produce lattice-based, meta-material objects that exploit both geometry and material distribution [25] to achieve extreme performance. In another example, the critical stress and strain in a soft material could be independently adjusted by controlling the material distribution [26]. Further, the complete functional mechanism of a door latch was realized by creating specific architected sub-domains within an overall structure [27]. The same authors showed a variety of other compliant structures, for which they created a custom design-automation editor specific to the compliant structures that were the focus of the study. More broadly, the application of two or more material components characterized by continuous gradual changes in material distribution is known as Functionally Graded Materials (FGMs) [2]. Advancements in AM hardware have made it possible to fabricate FGMs with ease, leading to FGMs becoming a driving force behind the development of multi-material heterogeneous design methods [28,29]. As such, a heterogeneous design method for AM should include features that enable the easy and efficient expression of FGMs.

Besides fabricating FGMs, another major benefit of AM techniques is the simplification of assemblies. Emergent methods for multi-material fabrication have facilitated hybrid electronics - the co-printing of structural material alongside conductive structures [30]. There has been development in printable strain sensors for monitoring structural health by printing with both non-toxic and conductive polylactic acid filaments [31]. In their survey paper on 3D-printed electronics, Persad et al. stated that "there is still room for the development and integration of software tools for the design and modeling of antennae and sensor structures which are intended to be manufactured using these 3DP technologies. [32]" As such, a robust heterogeneous AM design method should interface with simulation software.

Another emerging area of multi-material AM fabrication is printed robotics. Recent work in multi-material inkjet fabrication has shown

that co-printing solids and liquids is possible [33]. This allows for complex fluidic designs that other methods of manufacturing either cannot produce or would require onerous steps to assemble. This previous work demonstrated the complete fabrication of a hexapod robot, fully-automating the production in a single multi-material printing step [34]. Additionally, there has been further work detailing the liquid–solid fabrication method and providing examples of other structures that could be fabricated in this manner. Heterogeneous design is necessary to facilitate the expression of both solid and liquid regions in these fabricated objects. Related work in liquid and solid co-printing on inkjet systems has shown that viscoelastic behavior can be tailored by intermixing fluid and polymer during the printing process [35,36]. MacCurdy and Lipton presented a method to architect impact and vibration absorbing meta-materials for integration into robots using an algorithmic approach [35]. Their work demonstrated that material concentration can be engineered to yield architected viscoelastic behavior. Lipton expanded on this work by employing liquid and solid co-printing to embed fluid regions in impact absorbing closed-cell foams [36]. These applications rely on fitting numerical models using mechanical testing and characterization to establish a relationship between material concentrations and mechanical behavior. The characterization of these printed materials results in a mathematical expression that can be used by designers to control the mechanical behavior of their part. As such, a heterogeneous design method should allow designers to express material concentrations as spatially parametric expressions.

Medical image printing and pre-surgical planning models are another area of particular interest for heterogeneous multi-material design. Recent work has shown that medical images can be processed for and printed on inkjet systems [37,38]. These works include printing both visually and mechanically realistic pre-surgical planning models. Jacobson presented a workflow for performing bitmap-based printing of pre-surgical planning models using voxels [39]. However, one key problem that remains is creating realistic tissue analogs. Related works have shown that meta-materials can be architected with desired material properties on inket systems [35,36]. Similar work by Doubrovski et al. has shown that medical images can be used to design multi-material heterogeneous prosthetic objects [40]. As such, a robust heterogeneous design method should support medical scan data as input and provide image processing utilities such as thresholding and convolution to support the processing of raw data into realistic pre-surgical planning models. Furthermore, a method for processing medical images should include the ability to interface scan data geometry and materials with engineered objects, such as prosthetics, to yield custom-fit orthotics.

2.3. Homogeneous multi-material design

By far the most common method of multi-material design employed by designers today is homogeneous solid body modeling. In this process multi-material objects are expressed as multiple discrete solid bodies which are each assigned a material channel before printing. This multi-material AM work reflects the underlying geometric representation employed by modern CAD software such as SolidWorks or Fusion360. These CAD programs use a mixture of CSG and b-reps to visualize and create solid bodies. A b-rep is a solid whose surface has been subdivided into a collection of cells made up of vertices, edges, and faces. Further, it is a collection of orientable surfaces bounded and connected without self-intersection to form an object that conveys topological information [41]. Similarly, traditional CSG defines a solid only by its surface. A GPU-accelerated CSG slicer, known as IceSL,[1] was proposed by Lefebvre et al. that can do multi-material homogeneous design [42]. CSG was created to visualize 3D-objects and intended to be used with ray tracing to determine where external surfaces should be displayed on screen [43]. Thus, CSG and b-rep approaches are limited in the expression of heterogeneous material distributions. As an example, when using b-rep, each model can only consist of a single material that is solid throughout the entirety of the bounded surface [44]. Practically, this dictates that for every independent volume and material type, there must be a separate boundary. When representing the continuous material distributions that are common in FGMs, sampling and thresholding is necessary to separate the continuous material into discrete assignable regions. As the number of threshold regions increases, so does the number of required b-reps and corresponding computational cost of toolpath planning. Additionally, this threshold approach creates sharp material transition zones that can lead to structural weaknesses [45]. While solid body modeling has historically been sufficient for subtractive manufacturing processing, surface representations lack the volumetric information required for emerging multi-material AM methods.

2.3.1. Functional geometry representations

One common problem faced in computer aided design is the compactness of expressing complex geometry. Likewise, many AM CAD programs rely on verbose and approximate geometry representations like triangulated meshes. These b-rep files have two primary disadvantages: file size and approximation. B-reps typically also require interactive design software to define objects. This has inhibited b-rep adoption in simulation and optimization workflows because manual design iteration is required by the user after each step. Pasko et al. proposed an alternative to b-rep called functional representation (f-rep) [46]. F-reps are composed of functions in the form $f(x) \geq 0$. The surface of an object is the set of points such that $f(x) = 0$. A point in space that evaluates to <0 is outside of the object and evaluates to >0 if the point is inside of the object. The function, $f(x)$, is viewed as a "black box" and can contain math expressions or parametric code. However, $f(x)$ must have parameters for spatial sampling. Furthermore, f-rep objects can be combined using union, intersection, and subtraction operators. Since all operations in F-reps can be defined analytically, they are particularly well suited for applications that required a high degree of programmatic parameterization such as optimization [47]. F-reps have also been effective in modeling volumetric lattice structures [48]. Additionally, f-reps have been extended to not only represent geometry, but also describe volumetric features of an object such as opacity or material distribution [49]. We will discuss this application and how it pertains to heterogeneous design in more detail in the next section.

2.4. Heterogeneous multi-material design

We will classify related heterogeneous design methods into two broad categories: those directly applicable to AM workflows, and those which are more broadly intended for computer graphics and rendering. We classify a heterogeneous design method as applicable to AM if it directly address its application to the slicing and preparation of print files for a multi-material AM process. At the conclusion of these sections we provide a table that synthesizes the current state of heterogeneous design methods and highlights the shortcomings that we address with our proposed method.

2.4.1. Heterogeneous design for graphics and rendering

Heterogeneous design is a common problem faced by the graphics and rendering community. Typical applications of heterogeneous design involve the expression and rendering of volumetric data such as medical images, water, clouds, and fire [50]. This data is routinely expressed as 3D voxel grids that store information on material concentrations, opacity and density. Two of the most common methods for rendering voxel data are texture slicing and ray-tracing. Texture slicing involves sampling the heterogeneous design at regular intervals into texture memory on a graphics processing unit (GPU) using data-slice polygons [51]. By rending the slices from farthest to nearest, a

[1] https://icesl.loria.fr/.

translucent object can be rendered. Similarly, ray-tracing can be done by first sampling a heterogeneous design into voxels. Rays are then cast from a camera though the voxel-grid to yield a render of the design. Both of these methods rely on heterogeneous designs stored as voxels grids, requiring $O(n^3)$ space complexity. The build volume and resolution of modern inkjet systems such as the Stratasys J750 yield designs with up to 396 billion voxels, presenting a computational barrier for using traditional voxel design tools with modern AM.

Museth proposed OpenVDB, a sparse-voxel data structure that can improve the memory footprint [50]. OpenVDB provides a suite of design tools for manipulating voxel design such as Boolean operations and affine transforms. Museth improved on the CPU-only limitation in OpenVDB by providing a GPU-native sparse-voxel data structure with NanoVDB [52]. However this massively parallel version does not include many of the design tools that were provided with the original OpenVDB library. Although OpenVDB and NanoVDB can provide an efficient representation for heterogeneous models that contain large sparse regions, designs that contain random material distributions, as is common with stochastic sampling used in inkjet printing, diminish the efficacy of the sparse data structure.

A key limitation of voxel data structures is their discrete sampled nature. Similar to how f-rep geometry has a smaller storage footprint compared to the same geometry sampled as a triangulated mesh, functional representation for materials is more efficient than voxels. Pasko et al. proposed a method that uses n user-defined scalar fields, coupled with a geometry primitive, to represent geometry and other attributes such as density or opacity of an object [49] called *Constructive hypervolume modeling*. Using Pasko's method, a designer can combine and modify multiple objects, each with their own multidimensional attribute sets, using operators in a tree. In this structure, object primitives are in the leaves and operations in the nodes. Ambiguities in how min/max operations were applied to heterogeneous material distribution attributes in Pasko's original work were later solved by Fayolle & Pasko using signed approximate real distance functions [53]. Pasko's method uses a special constructive tree similar in structure to a Blob-Tree [54–56]. Pasko's constructive tree is a hierarchical structure where implicit surfaces are leaves and operations (blending, warping, and Booleans) are nodes. When attributes are defined in non-leaf nodes of a constructive tree, values override attributes lower down the hierarchy. *Constructive hypervolume modeling* is a theoretical framework that was applied to heterogeneous texturing of rendered models and lacks the necessary methods for multi-material design for AM systems. One key feature that is missing is a method to sample the continuous f-reps that define material distributions into discrete voxel print formats such a bitmap stacks. Additionally, *Constructive hypervolume modeling* defines attributes as a high level concept. Although this definition allows for attributes representing many different features such as opacity, density and temperature, their focus on abstraction lacks a method to ensure a valid volume-fraction is defined everywhere in $\mathbb{R}^3$ when modeling material distributions.

2.4.2. Heterogeneous design methods for AM

OpenVCAD builds on insights from several existing heterogeneous design methods specifically targeted at AM. VoxelFuse is a tool for design and fabrication of functionally graded materials and solid bodies, and incorporates Brauer and Aukes's voxel-based CAD framework [57, 58]. The framework allows for modeling with multiple materials by storing CSG-constructed solid bodies and material concentrations as voxels. Voxels allow for the simultaneous definition of shape and material. Additionally, the voxel-based framework allows users to define keep-out, clearance, web, and support regions that simplify design processes specific to the intended method of manufacturing. Furthermore, VoxelFuse incorporates a voxelization engine and simulation capabilities. The voxelization engine can convert mesh files into voxel representation and allows for the mixed use of VoxelFuse with other common CAD tools that produce mesh files, such as OpenSCAD, SolidWorks, or Fusion360. The included simulation capabilities are based on the VoxCAD and Voxelyze frameworks [59]. While VoxelFuse offers a variety of features for designing printed FGMs, its primary drawback is its reliance on a voxel-backed data-structure. VoxelFuse's reliance on a dense voxel data structure prevents it from scaling to modern inkjet volumes. In VoxelFuse, operations are $O(n^3)$ with the number of voxels, limiting the size and complexity of objects that can be designed.

Foundry and OpenFab are design tools that enable multi-material volumetric design [60,61]. OpenFab focuses on using multiple materials of varying color for rapid manufacturing and has been applied to both robotics and aesthetic design. OpenFab uses a two stage approach starting with a boundary-surface phase as input, followed by a volume-definition phase. In both phases, textures and "fablets" are applied to render a graphic across the constructed geometry. Shapes are created by using a scene graph describing a set of object b-reps that is then tessellated and eventually voxelized before being dithered and exported as a final output. Foundry boasts a more polished user interface, but also focuses on texture rendering and synthesis. The premise of Foundry is to modify the material composition of existing geometries. This tool allows the user to create alloyed or functionally graded materials, but requires the user to generate the geometry using a separate tool.

A related tool, GraMMaCAD, allows a user to import an existing boundary-surface definition and interactively define material gradients in various regions of the part [62,63]. While intuitive to use, interactive tools that import existing solid-body geometry become tedious to use when many sub-regions of the part need to be independently identified and addressed by the user. These complex geometric features are common in many applications including soft robotics, functional meta-materials, and hybrid electronics. Instead, design tools that natively operate on volumes, and which allow programmatic design via scripting are necessary to allow the creation of complex multi-material objects.

Elber et al. explore the evolution of geometric modeling tools, emphasizing the shift from Non-Uniform Rational B-splines (NURBs) to a B-spline based volumetric representation (V-rep) [64]. Their paper discusses how V-reps can support the creation of porous, heterogeneous, and anisotropic objects. They argue that V-reps allow for a tighter integration between multi-material design and finite element analysis, facilitating the manufacturing of functionally graded materials and geometries. The source code for Elber's work is available as part of the IRIT Modeling Environment.[2]

2.4.3. Current challenges for heterogeneous AM design

In their review paper titled "Modeling and Visualization of Multi-material Volumes", Fayolle et al. stated the following [2]: "Currently, two extreme cases are common in the research literature; it is either an object with simple geometry (e.g., cube or cylinder) with a complex material distribution or a complex object with simple material composition. However, the ultimate goal in this research direction is to provide methods and tools of defining complex, shape conforming material distributions for objects with complex geometry." Achieving both geometric and material complexity has been an ongoing challenge in the heterogeneous design space. Volumetric objects can be space inefficient, hard to express, and incompatible with existing AM workflows. Fig. 1 shows a summary of the existing homogeneous and heterogeneous design methods and the features they support. These features include items identified by Fayolle et al. as critical for enabling simultaneous complexity of object geometry and material distribution [2]. Additionally, we include relevant features such as geometry types supported, medical imaging utilities, and methods that enable direct application to multi-material AM systems. To the authors'

[2] https://gershon.cs.technion.ac.il/irit/.

		Traditional Reps		Geometry Compilers		Voxel Design		FGM Design		Medical Image		Hybrid Methods	
		B-reps[a]	F-reps[b]	Hyperfun[c]	OpenSCAD[d]	OpenVDB/ NanoVDB[e]	VoxelFuse[f]	GraMMaCAD[g]	OpenFab/ Foundry[h]	Jacobson[i]	Doubrovski[j]	Hypervolume[k]	OpenVCAD
Geometry & File Features	Constructive Solid Geometry	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes
	Functional Geometry	No	Yes	Yes	No	Yes	No	No	No	No	No	Yes	Yes
	Surface Geometry	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
	Implicit Curvature	Yes	Yes	Yes	Yes	No	No	Yes	No	No	Yes	Yes	Yes
	Affine Invariant	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes
	Tree structure	No	No	Yes	Yes	No	No	No	No	No	No	Yes	Yes
	Exact Geometry	Yes	Yes	Yes	Yes	No	No	Yes	No	No	No	Yes	Yes
Multi-material Design	Homogeneous Design	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
	Heterogeneous Design	No	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
	Explicit Material Distribution	NA	NA	NA	NA	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
	Implicit Material Distribution	NA	NA	NA	NA	No	No	Yes	Yes	No	No	Yes	Yes
	Spatial Parameterization of Materials	NA	NA	NA	NA	No	Yes	Yes	Yes	No	No	Yes	Yes
	Functionally Graded Materials	NA	NA	NA	NA	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Exchange Formats	Provides Realtime Visualization	NA	NA	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
	Forward/ Reverse Integration for Optimization	NA	NA	No	No	No	No	No	No	No	No	No	Yes
	Direct Simulation Output	NA	NA	No	No	No	Yes	No	No	No	No	No	Yes
	Source Code Available	Yes	Yes	No	Yes	Yes	Yes	No	No	No	No	No	Yes
	Compact/ Portable File	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes
Medical Imaging	Medical Image Input	NA	NA	No	No	No	No	No	No	Yes	Yes	No	Yes
	Image Processing	NA	NA	No	No	No	No	No	No	No	No	No	Yes
Multi-Material AM Features	Dithering	NA	NA	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes
	Bitmap Exporting	NA	NA	No	No	Yes	No	Yes	Yes	Yes	Yes	No	Yes
	Asymmetric Voxel Sampling	NA	NA	NA	NA	No	No	Yes	Yes	Yes	Yes	NA	Yes
	Interactive Speeds at Inkjet Scale (10^9 voxels)	NA	NA	NA	NA	Yes	No	Yes	Yes	Yes	Yes	NA	Yes

Fig. 1. Comparison of single and multi-material computer aided design methods. (a) *Open CASCADE* [65]. (b) *Function representation in geometric modeling: concepts, implementation and applications* [46]. (c) *HyperFun project: a framework for collaborative multidimensional F-rep modeling* [66]. (d) *OpenSCAD- The Programmers Solid 3D CAD Modeller* [67]. (e) Collective work of Museth et al. [50,52]. (f) Collective work of Brauer and Aukes [57,68]. (g) *GraMMaCAD: Interactively Defining Spatially Varying FGMs on BRep CAD Models* [62]. (h) Collective work of Vidimče et al. [60,61]. (i) Collective work of Jacobson et al. [37–39]. (j) *Voxel-based fabrication through material property mapping: A design method for bitmap printing* [40]. (k) *Constructive hypervolume modeling* and its derivative work [49,53].

knowledge, OpenVCAD is the only available method that can address this range of needs.

Several methods depicted in Fig. 1 illustrate a lineage of development in computer-aided 3D-design. For instance, *Hypervolume* is developed upon *Hyperfun*, which in turn is based on *F-reps*. Our intent with the introduction (and Fig. 1) is to highlight the historical progression of these methods, showcasing the progress in computer-aided design, while pointing out that no single tool implements the suite of capabilities that we think are currently necessary. We highlight the progression by referencing both foundational and derivative works.

Additionally, this approach provides a contextual framework for understanding related developments that diverged from these intermediate methodologies. For example, *OpenSCAD* extends upon *Hyperfun's* foundation but does not incorporate the heterogeneous design functionalities featured in the *Hypervolume* extension. Similarly, by comparing geometric representations like *B-reps* and *F-reps* with software tools, we demonstrate how different implementations selectively incorporate elements from each representation. This is relevant because our goal in this paper, and for OpenVCAD, was to integrate and extend ideas from the past 40 years of computer-aided 3D design that we think are useful to users of modern volumetric multi-material 3D printers.

2.5. Geometry compilers

OpenVCAD is a multi-material geometry compiler. Therefore it is necessary to discuss related work on geometry compilers. Boundary representations use primitive geometry types such as points, lines and curves combined with basic CSG-like operations to form objects. Similarly, volumetric design routinely employs a single primitive, the voxel, to express geometry. These geometry representations are usually created using interactive CAD software. Parametric features are enforced by the CAD system and can only be modified using a graphical user interface, limiting the complexity of highly parametric objects. To address these concerns, attempts have been made to define geometry and heterogeneous material distributions using a compiled approach. In the geometry compiler paradigm, the designer expresses an object using a specific programming language. Various keywords, functions, variables, and loops are used to define the object's geometry and attributes. The text is passed to a compiler that converts it into another desired format such as a triangulated mesh or render.

One of the earliest geometry compilers was proposed and released as an open source project by Pasko [66]. Known as Hyperfun, this

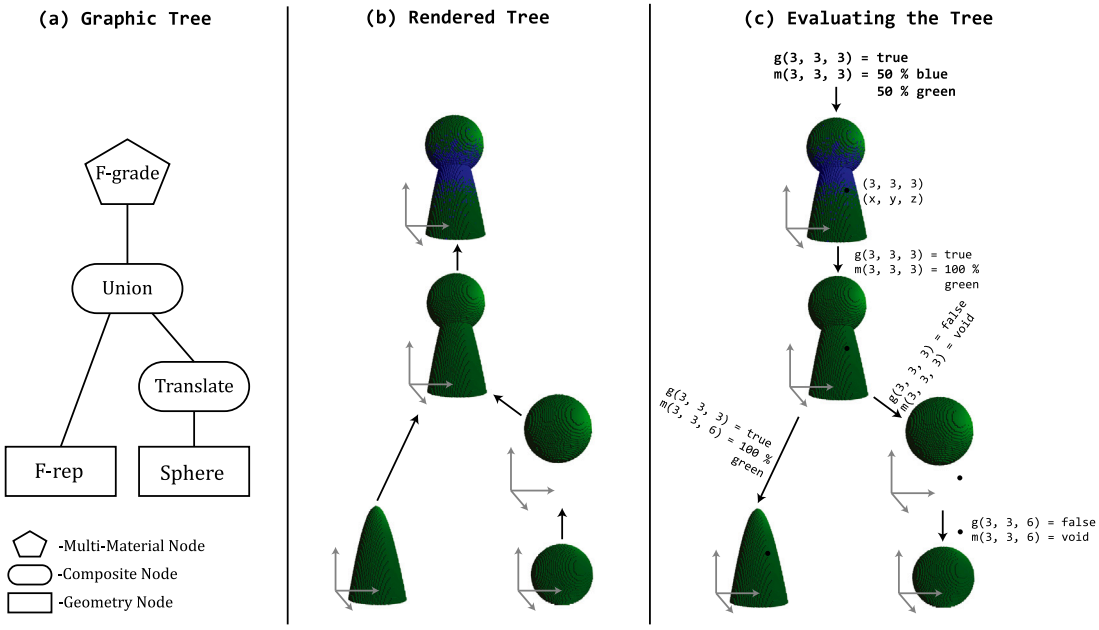


Fig. 2. An OpenVCAD tree that builds an object using mixed geometry and materials. The tree uses a functionally defined cone for the base, a sphere for the top, and functional grading to mix materials.

tool used F-rep in conjunction with a special programming language to define highly parametric objects. The Hyperfun language provided several different geometric operations like union, difference, intersection, as well as math function like `sqrt`, `exp`, `log`, and `sin`. An extension of Hyperfun into heterogeneous design was done by Pasko in "Constructive hypervolume modeling" [49] and later expanded by Fayolle [53]. Hyperfun was also extended to support functional microstructure design and 3D printing [48].

Similarly, OpenSCAD is a programming-oriented, parametric design tool for creating solid bodies from boundary surface representations. Users define geometry by writing a script with OpenSCAD's program-specific language. OpenSCAD serves as the compiler that processes the scripts as CSG trees and renders the 3D object. CSG trees represent Boolean combinations of primitive shapes (e.g. cubes, spheres, and cylinders) to describe more complex geometries. While OpenSCAD is a robust and accessible tool for 3D modeling, it retains the limitations of the underlying boundary surface representations. To represent objects with multi-materials using OpenSCAD, the designer must export each discrete material region as a single body; designs with regions of multi-material graded or densely interdigitated material distributions at realistic resolutions require unreasonable file sizes and compile times. This constraint hinders OpenSCAD's adoption in multi-material 3D printing workflows.

3. Volumetric multi-material design for AM

To leverage advances in multi-material additive manufacturing, designers need access to easy and intuitive design methods. Toward this goal, we propose OpenVACD, an implicit geometry scripting language and compiler for volumetric representations. Our scripting interface enables a hybrid design approach, combining f-rep geometry with a user-definable vector field of material distributions and specific operations relevant to AM. The OpenVCAD language consists of geometric primitives and combinatory operations to describe complex 3D shapes made of multiple complex material distributions. Designers can use familiar scripting components, such as keywords, functions, and braces, to express a hierarchy of primitives and combinatory operations. Leveraging the advantages provided by a geometric compiler paradigm, our scriptable approach allows for parametric designs that can be configured for application specific needs. Furthermore, many of the application domains of OpenVCAD require the modularity provided by a scriptable geometry approach. For instance, with a scripted approach, forward and reverse integration of parameters for optimization of material distribution and geometric features is possible. This can be achieved by first assigning variables to parameterize geometry and material distribution in the design script. A method to compile VCAD designs into a volumetric meshes for finite element analysis (FEA) is discussed later. The FEA mesh can then be simulated for multiple desired mechanical or material behaviors and fitness functions derived. The fitness functions can then be used in a heuristic optimization method such as NSGA II [69] to determine new variables for future design generations.

3.1. Tree definition

OpenVCAD scripts are parsed into a tree structure that acts as a blueprint for the object to be designed. Leaf nodes express geometric features and optionally material distribution. Leaf nodes are specified with various keywords within the scripting language. Parent nodes can be either composite nodes that define transformations or combinations of geometric primitives, or multi-material nodes that define material distributions. In the scripting language, parent–child relationships are denoted with functions and braces. The tree structure enables efficient look-up for fast sampling of the composite design in $\mathbb{R}^3$. As a result, all nodes in the OpenVCAD tree are of the form $Node(x, y, z)$, where the geometry and material (or absence of) at any point in space can be accessed using Cartesian coordinates. Fig. 2 shows an object expressed using the OpenVCAD tree. Geometry is defined as leaves in the tree using the `f-rep()` node to define a cone and `sphere()` node. A composition node is then used at the parent level to translate the sphere upwards. Furthermore, a `union()` node is applied to combine the discrete geometries into a single object. Finally, a functional gradient is applied across the object to blend two materials.

Geometry and material distribution are sampled independently in OpenVCAD. Geometry is sampled in the form $\mathbf{g}(x, y, z)$, where values <0 correspond to inside, $=0$ are on, and >0 is outside of the surface. Similarly, let $\mathbf{m} : \mathbb{R}^3 \rightarrow \mathbb{R}^n$ be a function representing material distribution. For any point $(x, y, z) \in \mathbb{R}^3$, the function $\mathbf{m}(x, y, z)$ returns

Table 1
Summary of how nodes in the OpenVCAD affect the evaluation of the geometry $\mathbf{g}(x, y, z)$ and material $\mathbf{m}(x, y, z)$ spatial functions. These nodes are not an exhaustive list of the nodes available in OpenVCAD, but are representative of the set of nodes that perform functional material representation, "boolean" operations, affine transforms, functional geometry representation, and primitive geometries, respectively from top to bottom in the table.

Node	Node parameters	$\mathbf{g}(x, y, z)$	$\mathbf{m}(x, y, z)$
`F-grade`	$m_{\text{new}}(x, y, z)$	$g\_child(x, y, z)$	$m_{\text{new}}(x, y, z)$
`Union`	NA	$\min(g_{\text{child\_a}}(x, y, z), g_{\text{child\_b}}(x, y, z))$	normalized sum of $m_{\text{child\_a}}$ and $m_{\text{child\_b}}$ OR specified $m_{\text{child\_x}}$
`Translate`	$(\Delta_x, \Delta_y, \Delta_z)$	$g_{\text{child}}(x-\Delta_x, y-\Delta_y, z-\Delta_z)$	$m_{\text{child}}(x-\Delta_x, y-\Delta_y, z-\Delta_z)$
`F-rep`	$f(x, y, z)$, specified material	$f(x, y, z)$	100% specified material
`Sphere`	Radius, specified material	$x^2+y^2+z^2-R^2 \le 0$	100% specified material

Table 2
Time complexity of evaluating individual nodes in OpenVCAD. For leaf nodes, an individual node refers to a tree with only that node in it. For operator nodes, `convolve()` and `f-grade()`, the `sphere()` node is used as the only child, forming a two-node tree.

Node	Time complexity	Where
Sphere	$O(1)$	–
Cube	$O(1)$	–
Mesh	$O(1)$	–
Convolve	$O(k_x \times k_y \times k_z)$	$(k_x \times k_y \times k_z)$ is the size of the kernel
F-grade	$O(n)$	n is the number of tokens in the largest equation

a vector in $\mathbb{R}^n$ representing the volume fractions at that point. Implicit in this definition is that everywhere $\mathbf{m}(x, y, z)$ is defined, $\mathbf{g}(x, y, z)$ must also be defined. Evaluation of a tree starts at the root node and in-order traversal is performed recursively on children. Fig. 2(c) and Table 1 detail how various nodes affect the evaluation as the traversal is performed. Starting at the root and evaluating $\mathbf{m}(x, y, z)$ the `F-grade` node will replace the child material distribution with a new distribution by evaluating functions supplied as parameters when the node was created. `F-grade` does not affect geometry, so the result of calling $\mathbf{g}(x, y, z)$ on the child is returned. Moving to the second level down, a `union` is being performed between `child_a` and `child_b`. The Union node is defined with two child nodes. Initially, we calculate $\alpha = \mathbf{g}_{\text{left}}(x, y, z)$ and $\beta = \mathbf{g}_{\text{right}}(x, y, z)$. Following this, the union of these two values is determined by computing $\min(\alpha, \beta)$, which selects the minimum of the two computed values. $\mathbf{m}(x, y, z)$ for the `union` node first performs component wise addition between the volume-fraction vectors returned by calling $m_{\text{child\_x}}$ on each child. The resultant vector is then normalized to ensure components sum to one (alternatively, there is an option to simply return the $\mathbf{m}(x, y, z)$ from one of the children). Following the left hand sub-tree rooted at the `union` node, an `f-rep` leaf node is reached. This node evaluates a user-supplied signed-distance function, $f(x, y, z)$, for $\mathbf{g}(x, y, z)$. The `f-rep` node always returns 100% concentration of a user-specified material for $\mathbf{m}(x, y, z)$. The right hand sub-tree rooted at the `union` leads to a `translate` node. When defining a `translate` node, the designer species translation values, $(\Delta_x, \Delta_y, \Delta_z)$. These values are subtracted from the sampled point (x, y, z) to yield new transformed coordinate space. Finally, following the child of the `translate` node, we reach a `sphere` leaf node. The `sphere` node is a CSG primitive that evaluates the sphere equation $x^2+y^2+z^2-R^2 \le 0$ for $\mathbf{g}(x, y, z)$. Identically to the `f-rep` node, the `sphere` node always returns 100% concentration of a default material for $\mathbf{m}(x, y, z)$ (see Table 1).

3.1.1. Leaf nodes: Geometric primitives

OpenVCAD supports two types of leaf nodes for defining geometric features: (1) CSG primitives and (2) implicit geometry in $\mathbb{R}^3$. CSG primitives include cuboids, cylinders and spheres, using `cube()`, `cylinder()`, and `sphere()` syntax respectively. Alternatively, designers can express implicit geometry using closed-form expressions in the form $f(x, y, z)$. When $\mathbb{R}^3$ is sampled, a point is considered inside a 3D shape if the evaluation of $f(x, y, z)$ returns a number less than zero. Likewise, the point lies on the surface of the object if $f(x, y, z)$ is exactly zero. Designers supply the implicit expression as a parameter of the f-rep node when writing their OpenVCAD script. The `f_rep()` node uses the Exprtk C++ library to parse the mathematical expressions into a real value [70]. This expression parser is capable of parsing many common math functions such as `exp`, `sin`, `pow` and `sqrt`, in addition to more complicated Boolean logic. When evaluating an expression, OpenVCAD provides the sampled point in cartesian, cylindrical, and spherical coordinates. As an example of an implicit geometry node, Eq. (1) shows the expression for a sphere of radius R.

$$x^2+y^2+z^2-R^2 \le 0 \tag{1}$$

3.1.2. Leaf nodes: External geometry

In some applications, it can be advantageous to use traditional CAD tools to express geometry and OpenVCAD to define material distribution. Consequently, OpenVCAD allows for the import of geometry through two auxiliary nodes: (1) triangulated meshes and (2) voxel data. The `mesh()` node allows the designer to import a triangulated mesh as a leaf node in an OpenVCAD tree. Before the node is sampled in $\mathbb{R}^3$, the triangulated mesh is preprocessed into a signed distance field stored within a sparse voxel grid using the method described by Museth [50].

Similar to `mesh()` node, the `voxel()` node provides an interface for designers to import existing voxel data into an OpenVCAD tree. The `voxel()` node preprocesses bitmap stacks into a 3D sparse voxel grid that allows for constant time access of discrete material values in $\mathbb{R}^3$. As an additional step to preprocessing, the `voxel()` node remaps asymmetric input data to uniform voxels. This is a critical component because many volumetric inputs, such as medical imagery, sample using asymmetric resolutions and need to be remapped to the voxel size of the target printer. The voxel input workflow is useful for volumetric 3D printing of presurgical planning models. In this application space, medical imaging data is processed into stacks of bitmap files that represent a discrete voxel space. The bitmaps are often printed with materials that emulate the color of the anatomic object they represent. OpenVCAD's multi-material design toolkit is well suited to perform 3D processing on these medical models to reduce noise, highlight features, and modify material distribution. For example, Fig. 3 shows cross-sectional slices of a 3D printed brain. When printed with transparent materials, the object can provide a practitioner with a handheld 3D visualization to aid in presurgical planning. However, the original scan data shown in 3(a) contains noise that obscures the view of interior target regions shown in pink. Fig. 3(b) shows OpenVCAD's discrete convolution node (discussed below) being applied as a sharpen operation in $\mathbb{R}^3$ to improve the transparency of the green region by reducing the noise of the image. Furthermore, OpenVCAD can be used to map discrete and continuous regions to meta-materials that simulate the mechanical behavior of specific tissue.

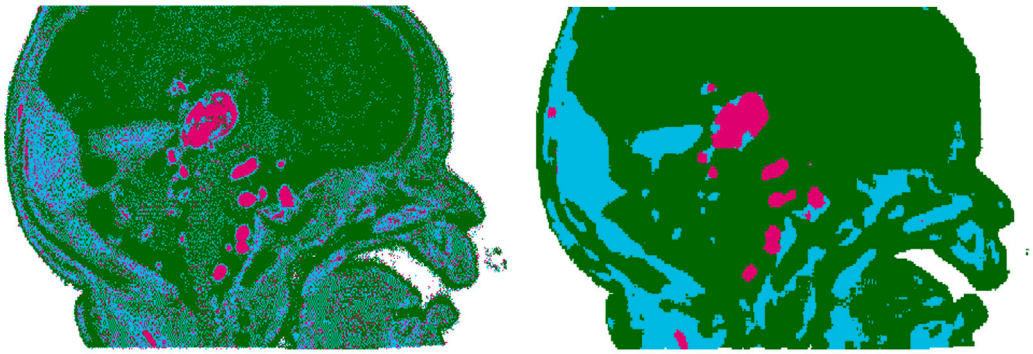


Fig. 3. (a) The voxel geometry node is used to import volumetric medical imagery for processing. (b) A sharpening kernel is used with OpenVCAD's 3D convolution node to reduce noise in the model and improve transparency. Although these images show a single slice for visualization, the image processing occurs in $\mathbb{R}^3$. We thank Nicholas Jacobson from CU Anschutz for providing the medical images. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

```
Union{
  Rotate(axis = x, deg = 45){
    Cube(s = 5, material="red")
  }
  Translate(x = 5, y = 10, z = 5){
    Sphere(r = 3, material="blue")
  }
}
```

Fig. 4. OpenVCAD code that shows two geometric primitives being modified by translation, rotation, and union composite nodes. The overlapping red and blue regions will form a new purple region.

3.1.3. Composite nodes

Composite nodes are critical for describing complex shapes. These nodes define transformations and combinations of both geometric primitives and any valid subtree. Transformations include `translate()`, `rotate()`, and `scale()`. Combinations refer to the boolean operations of `negate()`, `union()`, `difference()`, and `intersection()`. When two nodes with different materials are combined through a union operation, their intersecting material distributions are summed and then normalized to yield a blended region, or the designer can require that the material from one node take precedence. Other material transitions can be defined with multi-material nodes, discussed in the next section. Fig. 4 shows the code format the designer uses to define composition and geometric primitive nodes in the OpenVCAD language.

3.1.4. Multi-material nodes: Functional grading

Multi-material nodes represent OpenVCAD's unique ability to express heterogeneous volumes. One such multi-material node is the functional grading (FG) node. FG nodes enable smooth material interfaces and precise control over the material properties of the designed components. FG nodes employ a user-defined 3D vector field to assign material values to existing geometry. The vector field is n-dimensional, where each component represents the volume fraction of a particular material. To create the material distribution, a user first defines multiple scalar field equations in $\mathbb{R}^3$ of the form $f\_grade(x, y, z)$. These expressions are independent, but spatially overlapping, volume fraction functions that each describe the concentration of a single associated material for all points in a 3D region of interest. Since each scalar field must be overlapping, we combine them component-wise into a single vector field. The volume fraction functions are normalized so that they sum to one for all points. Normalization is essential to guarantee that the evaluation of each volume fraction function accurately reflects the concentration of the corresponding material present at a specific point in space.

Although a continuous vector field distribution of materials provides a compact method for expressing heterogeneous objects, it is not directly compatible with AM systems that require voxels as input. Stochastic sampling is needed to convert from a continuous distribution of materials to discrete voxels. Stochastic sampling, sometimes referred to as stochastic dithering, is a common procedure used in both 2D and 3D inkjet printing to create the illusion of more colors than are present in material channels of the printer. Fig. 5 shows a comparison of stochastically sampled objects at different resolutions. Each object was graded across the x-axis with a blue and a red material using Eqs. (2) and (3) respectively.

$$P_{\text{red}} = \frac{1}{1 + e^{x}} \tag{2}$$

$$P_{\text{blue}} = -\frac{1}{1 + e^{x}} + 1 \tag{3}$$

OpenVCAD offers two distinct sampling modes: probability and threshold. In probability mode, the volume fraction functions are treated as Probability Density Functions (PDFs). This approach interprets a sampled point in the three-dimensional space, $\mathbb{R}^3$, through these PDFs. Consequently, each sampled point generates a probability vector, which indicates the likelihood of each material being present at that location. The final material assigned to a specific location (we stipulate that only one material can occupy a discrete location) is determined by a stochastic decision, which is weighted according to the calculated probability vector. Algorithm 1 shows how a continuous vector field of material distribution can be stochastically sampled into discrete voxels for printing. The stochasticity ensures that a distribution of materials

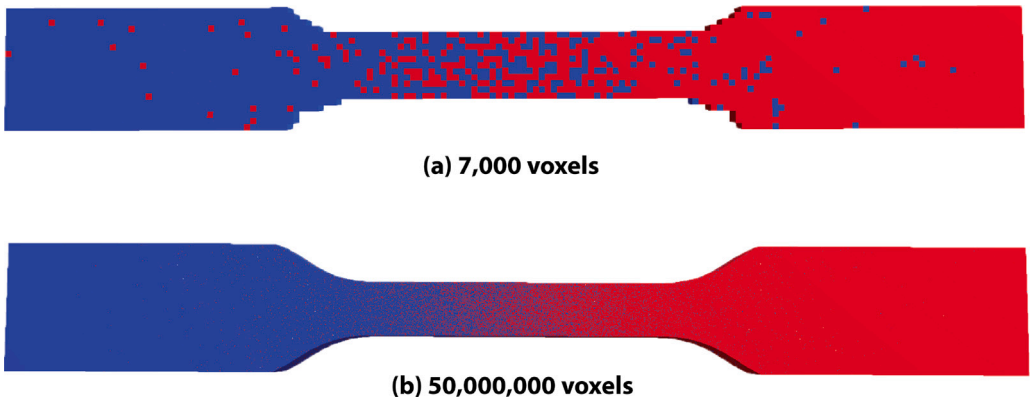


Fig. 5. The same object stochastically sampled into voxels at two different resolutions. As the voxel resolution approaches the scale of an inkjet system (10^9 voxels) the individual cell colors blend together. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

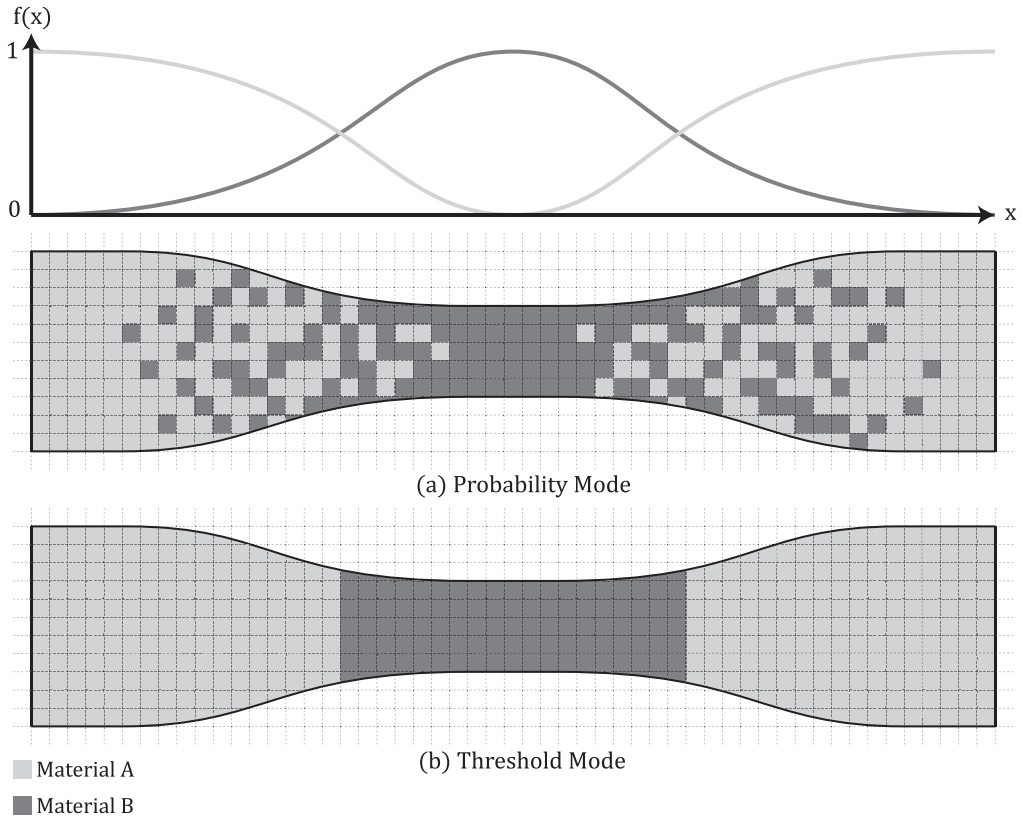


Fig. 6. $f\_grade(x, y, z)$ performed over an ASTM D412 Dogbone using two overlapping probability density functions in (a) probability mode and (b) threshold mode. In this example, a tensile strength validation geometry is enhanced with a stronger Material B in the thinner section of the part.

is present when performing functional grading. From Fig. 6(a) we see this mode allows for the creation of smooth transitions and "digital materials" by specifying the mixture ratio of multiple materials over a region.

In contrast, threshold mode allows for deterministic control over the material composition of a component. In this mode, users define the same field equations as volume fraction functions; however, the weighted randomization phase is replaced by selecting the material corresponding to the highest volume fraction. From Fig. 6(b) it can be observed that this mode generates sharp material transitions within geometry. While sharp material transitions can also be created using only boolean combinations of geometric primitives, thresholding FG nodes allow for more concise design trees and shorter files. For example, we designed the soft-actuator in Fig. 7 using functional grading. We used a single implicit geometry leaf node to generate the shape of the actuator, and an FG node in threshold mode to define regions of different material. In contrast, the design would require the union of 19 leaf nodes, one for each discrete region of a single material, if it were represented solely as Boolean combinations of primitives.

3.1.5. Multi-material nodes: Convolution

While functional grading provides a mechanism for defining material transition within geometry, it can be cumbersome for blending across multiple pre-existing material boundaries, as might occur with imported geometry. Fig. 8(a) shows an XY slice of a situation where a designer defined four squares with varying materials. The individual squares were combined with a `Union` operation to form a single, multi-material object. The designer wishes for the four material regions to be

Fig. 7. Soft actuator printed in three materials. This example was created using a scalar field equation in polar coordinates with a threshold of 1 to generate the geometry, and was then functionally graded in threshold mode using three distinct scalar fields over the entire bounding box of the model. The above image is the resulting model, printed using the Stratasys J750 InkJet Printer. The OpenVCAD script for this object is provided in Fig. A.19.

Algorithm 1 Weighted Stochastic Sampling Algorithm

Require: A vector field of material distribution PDFs defined in $\mathbb{R}^3$

```
sampled_object ← allocate memory to store material IDs for every sampled point
for each point in sample space ℝ³ do
    distribution vector ← value of vector field at point
    rnd ← a random float between 0 and 1
    for each component in distribution vector do
        weight ← component.value
        if rnd < weight then
            sampled_object[point] ← component.id
            break
        end if
        rnd ← rnd − component.value
    end for
end for
return sampled_object
```

blended at their boundaries to exhibit a smoother transition from one material to another. To accomplish this goal with functional grading, the designer would need to develop a set of complex probability density functions expressed in terms of `x` and `y`.

Instead, discrete convolution provides a simpler approach to smoothing material transitions. Convolution is a common operation in signal processing, computer vision, and machine learning used to perform many operations including blurring, sharpening, eroding, dilating and other image manipulations [71,72]. The convolution operation expresses how the shape of one function is modified by another. In image processing, convolution is done with an input image and 2D kernel. Each pixel of the image is sampled and replaced with the sum of its neighbors multiplied by the kernel weights. Using a specific kernel, convolution can be used to perform Gaussian blurring and sharping on an image.

OpenVCAD defines a node that can be used to perform discrete convolution over 3D volumetric data. As illustrated in Fig. 9, the OpenVCAD convolution operation first considers the distribution of materials in the kernel region. Analogous to 2D image convolution, an `N` by `M` by `K` 3D kernel expresses the weight that neighboring voxels have at a specific location. These weights are multiplied by the corresponding material counts at each location to yield a modified distribution. The material distribution is then used as probabilities for a stochastic decision on what the material at the center of the kernel should be replaced with. In Fig. 9, the process results in a 56% chance that material `A` and a 44% chance material `B` will be selected. The weighted stochastic process then selects and replaces the material at the center of the kernel. The process is repeated at all sampled locations in $\mathbb{R}^3$ using the original material values to create the mixed boundaries. Fig. 8(b) shows the result of performing the convolution operation over an object using a $10 \times 10 \times 1$ blurring kernel.

3.2. Multi-material compiler

The representation of objects as a "tree of operations" allows for the efficient storage of geometric and volumetric (material assignment) information in OpenVCAD. Likewise, the expressions of leaf nodes as implicit geometry maintains a high level of dimensional accuracy. To maintain high degrees of both object and material complexity, OpenVCAD allows for material distributions to be defined as operators. However, defining material distributions in operators is not a universally better approach. For instance, medical scan data directly couples material assignments to geometric regions. As such, OpenVCAD allows for material distributions to be specified alongside geometry for the `voxel()` input node. Flexibility in where the material distributions defined in the tree allows for users to select the method that best suits the needs of their design.

The flexibility in material specification is made possible by the OpenVCAD multi-material compiler. Fig. 10 shows an overview of the various components of the compiler. Designs are specified using the OpenVCAD modeling language as a text script. The text is supplied to the parser along with configuration for the specific machine that will be used print the object. The parser translates the text script into a tree data structure. Stochastic sampling is then performed based on the methods specified in the nodes of the tree. A user-selected volumetric exporter is then used to save the design into a desired format.

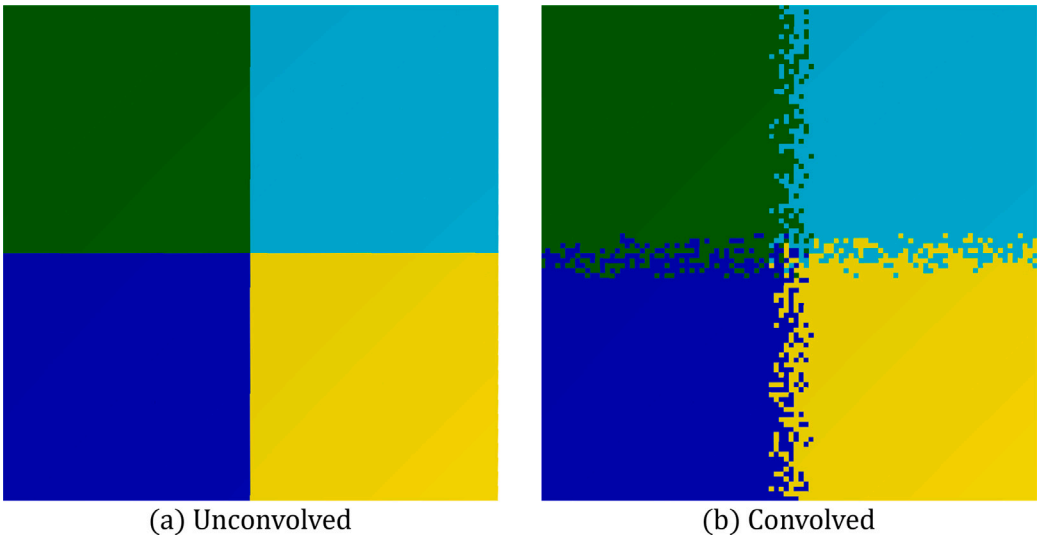


Fig. 8. (a) four adjacent squares are defined with different materials. (b) the convolution is performed on the composite object using a $10 \times 10 \times 1$ kernel. The resulting object has blended material boundaries.

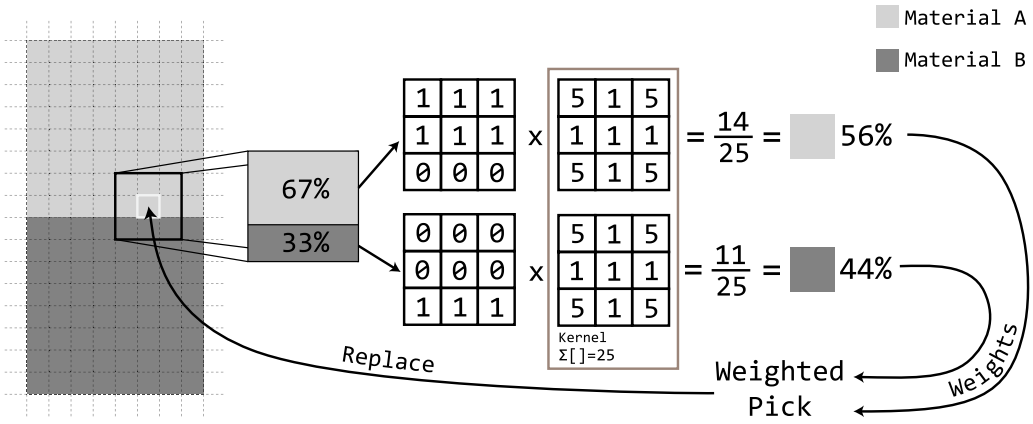


Fig. 9. Convolution is performed over two material boundaries. The method multiplies the component-wise material distribution to determine corresponding weights for stochastic selection. The material at the center of the kernel is replaced. In OpenVCAD, this method is extended into three dimensions to perform the discrete convolution over the entire object.

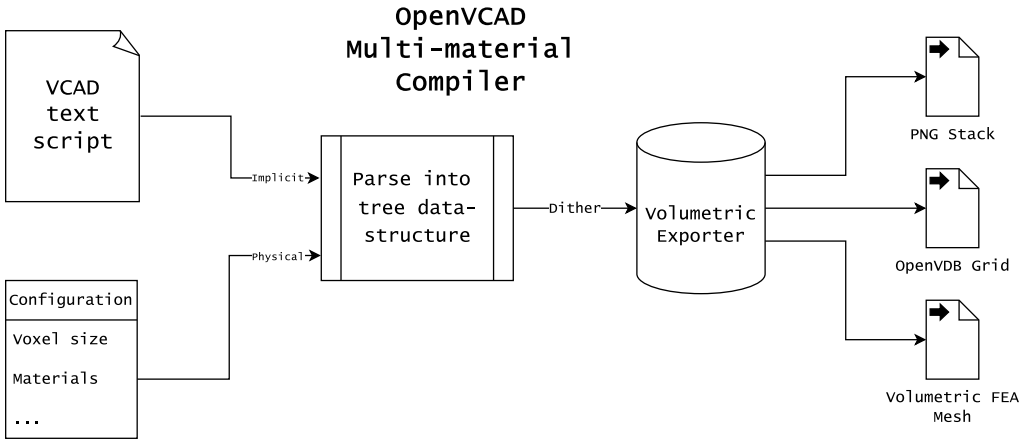


Fig. 10. An overview of the multiple components of the OpenVCAD multi-material compiler.

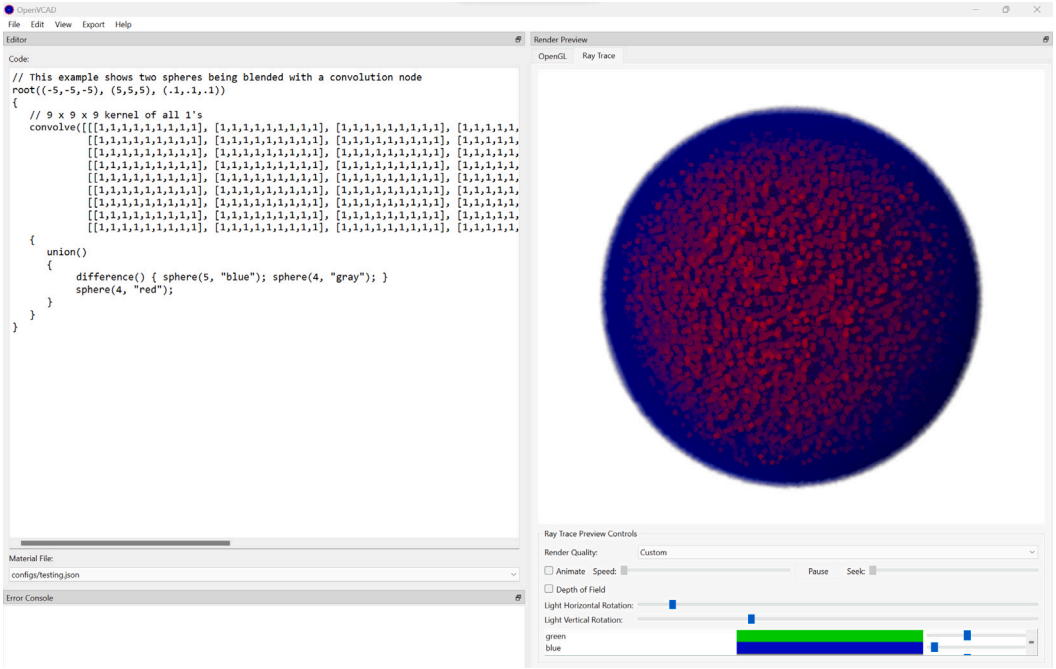


Fig. 11. A screenshot of the OpenVCAD development environment. The left panel contains an editor for the script text. The right panel is a ray-traced sparse-voxel rendering of the compiled design. The design is first compiled into a NanoVDB grid which is ray-traced. The render is showing two spheres that were convolved along their boundaries to create a gradient. The opacity control of the view allows us to visualize the internal red material through the outer blue sphere. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

AM workflows use slicing software to convert 3D objects into layer information that is commonly expressed as either g-code pathing or pixel-based bitmap stacks. Consequently, numerous slicing packages exist to convert explicit geometry files, such as Standard Triangle Language (STL) or the 3D Manufacturing Format (3MF), to a printable format [73]. OpenVCAD implements slicing functionality through volumetric exporters. In the current implementation, we provide three exporters that can be used for voxel-based 3D-printers, graphics & rendering, and simulation & optimization. The most common exchange format accepted by voxel-based printers is the PNG stack. If a design is exported from OpenVCAD as PNG stacks, each image file represents a single layer-slice. Furthermore each pixel in a layer image contains a color mapped to specific material channel in a printer. The directory of PNG images can then be transferred to the printer for construction.

Interactive rendering is another critical feature of a multi-material design method. As such, OpenVCAD can export designs to OpenVDB/NanoVDB voxel-grids [50,52]. In this format, each voxel contains an integer ID that corresponds to the user-defined IDs in the material configuration supplied to the compiler. Fig. 11 shows a screenshot of the OpenVCAD integrated development environment (IDE). The IDE provides a text editor for the VCAD modeling language script in the left panel. An option can be selected to compile the design for rendering in the right panel. The NanoVDB volumetric exporter is used to export the design as a voxel-grid to the GPU. A ray-tracer is then used to render the design with realistic shading. Along the bottom of the render windows are sliders that control the opacity of the materials present in the design. As seen in the render panel in Fig. 11, the custom ray-tracer is capable of rendering objects that contain transparent elements, allowing the designers to visualize internal volumetric structures.

In their review paper on multi-material design, Fayolle et al. stated that a robust heterogeneous design method must be interoperable with other computer-aided engineering tools. The OpenVCAD compiler can export designs as volumetric finite element meshes for the simulation of designs using the Abaqus FEA package [74]. Forward integration (simulation and analysis) and inverse integration (shape and/or material optimization) is possible by using the finite element volumetric exporter and scripting language input respectively.

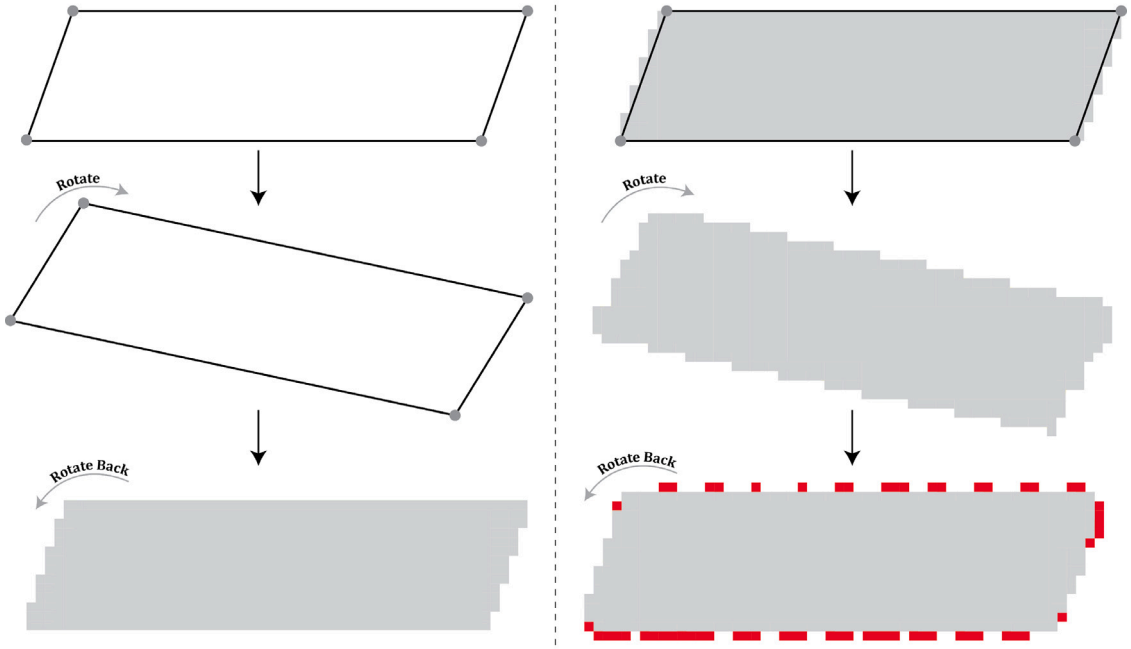
When sampling with the volumetric compiler, the designer must specify the range of $\mathbb{R}^3$ and the voxel-size they wish to use for their design. It is common for voxel-based 3D-printers to use asymmetric voxels. As a result, OpenVCAD allows designers to specify each XYZ dimension of the voxels independently used to discretize the design. This customization allows for different sampling rates and ranges to be used for each axis. To ensure that all geometric features are expressed, the designer must ensure that the Nyquist–Shannon sampling theorem holds for the spatial sampling rate they pick in all three dimensions [75]. This means picking a sampling rate whose frequency is at least twice that of the highest frequency component among all of the geometry and material nodes in the design tree. If periodic functions are used to define material distributions or geometry, the minimum sample frequency, F_{min}, can be calculated using:

$$F_{min} = min\left(\frac{2}{P_0}, \ldots, \frac{2}{P_i}\right), \tag{4}$$

where P_i is the period of the ith function in an OpenVCAD tree.

3.2.1. Affine invariance

One of the major disadvantages of sampled representations like voxels is they are not affine-transformed accurately. Unlike b-reps, voxelized objects are not affine-invariant. Fig. 12 shows the same affine transformation applied to shapes designed with points & lines and pixels. As demonstrated in the figure, transforming discrete sampled objects can lead to decreased model accuracy. Affine variance presents a major roadblock for designers wishing to create engineering-grade



(a) Invariant transforms (b) Variant transforms

Fig. 12. Example of using affine invariant (a) and non-affine-invariant (b) representations while rotations are applied to an object. In (a), rotations are performed on the points before sampling to maintain affine-invariance. However, (b) is sampled before rotations are applied, leading to error (shown as red pixels) when rotated back to the original orientation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

objects using discrete volumetric information. OpenVCAD avoids these issues by leveraging the design tree. Because objects are represented implicitly as nodes in a tree, we can perform all affine transformations before the object is sampled. This is accomplished by propagating parent transformations down to children as the tree is evaluated.

4. Performance analysis

Through benchmarking and analysis, we aim to demonstrate that the methods proposed here yield a feasible tool for real-world volumetric design. Note that these results demonstrate the performance of the initial version of OpenVCAD, which omits potential anticipated optimizations. First, we analyze the algorithmic complexity of evaluating each of OpenVCAD's node types. Then, we present a run-time analysis for slicing single nodes. Finally, we assess the effect of tree complexity on slice time.

The time complexity of evaluating a single location in $\mathbb{R}^3$ for each node in the OpenVCAD language is given in Table 2. CSG primitives, such as the `sphere()` and `cube()` nodes, are evaluated against pre-compiled code that runs in $O(1)$ time. Similarly the `mesh()` node is pre-processed into a sparse grid that is evaluated in $O(1)$ time. The time complexity of the `f_rep()` and `f_grade()` nodes are driven by their reliance on Dijkstra's Shunting Yard Algorithm to parse mathematical string expressions [76]. The Shunting Yard algorithm runs in $O(n)$ where n is the number of tokens in the equation. Even though the complexity for functional nodes is not constant, the complexity of evaluating each location is independent of the model and grid size, which is critical for keeping run times tractable. In most use cases, the number of tokens in the `f_rep()` and `f_grade()` equations is small and near constant. Similarly, the performance of `convolve()` nodes is dependent on the kernel size, which in practice is typically small enough to produce near constant-time behavior. This relationship is illustrated by the timing results of Table 3.

OpenVCAD supports multithreading, therefore the timing benchmarking tests were conducted using an AMD Ryzen 7 7700X 8-core processor with 16 threads. In these tests, individual nodes are sampled across a large grid and the evaluation and slicing times were recorded. Evaluation time refers to the time required to evaluate the tree and determine the material for each voxel. Slicing time refers to the time needed to output the voxel information as a PNG file. The workload was distributed in a way that allowed each thread to individually sample and export a distinct layer of the object. The data reported in Table 3 are the wall-clock times taken by the multithreaded evaluation. The results of benchmarking tests show that the slicing time remains constant across node types. As indicated by the algorithmic complexities, the evaluation time was higher for nodes that express materials compared to nodes that only express geometry. The throughput of each test is expressed as the voxels sampled per second. This metric indicates the total geometric bandwidth OpenVCAD can compile per unit time. The evaluation time is of particular interest because this operation is fundamental to all volumetric exporters. The slicing time is dependent on the volumetric exporter used and can vary depending on disk quality and format used. These results show that evaluation of all nodes in OpenVCAD perform within an order of magnitude of slicing time.

Additionally, we investigated how the complexity of the tree impacts the compile time of an object. This assessment uses the same

Table 3
Benchmarking results for individual nodes in OpenVCAD. Each test was completed with a grid size of $[2^{11}, 2^{11}, 2^{12}]$ resulting in $2^{11} \times 2^{11} \times 2^{12} \approxeq 1.7 \times 10^{10}$ voxels. Evaluation time is the number of seconds taken to compile the multi-material design into a volumetric format. Slicing time is the number of seconds required to save this volumetric data to stacked PNG files. For a single layer, the slicing time refers only to the time required to encode that layer's matrix of color values into the PNG format and save to disk. The slicing time reported in the table refers to the sum of this time for all layers saved to disk.

Node	Evaluation time (s)	Slicing time (s)	Voxels per second
Sphere	35	248	3.7×10^7
Cube	44	241	3.6×10^7
Mesh	163	251	2.5×10^7
F-Rep	584	248	1.2×10^7
Convolve	62	250	3.3×10^7
F-Grade	1147	223	7.5×10^6

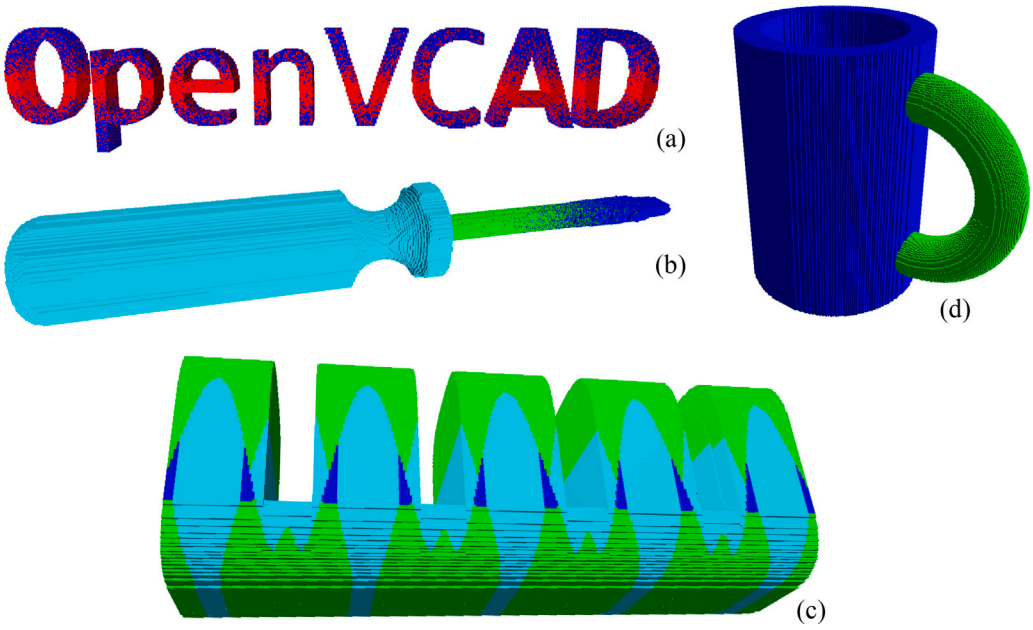


Fig. 13. Example objects used for benchmarking data in Table 4. All examples make use of the `f-grade()` node. (a) and (b) are defined using the `mesh()` node. (c) and (d) are created using the `f-rep()` node.

Fig. 14. The screwdriver from 13 (b) constructed using a Stratasys J750 Polyjet printer. Multiple materials are used to add a soft-touch handle and a multi-color shaft. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

CPU configuration that was used for Table 3. Fig. 13 shows renders of the four composite objects that were chosen to demonstrate multiple OpenVCAD tree configurations and real world use cases. These objects are varied in physical size and type of nodes used. Fig. 13(a) shows a mesh node that is graded with two materials. Figs. 13(b) and 14 shows a screwdriver that is comprised of a mesh node and graded with three materials. Fig. 13(c) is a soft actuator that is comprised of a functional geometry node and graded with three materials. Fig. 13(d) shows a mug that is designed using multiple functional geometry nodes and boolean operations. The results of evaluating and slicing these four objects are recorded in Table 4. As listed in Table 4, the largest and most complex object took approximately 16 min to be sampled into printable files. In contrast, smaller and simpler objects were converted in under five minutes. As in the previous benchmarking test, the voxel outputs per second were within an order of magnitude for all objects tested. This finding indicates that although the time taken to evaluate an object is dependent on the complexity of the tree, the processing time is dominated by the size of the objects being expressed.

5. Case studies

The following case studies were conducted using OpenVCAD to demonstrate its efficacy as a multi-material additive manufacturing workflow. Both test cases were chosen to highlight OpenVCAD design trees that use different forms of geometry, material usage, and compositing. Likewise, both case studies highlight OpenVCAD's capability to design meta-materials and compliant mechanisms.

Table 4
Benchmarking results various real-world objects shown in Fig. 13.

Example	Object size (mm)	Voxels	Evaluation time (s)	Slicing time (s)	Voxels per second
OpenVCAD Text[a]	140 × 7 × 28	2.8×10^8	21	7	1.0×10^7
Screwdriver[b]	22 × 144 × 22	6.7×10^8	97	15	5.9×10^6
Soft Actuator[c]	48 × 48 × 75	1.8×10^9	251	41	6.3×10^6
Mug[d]	96 × 63 × 90	5.6×10^9	836	120	5.9×10^6

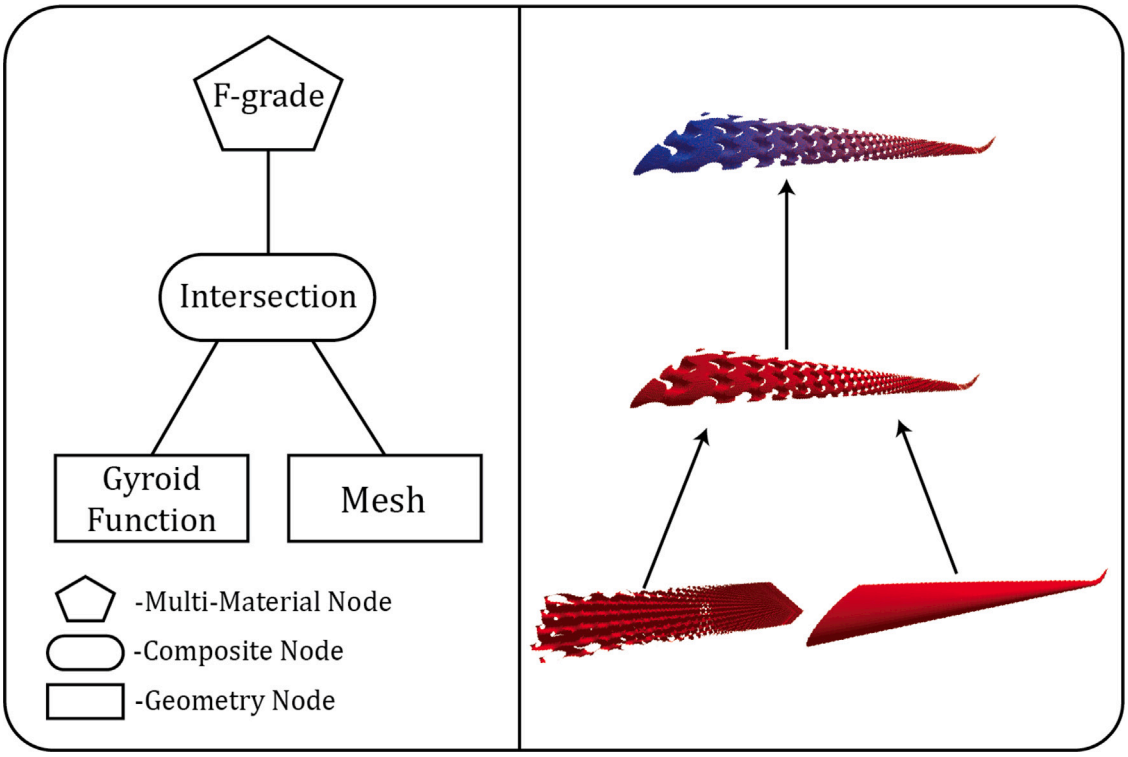


Fig. 15. Shows a tree that generates a triply periodic gyroid infill pattern for a wing. The network makes use of the `f_rep()` node to generate the gyroidal surface and an `intersection()` node to combine it with the wing geometry defined in a `mesh()` node. Finally, a `f_grade()` node is used to cross-grade two materials together along the length of the wing. The full 8-line VCAD script is given in Fig. A.20.

5.1. Functionally graded wing

OpenVCAD's functional geometry node is well suited to express triply periodic geometry like the gyroid pattern in Fig. 15. Eq. (5) expresses a gyroidal surface as a function of `(X,Y,Z)`. Using the `f_rep()` node in OpenVCAD, we can use this equation to generate a uniform gyroidal pattern and bound it with a mesh surface to form the infill as shown in Fig. 15. Adding a functional grading node to the network creates a cross-grade of two materials to form a meta-material. The material distribution could be modified using functional grading to optimize for multiple factors such as stiffness, durability, or thermal conductivity depending on the location on the wing.

Similarly, Fig. 16 shows the gyroidal infill being further parameterized by varying the unit cell length, a, across the length of the wing. The model was oriented such that the longest dimension of the wing was aligned with the x-axis. A parameter of $a = -0.063x + 1.55$ was used to linearly decrease the unit cell size. This method could be used in conjunction with simulated stress distribution to achieve a similar effect to that proposed by Kim et al. [77,78]. In all, the object shown in Figs. 15 and 16 takes only 8 lines of OpenVAD script to generate and is provided in Fig. A.20.

$$\sin\left(\frac{2\pi}{a}x\right)\cos\left(\frac{2\pi}{a}y\right) + \sin\left(\frac{2\pi}{a}y\right)\cos\left(\frac{2\pi}{a}z\right) + \sin\left(\frac{2\pi}{a}z\right)\cos\left(\frac{2\pi}{a}x\right) = h \tag{5}$$

5.2. Multi-material organic lattices

A key application of functional grading is with lattice structures. Lattices leverage the geometric freedom provided by additive manufacturing to construct objects that yield better mechanical performance compared to objects with a homogeneous composition. Multi-material design adds another dimension by which lattice structures might improve designs. Fig. 17 shows how a basic tetrahedron lattice unit cell can be created using OpenVCAD. The `strut()` geometric primitive node allows the designer to define a cylindrical strut object of a given length. Using this, a functional grading is applied to each individual unit strut to yield a member that might resist compression in the center, and bond well with adjacent struts at the ends. The unit struts are rotated, translated, and scaled to fit the designed dimension of the tetrahedron. To combine the six struts into a single object, the `sum()` node is used. This node is similar to the `union()` node in that it combines multiple children into a single object; however, it differs in how it combines the child geometry. The `sum()` node treats a child geometry node as a scalar field of distances from the surface of the strut. When sampled in $\mathbb{R}^3$ the scalar values are summed and a level set is taken through it. This, combined with a blending constant, cause the joints of the tetrahedron to organically morph together similar to how metaballs behave [79].

Adding members to the lattice is as simple as computing another point in the 3D triangulation and repeating the steps to define, grade, transform and sum them with existing struts. Fig. 18 shows this idea

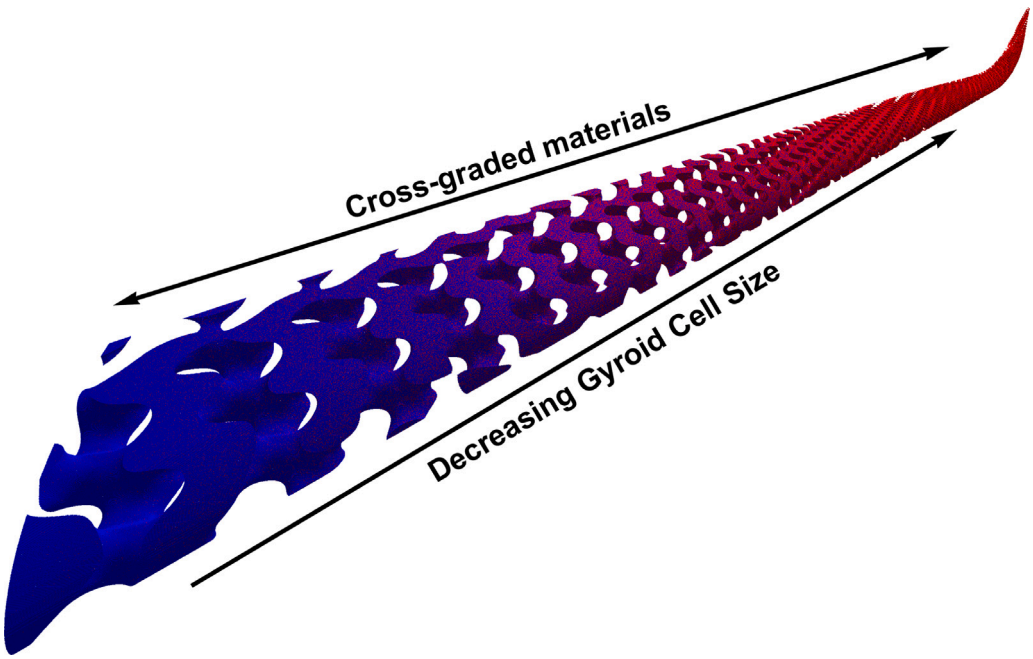


Fig. 16. Shows the final object from Fig. 15 in detail. The object's quality was enhanced by decreasing the cell size of the gyroid pattern along the length of the wing. This preserves geometric detail in thinner sections of the wing. Similarly, a heavier, stiffer material (blue) gives way to a lighter, more flexible material (red) along the direction of the wing. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in practice on a larger lattice structure. The strut geometry for this object was generated by first performing a 3D tetrahedral meshing of a rectangular prism using the method prescribed by Tournois et al. [80]. For each edge in the tetrahedral mesh a strut is generated and two overlapping normal distributions are applied to functionally grade materials together along the length of the strut. A `sum()` node is used to combine all of the struts into a single composite object. This example expands on the mechanical motivation of Fig. 17 by highlighting a composite lattice structure with mechanical properties that could be optimized using functional grading. For example, a designer could tweak the concentration of an energy absorptive material graded into the center of the strut. Similarly, architecting the concentration of stiff materials in certain struts could yield a compliant mechanism that deforms more under load in certain areas of the design.

6. Bottlenecks and areas for improvements

Although OpenVCAD performs well with the example objects in Fig. 13, processing time is highly dependent on the object size and printer resolution. As such, we identify bottlenecks in the current implementation and propose potential improvements.

One limitation in the current implementation of OpenVCAD is the inability to prune or simplify design trees containing redundant expressions. An example of redundant design at the cost of performance would be expressing periodic geometry, such as the soft actuator in Fig. 13(c), with multiple discrete nodes for each period defined in Cartesian space. Instead, this same geometry should be expressed over multiple periods with a parameterized equation using cylindrical coordinates. Similar to how traditional programming language compilers, like GCC, optimize the code before generating machine instructions, the OpenVCAD multi-material compiler would benefit from analogous methods to optimize design trees.

Furthermore, performance could be improved by reducing the number of locations sampled to determine the value of each voxel. This is possible if there are large regions in an object that do not vary in geometry or material, which is common in real-world applications. Duff proposed a method that uses interval arithmetic to partition space to simplify an object's geometry [81]. Similarly, Keeter proposed an interval arithmetic method that can be used to rapidly evaluate closed-form implicit surfaces by skipping regions that do not have varying geometry [82]. Investigation into applying these interval arithmetic methods to both the geometric and material compositions of an object could reduce the sampling space and improve performance.

GPU acceleration could improve OpenVCAD's object compilation speed. Keeter proposed a method that compiles closed-form implicit surfaces into an intermediate language that is interpreted using massively parallel rendering on GPUs [82]. A similar approach applied to the OpenVCAD modeling language could dramatically reduce object rendering and sampling times.

Likewise, OpenVCAD allows for full control of material distribution throughout an object. This puts designers in control of where and what materials compose a region. However, it is currently up to designers to ensure that material combinations are printable. This can lead to a situation where an invalid material combination results in a failed print, leaving the designer to find a 'needle in a haystack' to fix the error. More investigation into a design validation mechanism is needed to ensure that unprintable regions are identified and corrected automatically during the design process.

7. Applications of OpenVCAD to non-inkjet systems

OpenVCAD is currently targeted at AM systems such as inkjet that support design import via stacked image files, though it also supports export of boundary surface representations compatible with virtually all AM systems. Further investigation is needed to apply this method to systems such as fused filament fabrication, directed energy deposition, and powder bed fusion. One potential avenue for the application of OpenVCAD to fused filament fabrication is using multiple hot ends with appropriate halftoning [4], or mixing extruders. Recent work has shown that it is possible to mix multiple filaments in melt chamber of hot ends to create gradients [83–85]. Similarly, researchers have employed an analogous strategy by using multiple wire feeders to create material mixtures in the melt pool of a directed energy deposition systems [86]. The mixing ratios of the independent extruders or wire feeders could be defined using OpenVCAD and sampled across toolpaths to define gradients. For laser powder bed fusion systems it has

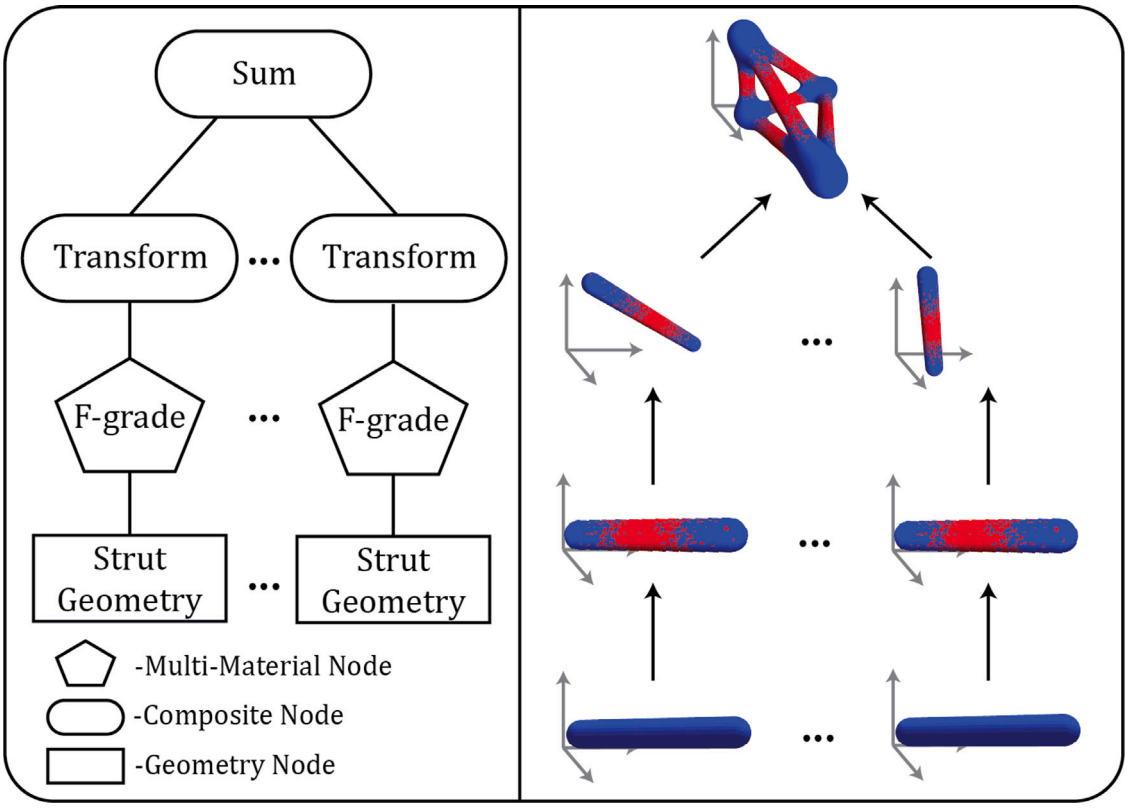


Fig. 17. Shows a network that generates an organic tetrahedron with graded members. Component geometry is defined as individual unit struts along the x-axis. The struts are then functionally graded to form a meta-material where the center (shown as a red material) exhibits better compression resistance when compared with the blue material at the ends. The blue material exhibits better adhesion between connected strut endpoints. The struts are transformed and combined via a summing operation to yield a composite tetrahedron. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

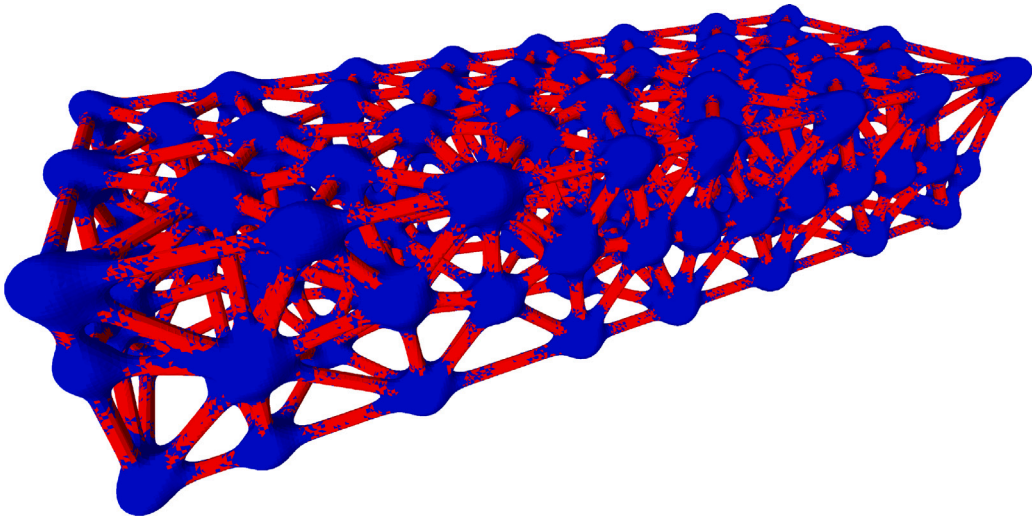


Fig. 18. A 3D lattice structure composed of the organic tetrahedra outlined in Fig. 17.

been shown that varying laser power can yield objects with different mechanical properties [87]. Future extensions of OpenVCAD could involve grading laser power to yield objects with similar mechanical properties to those printed with multiple materials, even with just one powder.

8. Conclusion

In this paper, we have presented a novel volumetric design method for multi-material additive manufacturing. The OpenVCAD language and compiler creates a compact and portable representation of dense volumetric designs. The proposed set of functional nodes establishes a robust framework for applied research into volumetric 3D printing.

```
root((-2, -2, -3.1415), (2, 2, 3.1415), (.07, .07, .07))
{
    fgrade(["0.2", "sin(4.5 * z + 1.57) - 0.1", "100 * (sin(
   atan(50 * sin(4.5 * z - 1.57) - 60)) + 1) - rho * sin(phic)
    + 0.1"],["blue","cyan","yellow"], "thresh")
    {
        function("(0.5 * (rho * sin(phic) - 0.5)^4 + (rho * sin
   (phic + 0.5 * pi))^4 + 0.5 * (sin(atan(100 * sin(4.5 * z -
   1.57) - 60)) + 1) * (sin(atan(20 * sin(phic))) + 1) - 1)",
   "red");
    }
}
```

Fig. A.19. VCAD code used to generate the actuator in Fig. 7.

```
root((-23, -5, -1), (23, 5, 3), (0.05, 0.05, 0.05)){
  fgrade(["0.021739 * x + 0.5", "-0.021739 * x + 0.5"], ["red",
    "blue"], "prob"){
    intersection(){
      mesh("OpenVCAD-Examples/multi-material/wing/wing.stl", "
   red");
      function("sin(((2 * pi) / (-0.06304347 * x + 1.55)) * x)
   * cos(((2 * pi) / (-0.06304347 * x + 1.55)) * y) +
              sin(((2 * pi) / (-0.06304347 * x + 1.55)) * y) *
   cos(((2 * pi) / (-0.06304347 * x + 1.55)) * z) +
             sin(((2 * pi) / (-0.06304347 * x + 1.55)) * z) *
   cos(((2 * pi) / (-0.06304347 * x + 1.55)) * x)", "red");
}}}
```

Fig. A.20. VCAD code used to generate the wing in Fig. 15.

Functional geometry nodes allow designers to express complex and parametric shapes that retain their accuracy when transformed and combined to form composite designs. Functional grading and convolution nodes allow research into lattice structures and meta-materials that can vary material composition throughout geometric regions. The flexibility of the OpenVCAD tree and compiler enables designs that have high geometric and material complexity, a dichotomy that has limited previous methods. Further exploration of "digital materials" and pseudo-alloys is made possible with OpenVCAD's extensible modeling language and compiler. Towards these goals, we have released OpenVCAD as an open-source project to advance multi-material additive manufacturing research. While OpenVCAD introduces a new method for multi-material design, the limitations with evaluation speed and design validation should be addressed to offer speed that is comparable with commercially available single-material design software.

CRediT authorship contribution statement

Charles Wade: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Graham Williams:** Writing – review & editing, Writing – original draft, Validation, Resources. **Sean Connelly:** Conceptualization. **Braden Kopec:** Conceptualization. **Robert MacCurdy:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Robert MacCurdy has patent Multi-Material Volumetric Three-Dimensional Modeling pending to University of Colorado Boulder.

Data and material availability

All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials as well as provided on the public OpenVCAD website (https://matterassembly.org/openvcad). Additional data related to this paper may be requested from the corresponding author.

Funding

This work is supported by startup funds to R. MacCurdy provided by the University of Colorado Boulder.

Appendix. VCAD scripts for examples

See Figs. A.19 and A.20.

References

[1] C. Wade, M. Borish, Hybrid curve fitting for reducing motion commands in object construction, in: 2022 International Solid Freeform Fabrication Symposium, University of Texas at Austin, 2022.

[2] P. Fayolle, L. McLoughlin, M. Sanchez, G. Pasko, A. Pasko, Modeling and visualization of multi-material volumes, Sci. Vis. 13 (2) (2021) 117–148.

[3] J. Gonzalez-Gutierrez, S. Cano, S. Schuschnigg, C. Kukla, J. Sapkota, C. Holzer, Additive manufacturing of metallic and ceramic components by the material extrusion of highly-filled polymers: A review and future perspectives, Materials 11 (5) (2018) 840.

[4] T. Kuipers, E. Doubrovski, J. Verlinden, 3D hatching: linear halftoning for dual extrusion fused deposition modeling, in: Proceedings of the 1st Annual ACM Symposium on Computational Fabrication, 2017, pp. 1–7.

[5] F. Fenollosa, J.R. Gomà, I. Buj-Corral, A.T. Otero, J. Minguella-Canela, R. Uceda, A. Valls, M. Ayats, Foreseeing new multi-material FFF-additive manufacturing concepts meeting mimicking requirements with living tissues, Procedia Manuf. 41 (2019) 1063–1070.

[6] S. Bijadi, E. de Bruijn, E.Y. Tempelman, J. Oberdorf, Application of multi-material 3D printing for improved functionality and modularity of open source low-cost prosthetics: A case study, in: Frontiers in Biomedical Devices, vol. 40672, American Society of Mechanical Engineers, 2017, V001T10A003.

[7] A. Castellví, L. Poudelet, A. Tejo, L. Calvo, R. Uceda, P. Lustig, J. Minguella, I. Buj, F. Fenollosa, L. Krauel, et al., The commissioning of a hybrid multi-material 3D printer, in: IOP Conference Series: Materials Science and Engineering, vol. 1193, IOP Publishing, 2021, 012044.

[8] J. McPherson, W. Zhou, A chunk-based slicer for cooperative 3D printing, Rapid Prototyp. J. 24 (9) (2018) 1436–1446.

[9] J. Prusa, Original Prusa MMU3 upgrade kit (for MK3S+) | Original Prusa 3D printers directly from Josef Prusa - prusa3d.com.

[10] J.W. Choi, H.C. Kim, R. Wicker, Multi-material stereolithography, J. Mater Process. Technol. 211 (2011) 318–328.

[11] R. Wicker, F. Medina, C. Elkins, Multiple material micro-fabrication: extending stereolithography to tissue engineering and other novel applications, in: 2004 International Solid Freeform Fabrication Symposium, 2004.

[12] H. Cui, D. Yao, R. Hensleigh, H. Lu, A. Calderon, Z. Xu, S. Davaria, Z. Wang, P. Mercier, P. Tarazaga, et al., Design and printing of proprioceptive three-dimensional architected robotic metamaterials, Science 376 (6599) (2022) 1287–1293.

[13] Q. Ge, Z. Chen, J. Cheng, B. Zhang, Y.-F. Zhang, H. Li, X. He, C. Yuan, J. Liu, S. Magdassi, et al., 3D printing of highly stretchable hydrogel with diverse UV curable polymers, Sci. Adv. 7 (2) (2021) eaba4261.

[14] R. Bahr, X. He, B. Tehrani, M.M. Tentzeris, A fully 3D printed multi-chip module with an on-package enhanced dielectric lens for mm-wave applications using multimaterial stereo-lithography, IEEE MTT-S Int. Microw. Symp. Digest 2018-June (2018) 1561–1564.

[15] D. Dikovsky, S. Shtilerman, System and method for fabricating a body part model using multi-material additive manufacturing (US patent 9,999,509), 2018.

[16] R.N. Leyden, J.S. Thayer, B.J.L. Bedal, T.A. Almquist, C.W. Hull, J.M. Earl, T.A. Kerekes, D.R. Smalley, C.M. Merot, R.P. Fedchenko, M.S. Lockard, T.H. Pang, D.T. That, Selective deposition modeling method and apparatus for forming three-dimensional objects and supports (US patent 6,193,923), 2001.

[17] A.D. Castiaux, E.A. Hayter, M.E. Bunn, S.R. Martin, D.M. Spence, PolyJet 3D-printed enclosed microfluidic channels without photocurable supports, Anal. Chem. 91 (10) (2019) 6910–6917.

[18] A. Hosny, S.J. Keating, J.D. Dilley, B. Ripley, T. Kelil, S. Pieper, D. Kolb, C. Bader, A.-M. Pobloth, M. Griffin, R. Nezafat, G. Duda, E.A. Chiocca, J.R. Stone, J.S. Michaelson, M.N. Dean, N. Oxman, J.C. Weaver, From improved diagnostics to presurgical planning: High-resolution functionally graded multimaterial 3D printing of biomedical tomographic data sets, 3D Print. Addit. Manuf. 5 (2) (2018) 103–113.

[19] R. Worsley, L. Pimpolari, D. McManus, N. Ge, R. Ionescu, J.A. Wittkopf, A. Alieva, G. Basso, M. Macucci, G. Iannaccone, et al., All-2D material inkjet-printed capacitors: toward fully printed integrated circuits, Acs Nano 13 (1) (2018) 54–60.

[20] T. Carey, S. Cacovich, G. Divitini, J. Ren, A. Mansouri, J.M. Kim, C. Wang, C. Ducati, R. Sordan, F. Torrisi, Fully inkjet-printed two-dimensional material field-effect heterojunctions for wearable and textile electronics, Nature Commun. 8 (1) (2017) 1202.

[21] T. Pinto, C. Chen, C. Pinger, X. Tan, 3D-printed liquid metal-based stretchable conductors and pressure sensors, Smart Mater. Struct. 30 (9) (2021) 095005.

[22] R. MacCurdy, R. Katzschmann, Y. Kim, D. Rus, Printable hydraulics: A method for fabricating robots by 3D co-printing solids and liquids, IEEE Int. Conf. Robot. Autom. (ICRA) 2 (2016).

[23] Y. Wei, Y. Chen, Y. Yang, Y. Li, Novel design and 3-D printing of nonassembly controllable pneumatic robots, IEEE/ASME Trans. Mechatronics 21 (2) (2016) 649–659.

[24] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R.W. Sumner, W. Matusik, B. Bickel, Computational design of mechanical characters, ACM Trans. Graph. 32 (4) (2013) 83.

[25] D. Chen, X. Zheng, Multi-material additive manufacturing of metamaterials with giant, tailorable negative Poisson's ratios, Sci. Rep. 8 (1) (2018) 1–8.

[26] M.J. Mirzaali, A. Caracciolo, H. Pahlavani, S. Janbaz, L. Vergani, A. Zadpoor, Multi-material 3D printed mechanical metamaterials: Rational design of elastic properties through spatial distribution of hard and soft phases, Appl. Phys. Lett. 113 (24) (2018).

[27] A. Ion, J. Frohnhofen, L. Wall, R. Kovacs, M. Alistar, J. Lindsay, P. Lopes, H.-T. Chen, P. Baudisch, Metamaterial mechanisms, in: Proceedings of the 29th Annual Symposium on User Interface Software and Technology, 2016, pp. 529–539.

[28] W. Gao, Y. Zhang, D. Ramanujan, K. Ramani, Y. Chen, C.B. Williams, C.C. Wang, Y.C. Shin, S. Zhang, P.D. Zavattieri, The status, challenges, and future of additive manufacturing in engineering, Comput. Aided Des. 69 (2015) 65–89.

[29] C. Zhang, F. Chen, Z. Huang, M. Jia, G. Chen, Y. Ye, Y. Lin, W. Liu, B. Chen, Q. Shen, et al., Additive manufacturing of functionally graded materials: A review, Mater. Sci. Eng. A 764 (2019) 138209.

[30] M. Gao, L. Li, Y. Song, Inkjet printing wearable electronic devices, J. Mater. Chem. C 5 (12) (2017) 2971–2993.

[31] N. Munasinghe, M. Woods, L. Miles, G. Paul, 3-D printed strain sensor for structural health monitoring, in: 2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2019, pp. 275–280.

[32] J. Persad, S. Rocke, A survey of 3D printing technologies as applied to printed electronics, IEEE Access 10 (2022) 27289–27319.

[33] B. Hayes, T. Hainsworth, R. MacCurdy, Liquid-solid co-printing of multi-material 3D fluidic devices via material jetting, Addit. Manuf. (2022) 102785.

[34] R. Maccurdy, R. Katzschmann, Y. Kim, D. Rus, Printable hydraulics: A method for fabricating robots by 3D co-printing solids and liquids, IEEE Int. Conf. Robot. Autom. (ICRA) (2016).

[35] R. MacCurdy, J. Lipton, S. Li, D. Rus, Printable programmable viscoelastic materials for robots, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2016, pp. 2628–2635.

[36] T.K. Brown-Moore, S. Balaji, T. Williams, J. Lipton, Fabrication of liquid-filled voronoi foams for impact absorption using material jetting technology, in: 2022 International Solid Freeform Fabrication Symposium, 2022.

[37] N.M. Jacobson, L. Smith, J. Brusilovsky, E. Carrera, H. McClain, R. MacCurdy, Voxel printing anatomy: Design and fabrication of realistic, presurgical planning models through bitmap printing, JoVE (J. Vis. Exp.) (2022).

[38] N. Jacobson, E. Carerra, L. Smith, L. Browne, N. Stence, A. Sheridan, R. MacCurdy, Defining soft tissue: Bitmap printing of soft tissue for surgical planning, 3D Print. Addit. Manuf. 9 (6) (2022) 461–472.

[39] N. Jacobson, H. McClain, M. New, Digital workflow for high-risk, low-volume procedure simulation, J. Biomed. Res. 4 (1) (2023) 1–7.

[40] E.L. Doubrovski, E.Y. Tsai, D. Dikovsky, J.M. Geraedts, H. Herr, N. Oxman, Voxel-based fabrication through material property mapping: A design method for bitmap printing, Comput. Aided Des. 60 (2015) 3–13.

[41] P. Lienhardt, Topological models for boundary representation: a comparison with n-dimensional generalized maps, Comput. Aided Des. 23 (1991) 59–82.

[42] S. Lefebvre, Icesl: A GPU accelerated CSG modeler and slicer, in: AEFA'13, 18th European Forum on Additive Manufacturing, 2013.

[43] S.D. Roth, Ray casting for modeling solids, Comput. Graph. Image Process. 18 (2) (1982) 109–144.

[44] Q. Li, Q. Hong, Q. Qi, X. Ma, X. Han, J. Tian, Towards additive manufacturing oriented geometric modeling using implicit functions, Vis. Comput. Ind. Biomed. Art 1 (1) (2018) 1–16.

[45] S. Hasanov, A. Gupta, A. Nasirov, I. Fidan, Mechanical characterization of functionally graded materials produced by the fused filament fabrication process, J. Manuf. Process. 58 (2020) 923–935.

[46] A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, Function representation in geometric modeling: concepts, implementation and applications, Vis. Comput. 11 (1995) 429–446.

[47] A. Penev, F-rep designer 2.0–everything is a code, Int. J. Comput. Sci. Issues (IJCSI) 15 (4) (2018) 7–13.

[48] A. Pasko, O. Fryazinov, T. Vilbrandt, P.-A. Fayolle, V. Adzhiev, Procedural function-based modelling of volumetric microstructures, Graph. Models 73 (5) (2011) 165–181.

[49] A. Pasko, V. Adzhiev, B. Schmitt, C. Schlick, Constructive hypervolume modeling, Graph. Models 63 (6) (2001) 413–442.

[50] K. Museth, VDB: High-resolution sparse volumes with dynamic topology, ACM Trans. Graph. 32 (3) (2013).

[51] T. McReynolds, D. Blythe, Advanced Graphics Programming using OpenGL, Elsevier, 2005.

[52] K. Museth, Nanovdb: A GPU-friendly and portable VDB data structure for real-time rendering and simulation, in: ACM SIGGRAPH 2021 Talks, 2021, pp. 1–2.

[53] P.-A. Fayolle, A. Pasko, B. Schmitt, N. Mirenkov, Constructive heterogeneous object modeling using signed approximate real distance functions, J. Comput. Inf. Sci. Eng. 6 (3) (2005) 221–229.

[54] G. Wyvill, C. McPheeters, B. Wyvill, Soft objects, in: Advanced Computer Graphics: Proceedings of Computer Graphics Tokyo'86, Springer, 1986, pp. 113–128.

[55] Tigges, Wyvill, A field interpolated texture mapping algorithm for skeletal implicit surfaces, 1999 Proc. Comput. Graph. Int. (1999) 25–32.
[56] B. Wyvill, A. Guy, E. Galin, Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system, in: Computer Graphics Forum, vol. 18, Wiley Online Library, 1999, pp. 149–158.
[57] C. Brauer, D.M. Aukes, Automated generation of multi-material structures using the VoxelFuse framework, in: Proceedings - SCF 2020: ACM Symposium on Computational Fabrication, Association for Computing Machinery, Inc, 2020.
[58] C. Brauer, D.M. Aukes, Voxel-based CAD framework for planning functionally graded and multi-step rapid fabrication processes, in: Proceedings of the ASME Design Engineering Technical Conference, vol. 2A-2019, American Society of Mechanical Engineers Digital Collection, 2019.
[59] J. Hiller, H. Lipson, Dynamic simulation of soft multimaterial 3d-printed objects, Soft Robot. 1 (1) (2014) 88–101.
[60] K. Vidimče, S.P. Wang, J. Ragan-Kelley, W. Matusik, OpenFab: A programmable pipeline for multi-material fabrication, ACM Trans. Graph. 32 (2013).
[61] K. Vidimče, A. Kaspar, Y. Wang, W. Matusik, Foundry: Hierarchical material design for multi-material fabricationdoubrovski2015voxel, in: UIST 2016 - Proceedings of the 29th Annual Symposium on User Interface Software and Technology, Association for Computing Machinery, Inc, 2016.
[62] T.H. Luu, C. Altenhofen, A. Stork, D. Fellner, GraMMaCAD: Interactively defining spatially varying FGMs on brep CAD models, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 86212, American Society of Mechanical Engineers, 2022, V002T02A014.
[63] S. Hasanov, S. Alkunte, M. Rajeshirke, A. Gupta, O. Huseynov, I. Fidan, F. Alifui-Segbaya, A. Rennie, Review on additive manufacturing of multi-material parts: Progress and challenges, J. Manuf. Mater. Process. 6 (1) (2022).
[64] G. Elber, A review of a B-spline based volumetric representation: Design, analysis and fabrication of porous and/or heterogeneous geometries, Comput. Aided Des. (2023) 103587.
[65] Open Cascade SAS, Open Cascade Technology 7.7.0, 2023.
[66] A. Pasko, V. Adzhiev, R. Cartwright, E. Fausett, A. Ossipov, V. Savchenko, HyperFun project: a framework for collaborative multidimensional F-rep modeling, in: Eurographics/ACM SIGGRAPH Workshop Implicit Surfaces' 99, Workshop Implicit Surfaces 1999, 1999, pp. 59–69.
[67] M. Kintel, Openscad - the programmers solid 3D CAD modeller, 2010.
[68] C. Brauer, D. Aukes, Applying graded material transitions with low-cost additive manufacturing, Rapid Prototyp. J. 29 (2) (2023) 378–392.
[69] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[70] A. Partow, C++ Mathematical Expression Library (exprtk).
[71] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[72] V. Jampani, M. Kiefel, P.V. Gehler, Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[73] A. Roschli, A. Messing, M. Borish, B.K. Post, L.J. Love, ORNL slicer 2: a novel approach for additive manufacturing tool path planning, in: 2017 International Solid Freeform Fabrication Symposium, University of Texas at Austin, 2017.
[74] M. Smith, ABAQUS/Standard User's Manual, Version 6.9, Dassault Systèmes Simulia Corp, United States, 2009.
[75] C. Shannon, Communication in the presence of noise, Proc. IRE 37 (1) (1949) 10–21.
[76] E.W. Dijkstra, ALGOL-60 Translation, Mathematisch Centrum, 1961.
[77] S. Kim, An integrated design approach for infill patterning of fused deposition modeling and its application to an airfoil, Tech. rep., Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2017.
[78] S. Kim, G. Dreifus, B. Beard, A. Glick, A. Messing, A.A. Hassen, J. Lindahl, P. Liu, T. Smith, J. Failla, et al., Graded infill structure of wind turbine blade core accounting for internal stress in big area additive manufacturing, in: Proceedings of the CAMX Composite and Advanced Materials Expo, Dallas, TX, USA, 2018, pp. 15–18.
[79] A. Wakita, A. Nakano, N. Kobayashi, Programmable blobs: a rheologic interface for organic shape design, in: Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction, 2010, pp. 273–276.
[80] J. Tournois, N. Faraj, J.-M. Thiery, T. Boubekeur, Tetrahedral remeshing, in: CGAL User and Reference Manual, 5.5.2, CGAL Editorial Board, 2023.
[81] T. Duff, Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry, ACM SIGGRAPH Comput. Graph. 26 (2) (1992) 131–138.
[82] M.J. Keeter, Massively parallel rendering of complex closed-form implicit surfaces, ACM Trans. Graph. 39 (4) (2020) 141.
[83] S.K. Adapa, et al., Design and fabrication of internal mixer and filament extruder for extraction of hybrid filament composite for FDM applications, Int. J. Interact. Des. Manuf. (IJIDeM) (2023) 1–14.
[84] J.T. Green, I.A. Rybak, J.J. Slager, M. Lopez, Z. Chanoi, C.M. Stewart, R.V. Gonzalez, Local composition control using an active-mixing hotend in fused filament fabrication, Addit. Manuf. Lett. (2023) 100177.
[85] Z.C. Kennedy, J.F. Christ, Printing polymer blends through in situ active mixing during fused filament fabrication, Addit. Manuf. 36 (2020) 101233.
[86] D. Feenstra, R. Banerjee, H. Fraser, A. Huang, A. Molotnikov, N. Birbilis, Critical review of the state of the art in multi-material fabrication via directed energy deposition, Curr. Opin. Solid State Mater. Sci. 25 (4) (2021) 100924.
[87] R. Zhao, C. Chen, W. Wang, T. Cao, S. Shuai, S. Xu, T. Hu, H. Liao, J. Wang, Z. Ren, On the role of volumetric energy density in the microstructure and mechanical properties of laser powder bed fusion Ti-6Al-4V alloy, Addit. Manuf. 51 (2022) 102605.