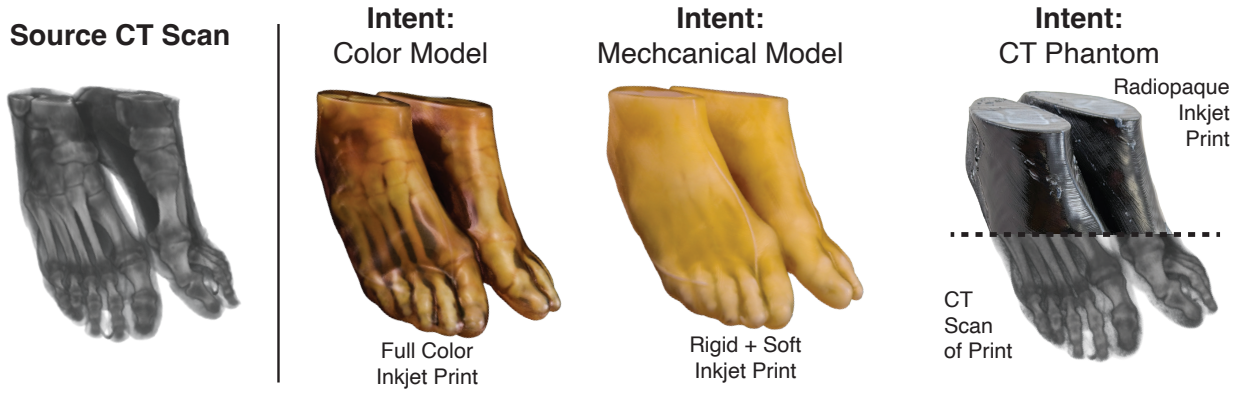
Design-Intent Compilation for Heterogeneous Fabrication

Charles Wade
charles.wade@colorado.edu
University of Colorado Boulder
Boulder, Colorado, USA
Draper Scholars, The Charles Stark Draper Laboratory, Inc
Cambridge, Massachusetts, USA

Devon Beck
dbeck@draper.com
The Charles Stark Draper Laboratory, Inc
Cambridge, Massachusetts, USA

Robert MacCurdy*
maccurdy@colorado.edu
University of Colorado Boulder
Boulder, Colorado, USA



Abstract

Computational fabrication increasingly requires designers to specify spatially varying behavior, appearance, material composition, and process state, yet most workflows encode these intentions directly in printer-specific representations. This coupling makes heterogeneous designs difficult to reuse across fabrication systems. We present a compiler architecture that treats heterogeneous fabrication as a staged, typed lowering problem. A source design consists of implicit geometry and named, typed spatial attribute fields over a shared domain. Translation models derive compiler-required realization attributes from source or intent attributes, while backend compilers emit machine-facing outputs such as voxel material stacks, process-annotated G-code, or configured slicer projects. This separation of source modeling, attribute translation, and backend compilation allows designs to remain expressed in fabrication-agnostic semantic terms while each toolchain determines their executable realization. We validate the method through fabricated objects and physical measurements spanning sampled volumetric data, CT-derived visual, mechanical, and radiographic models, Shore-hardness fields, and full-color fields, implemented through material jetting and material extrusion. Results show that one source design can be lowered into distinct material, process, and slicer representations without rewriting the design for each target. We provide the framework, compiler interface, and workflows as an open-source Python package.

Keywords

computer-aided design, additive manufacturing, volumetric design, functionally graded materials

1 Introduction

Heterogeneous additive manufacturing (AM), a strategy to enable functional-as-printed objects, requires designers to specify more than shape. Depending on the application, a printed object may need spatially varying appearance, mechanical behavior, material composition, or process state. Yet many fabrication workflows still ask designers to express these intentions in printer-specific terms: material fractions, voxel labels, mesh partitions, temperatures, or slicer settings. These quantities are necessary for fabrication, but they are poor primary representations of design intent. A printer or print preparation software may require material mixtures, discrete voxel assignments, or process-control commands to realize a design. However, a designer may want to specify a target mechanical property (e.g., stiffness or elasticity), preserve color across fabrication processes, or derive printable properties from CT radiodensity data. When the designer must author printer-specific quantities directly, the design becomes coupled to a specific fabrication backend and the intended physical property is obscured.

This coupling of fabrication process to intended properties constrains designers by requiring them to express intent in process-specific terms, a limitation further complicated by the fact that different fabrication systems may realize the same semantic attribute through different mechanisms. For example, Shore hardness can be realized through calibrated rigid–soft material fractions on

*Corresponding Author

material jetting systems or through temperature-controlled foaming on material extrusion systems. Similarly, color can be realized as voxel-scale material assignments in inkjet printing or as halftoned color regions in filament-based workflows. In each case, the printer-specific attributes differ, but the source attribute specification can remain unchanged. This observation motivates the central argument of this paper: heterogeneous fabrication should treat material mixtures and process parameters as derived attributes, not as the primary design representation.

Prior work has made heterogeneous objects increasingly representable and printable. Functional and volumetric representations describe spatially varying fields, while programmable fabrication methods produce material, voxel, or process-specific outputs [Chen et al. 2013; Pasko et al. 2001, 1995; Vidimče et al. 2016, 2013; Wade et al. 2024]. However, many existing workflows bind the design to a particular attribute type, material model, or printing system. For example, Stratasys Digital Materials workflows allow designers to request Shore hardness values and automatically compute soft–rigid material mixtures, but only within a proprietary PolyJet preparation pipeline and only for spatially uniform values. The same intent cannot be directly retargeted to another process that realizes hardness through a different physical mechanism. A more general representation should let designers specify object intent independently from the printer-specific attributes used to realize it.

We frame heterogeneous fabrication as a staged translation problem inspired by compiler-based workflows. In conventional computing, a programmer writes source code in a high-level language that expresses what a program should compute; compilers then lower that program through target-dependent representations into instructions that execute on a specific computer architecture. The programmer does not normally write machine instructions directly. Instead, the language, compiler, libraries, and architecture together determine how high-level intent becomes executable behavior. Our work applies this separation of concerns to fabrication.

This paper presents a field-based representation and compilation workflow that implements this view. A design consists of implicit geometry and named, typed spatial attribute fields over a shared object domain. A designer may directly specify attributes at either a high level, such as hardness, modulus, toughness, color, or radiodensity, or a lower level, such as material volume fractions, temperatures, flow rates, or slicer settings. Our method defines how these fields compose under modeling operations, how conflicts and undefined regions are resolved, and how derived attributes are computed from existing fields. Translation models encode calibrated, empirical, analytical, or user-defined mappings between attribute types. The resolver uses these models to derive compiler-required attributes when the source design does not provide them directly.

We validate the method through fabricated examples that test whether requested attributes survive translation and compilation. The experiments use a range of source inputs and target outputs, including analytic fields, sampled volumetric data, material-jetting outputs, color fabrication workflows, and material-extrusion process control. In several cases, we compile the same source design to distinct fabrication domains without rewriting the design in printer-specific terms. These examples show that intent-level attributes can be preserved through backend-specific realization, while also exposing the different constraints, fidelities, and artifacts of each fabrication process.

The contribution of this work is the staged translation approach to heterogeneous fabrication and the representation, resolver, and compiler interfaces that operationalize our approach. Rather than asking designers to author printer-specific material recipes or process schedules, the proposed workflow preserves intent as typed spatial fields and derives realization attributes only when a target compiler requires them. This enables design-intent-preserving cross-compilation across material jetting and material extrusion (MEX) workflows, while keeping the implementation extensible to new attribute types, translation models, and fabrication compilers. We provide an open-source Python implementation so that new attribute types, translation models, and fabrication compilers can be added by the research community.

2 Related Work

Heterogeneous additive manufacturing requires representations that describe more than boundary shape. A fabrication-ready design may contain spatially varying attributes such as material composition, color, opacity, mechanical behavior, sampled image data, process state, or slicer-level control. Prior work has developed many of these capabilities, but often under different assumptions about the properties that the designer can author and what the fabrication system can consume. Some methods emphasize expressive heterogeneous fields; others emphasize practical translation to a particular printer, material model, or voxel format. This section reviews these lines of work through the lens of the proposed representation: typed spatial attributes that can be composed, translated, and compiled to target-specific fabrication outputs.

2.1 Heterogeneous Field Representations

Early heterogeneous object models, summarized as M1 in Table 1, establish that material and property information should be represented as part of the object, not only assigned after geometry creation. Kumar and Dutta introduce a materially graded object model in which material composition functions are attached to geometric regions and represented locally as volume fractions [Kumar and Dutta 1997]. Kumar et al. generalize this view into an attribute-based object modeling framework in which geometry is the base attribute and other attributes, including material composition, microstructure, properties, and physical parameters, are defined over the object domain [Kumar et al. 1999]. These works provide an important precedent for treating heterogeneous material information as first-class modeling data. However, their fabrication-facing representation remains centered on prescribed material allocation rather than a general mechanism for translating intent-level attributes into different material systems or process parameters.

Functional and hypervolume representations, summarized as M2, are foundational implicit modeling approaches. Pasko et al. introduce functional representations for geometry, where solids are described by mathematical fields rather than boundary meshes [Pasko et al. 1995]. HyperFun and constructive hypervolume modeling extend this idea to multidimensional attributes, allowing objects to carry fields such as density, opacity, color, or material-like quantities within a constructive modeling tree [Pasko et al. 1999, 2001; Schmitt

et al. 2008]. Our work builds on these concepts of implicit modeling and heterogeneous representations. These prior works focus on rendering and theoretical heterogeneous modeling. They begin to address the challenges of designing geometry and attributes but leave a gap in translating this design to fabrication, which we address with our method.

Other heterogeneous modeling methods make spatial material variation more accessible for CAD and manufacturing workflows. Source-based heterogeneous solid modeling, summarized as M3, defines material gradation from editable point, axial, planar, and surface sources, making graded distributions easier to specify relative to geometric features [Siu and Tan 2002a,b]. Trivariate spline and subdivision-volume methods, summarized as M4, represent geometry and volumetric attributes with continuous spline or subdivision fields and develop extraction or slicing procedures for additive manufacturing [Luu et al. 2019; Martin and Cohen 2001; Sasaki et al. 2017]. GraMMaCAD, summarized as M5, provides an interactive BRep-based interface for assigning spatially varying FGM distributions to imported CAD models [Luu et al. 2022]. These systems provide important mechanisms for authoring, representing, and extracting heterogeneous material fields. However, the authored field is generally treated as the fabrication-relevant quantity itself: a material gradation, volumetric attribute, or FGM distribution to be sliced, sampled, or assigned. As a result, the representation does not clearly separate design-intent from fabrication-specific attributes, nor does it provide a general mechanism for deriving backend-specific material recipes or process parameters from a single fabrication-agnostic high-level source design. Our work addresses this gap by treating material fractions, colors, mechanical properties, sampled measurements, and process states as typed attributes that can be composed and translated according to compiler requirements.

Programmable multi-material fabrication methods, summarized as M6, move heterogeneous modeling closer to manufacturable output. OpenFab and Foundry introduce procedural volumetric material design workflows that generate printable multi-material representations [Vidimče et al. 2016, 2013]. OpenFab frames fabrication as a programmable pipeline and shows that procedural 3D textures and volumetric material descriptions can be transformed into material assignments for multi-material printing. In their workflow, however, the designer generally encodes the relationship between object behavior and material assignment directly in the design. OpenFab provides a powerful way to author heterogeneous material distributions, but it does not separate intent-level attributes from machine- or material-specific derived attributes.

Spec2Fab, summarized as M7, addresses the complementary problem of translating user-facing specifications into printable material assignments through a reducer–tuner framework [Chen et al. 2013]. It is one of the closest conceptual predecessors to our work because it recognizes that designers often want to specify perceptual or functional goals rather than low-level printer materials. Its translation, however, is organized around a comparatively monolithic optimization pipeline that couples the specification, material search, and target fabrication process to inkjet material assignment. Our work adopts the design-by-intent motivation, but represents translation models as modular field operators over typed attributes. This allows an authored field, such as shore hardness, modulus, toughness, RGBA color, or radiodensity, to be resolved differently depending on the selected compiler.

Voxel-based fabrication frameworks, summarized as M8, provide a direct route to manufacturable heterogeneous objects. VoxelFuse stores geometry and material information in voxel grids and supports multi-material design, manufacturing constraints, and simulation-oriented workflows [Brauer and Aukes 2020]. Voxels naturally encode occupancy and material labels, and many printers and simulation tools can consume sampled volumetric data. The limitations are that sampling fixes the resolution early and that voxel-based representations can degrade under affine transformations. High-resolution material jetting can require very large voxel grids, while toolpath-based processes still require additional translation from voxel labels into slices and G-code. Voxel methods therefore provide practical sampled representations, but do not by themselves preserve resolution-independent design intent or support retargeting one high-level source design to multiple realization fields.

OpenVCAD, summarized as M9, combines signed-distance geometry with implicit material volume-fraction fields for multi-material additive manufacturing [Wade et al. 2024]. Later extensions add a Python interface, simulation integration, lattice workflows, slicer-project export, and gradient-informed slicing for material extrusion [Wade et al. 2025a,b]. These works show the value of implicit geometry and implicit material fields for resolution-independent heterogeneous fabrication. Their primary heterogeneous representation, however, remains material volume fractions. Some process controls are supported, but they are introduced through a representation originally built around material fractions. Our work makes the distinction explicit: material fractions are one possible derived attribute, not a universal design representation. Typed attributes allow properties, measurements, visual fields, material recipes, and process states to coexist and be translated according to manufacturing requirements.

Table 1 summarizes these methods. The comparison is not intended to rank generality in the abstract, but to identify which capabilities were demonstrated in the cited works. Prior methods support important subsets of heterogeneous modeling and fabrication. We are not aware of a prior system that jointly demonstrates typed implicit attributes, attribute conflict semantics, attribute derivation, process attributes, and cross-fabrication compilation from a single fabrication-agnostic source design.

2.2 Programmatic and Data-Driven Authoring

Many heterogeneous and procedural design methods use programming interfaces because spatial fields are difficult to author through direct manipulation alone. HyperFun provided a language for functional representation modeling [Pasko et al. 1999]. OpenFab used a shader-like language for procedural material behavior [Vidimče et al. 2013]. OpenSCAD popularized script-based constructive solid geometry for conventional geometric CAD [Kintel 2010]. Early OpenVCAD versions likewise used a custom scripting language for implicit geometry and volume fractions [Wade et al. 2024], while later work moved the workflow into Python [Wade et al. 2025a]. These methods demonstrate that code is an expressive design representation. However, a programming interface does not by itself

Table 1: Comparison of representative heterogeneous modeling and fabrication methods. Ratings indicate demonstrated capabilities in the cited works; they are not intended to rank generality in the abstract or imply that a feature was a goal of that work. Method groups are M1: cellular and attribute object models [Kumar et al. 1999; Kumar and Dutta 1997]; M2: FRep, HyperFun, and constructive hypervolume models [Pasko et al. 1999, 2001, 1995; Schmitt et al. 2008]; M3: source-based heterogeneous solid modeling [Siu and Tan 2002a,b]; M4: trivariate spline and subdivision-volume models [Luu et al. 2019; Martin and Cohen 2001; Sasaki et al. 2017]; M5: GraMMaCAD [Luu et al. 2022]; M6: OpenFab and Foundry [Vidimče et al. 2016, 2013]; M7: Spec2Fab [Chen et al. 2013]; M8: VoxelFuse [Brauer and Aukes 2020]; and M9: OpenVCAD [Wade et al. 2025a,b, 2024]. For I1, 'Y' indicates that a usable implementation is currently obtainable through public download or by request. The comparison highlights that prior methods support important subsets of heterogeneous modeling, but do not jointly demonstrate typed implicit attributes, attribute conflict semantics, attribute derivation, process attributes, and cross-fabrication compilation.

Feature	M1	M2	M3	M4	M5	M6	M7	M8	M9	This Work
Representation features										
R1: Implicit geometry	N	Y	N	N	N	Y	N	P	Y	Y
R2: Explicit geometry / BRep	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R3: Sampled volume data	N	Y	N	Y	N	N	N	Y	Y	Y
R4: Implicit attributes	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
R5: Sampled attributes	N	P	N	Y	N	Y	Y	Y	P	Y
R6: Object / region composition	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
R7: Attribute conflict rules	Y	Y	Y	N	Y	Y	Y	Y	P	Y
R8: Attribute derivation	Y	Y	N	P	N	Y	Y	N	N	Y
Attribute-type features										
A1: Rendering attributes	N	Y	Y	Y	Y	Y	Y	N	Y	Y
A2: Physical attributes	Y	Y	N	Y	N	Y	Y	N	N	Y
A3: Material attributes	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A4: Process attributes	N	N	N	N	N	N	N	N	P	Y
Output-modality features										
O1: Rendering	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
O2: Inkjet / PolyJet: color	N	N	N	Y	N	Y	Y	N	N	Y
O3: Inkjet / PolyJet: material allocation	N	N	N	Y	Y	Y	Y	N	Y	Y
O4: Material extrusion material allocation	Y	N	N	N	Y	N	N	Y	Y	Y
O5: Process parameters	N	N	N	N	N	N	N	N	P	Y
O6: Slicer settings	P	N	P	N	N	N	N	N	P	Y
O7: Cross-fabrication	N	N	N	N	N	N	N	N	N	Y
Implementation availability										
I1: Implementation available for use	N	Y	N	N	Y	N	Y	Y	Y	Y

define the semantic structure of heterogeneous fabrication. There remains a research gap in what properties are being authored in these code-based languages and in how code is translated into manufacturable formats. Extending these prior methods, our work demonstrates the design of arbitrary high-level attributes, intermediate translation between attributes, and cross-compilation of a single design into multiple manufacturing-process specific formats.

2.3 Translation Models and Process Attributes

A design-intent workflow also requires models that translate desired properties into realizable material or process fields. One source for these models is recent material-calibration work that provides reusable local mappings from target properties to printable recipes. Smith et al., for example, introduce a data-driven mapping from mechanical targets such as modulus and toughness to digital material mixtures [Smith and MacCurdy 2024]. Such models are useful in our setting because they can be treated as field operators: if the source design contains spatially varying mechanical attributes, the same calibrated relationship can be evaluated throughout the object to produce compiler-required material fractions.

Process parameters further expand the set of attributes that a compiler may require and are therefore useful in translation models. Spatially varying behavior can arise from material assignment, but also from laser power, extrusion rate, print speed, beam focus, nozzle temperature, material mixing ratio, or flow rate, depending on the printing modality [Damanpack et al. 2021; Lalegani Dezaki et al. 2023; Loh et al. 2018; Tammaro et al. 2022; Wade et al. 2025b]. Mixing systems in material extrusion and direct ink writing likewise demonstrate that material ratio can be a controllable process state rather than only a static material label [Li et al. 2018; Liao et al. 2024; Ortega et al. 2019; Pelz et al. 2021]. These examples motivate process attributes as first-class fields in heterogeneous design. A compiler may require temperature, flow rate, mixture ratio, or slicer settings rather than a material-fraction vector, and a staged translation workflow can derive these realization attributes from higher-level intent when a valid model exists.

2.4 Voxel, Color, and Medical Image Fabrication

Material jetting and PolyJet-style systems motivate sampled output because they can assign discrete materials at droplet or voxel scale.

Color 3D printing has therefore developed sophisticated printer-dependent reproduction methods, including scattering-aware texture reproduction, geometry-aware compensation, neural acceleration of scattering prediction, and gradient-based material-assignment optimization [Elek et al. 2017; Nindel et al. 2021; Rittig et al. 2021; Sumin et al. 2019]. These methods address the final conversion from desired appearance to printer-specific material arrangements. In our workflow, color reproduction is treated as a compiler-side realization problem: RGBA color is one typed attribute that may be authored, sampled, or derived, and specialized color pipelines can then convert that attribute into process-specific voxel or slicer outputs.

Medical image-based fabrication presents a related sampled-data problem. Patient-specific models are used for pre-surgical planning, procedural rehearsal, education, and clinical communication [González-López et al. 2024; Valls-Esteve et al. 2023; Zoabi et al. 2022]. Many workflows convert CT or MRI data into segmented surfaces using marching cubes or related surface extraction methods [Bücking et al. 2017; Fogarasi et al. 2022; Kamio et al. 2020; Lorensen and Cline 1987; Mandaliana et al. 2019; Newman and Yi 2006; Wickramasinghe et al. 2020]. These workflows are practical, but they reduce volumetric image data to boundaries or labels before fabrication. Voxel-based medical printing preserves more of the original scan information and has been used to map medical images to full-color material-jetting outputs [Hosny et al. 2018; Jacobson et al. 2022a, 2023a,b, 2022b]. Our work follows the volumetric view, but treats scan intensity as a typed sampled source attribute rather than as a fixed route to one printer color space. The same input radiodensity field can be mapped to outputs such as visual attributes, mechanical attributes, or material recipes depending on the selected translation models and compiler.

2.5 Summary

Prior work provides strong components for heterogeneous fabrication: field-based representations, procedural material design, voxel fabrication, specification-to-material optimization, process-aware slicing, color reproduction, and medical image printing. The remaining gap is a representation and compilation model that makes these capabilities composable at the level of design intent. We address this gap by framing heterogeneous design as a staged compilation process in which high-level attribute specifications are lowered through interchangeable fabrication toolchains. This organization allows the same high-level design intent to be retargeted across fabrication modalities without rewriting the source design in the vocabulary of each machine.

3 Methods

We represent heterogeneous objects as spatial fields over a common domain. A source design consists of implicit geometry and a set of named, typed attribute fields. Later stages operate on this same representation: modeling operations compose fields, translation models derive additional fields when needed, and fabrication compilers sample the resolved fields into process-specific outputs. We first define the source design, since composition, translation, and compilation all depend on this mathematical framework.

3.1 A Field-Based Representation for Heterogeneous Objects

Let an object be defined by a scalar implicit geometric field

$$\phi : \mathbb{R}^3 \to \mathbb{R}, \tag{1}$$

where $p = (x, y, z) \in \mathbb{R}^3$ is a three-dimensional spatial point and $\phi(p) \in \mathbb{R}$ is its signed distance to the object's boundary. We use the convention that $\phi(p) < 0$ denotes points inside the object, $\phi(p) = 0$ denotes points on the boundary, and $\phi(p) > 0$ denotes points outside the object. The solid domain is therefore

$$\Omega = \{p \in \mathbb{R}^3 \mid \phi(p) \le 0\}. \tag{2}$$

This convention fixes the geometric meaning of all subsequent field evaluations. Objects may be authored directly from analytic signed distance functions, but the representation does not require analytic primitives. Triangular meshes and CAD boundary representations can also enter the workflow by first converting them to signed distance fields, preserving a single spatial query interface for downstream heterogeneous modeling.

A source design pairs this geometric field with a finite set of named, typed attribute fields,

$$\mathcal{A} = \{(n_i, T_i, A_i)\}_{i=1}^{m}, \tag{3}$$

where n_i is a semantic name, T_i is the return type, and

$$A_i : \Omega \to T_i \tag{4}$$

is an attribute field that returns a value at each point inside or on the object. An attribute may be implemented by an evaluator

$$\widetilde{A}_i : \Omega \times \mathbb{R} \to T_i, \tag{5}$$

such that

$$A_i(p) = \widetilde{A}_i\big(p, \phi(p)\big). \tag{6}$$

Here, the second argument is a one-dimensional scalar containing the signed distance from p to the object's boundary. This formulation allows attribute values to depend both on spatial position and on distance from the boundary while preserving A_i as a field defined over the solid domain. The name identifies the field's meaning, such as color, hardness, radiodensity, material composition, or process state. The type specifies the structure of the returned value. In this work, we demonstrate scalar attributes, three- and four-dimensional vector attributes, and material-fraction vectors, although the representation is not restricted to these types.

Attribute evaluators may be analytic, sampled, or derived. Analytic attributes are functions of position, object parameters, or geometric quantities. As shown in Figure 1, they can express parametric behavior such as Cartesian, radial, cylindrical, periodic, or boundary-dependent gradients. Sampled attributes incorporate externally generated volumetric data, such as CT or MRI volumes, image stacks, OpenVDB grids, or simulation outputs. Sampled data can be treated the same way as analytic data after interpolation defines an evaluator over Ω: it can be queried, composed, translated, and compiled. Derived attributes are produced by translation models described in Section 3.3.

This evaluator model is the representation-level abstraction. A source design is not a mesh, voxel stack, material partition, or printer program. It is a queryable field object whose geometry and heterogeneous properties share a common spatial interface, while

permitting discontinuities and definitions over multiple disjoint regions. This abstraction lets analytic design intent and sampled data coexist in one object, defers printer-specific material and process decisions until later stages, and provides the mechanism by which composition, translation, and compilation operate.

3.2 Composition Semantics for Attribute Fields

The field representation in Section 3.1 defines how a single heterogeneous object is evaluated. Design workflows, however, build objects compositionally. Designers define primitives, transform them into position, and combine them into larger structures. We represent this process as a design graph whose leaf nodes define geometric primitives, and whose interior nodes transform or combine the fields returned by their children. Evaluation at a point proceeds recursively through the graph: each node queries its children and applies a local operator to the returned geometry and attributes.

Transforms act by changing the coordinate frame in which a child is queried. Let T be a rigid transform or uniform scale, let s be the corresponding scale factor, and let c denote the child node. The transformed geometry is evaluated as

$$\phi_T(p) = s\,\phi_c(T^{-1}p), \tag{7}$$

with $s = 1$ for rigid transforms. Attributes follow the same inverse-coordinate rule:

$$A_{i,T}(p, \phi_T(p)) = A_{i,c}(T^{-1}p, \phi_c(T^{-1}p)). \tag{8}$$

Thus, transforms do not resample or rewrite the design. They only change the spatial query passed to the child evaluator. We restrict this rule to rigid transforms and uniform scaling because these operations preserve the signed-distance.

Boolean composition combines the fields returned by multiple children. For geometry, we use standard signed-distance Boolean operations for union, intersection, and difference [Pasko et al. 1995]. These operations define the composed solid domain for the node. The additional question for heterogeneous modeling is not how to combine geometry, but how attributes should behave when geometric operations combine regions that carry different fields. We handle this with two local semantic rules: conflict resolution, which applies when several children define the same attribute at the same point, and attribute availability, which records when a queried attribute is absent from part of the composed object, as discussed in the next section.

3.2.1 Hierarchy, Overrides, and Attribute Conflicts. Attributes can be attached at any node in the design graph. A parent-level attribute may introduce a new named field over the composed geometry, or override a field with the same name and type provided by its children. Parent override is the simplest hierarchical rule: if a node defines attribute a over its output domain, that field is exposed for a independent of child values. This lets a designer construct geometry from several components while assigning one attribute field over the result.

If no parent override is supplied, Boolean composition can create attribute conflicts. A conflict occurs only when multiple contributing children define attributes with the same name and type at the same point. Attributes with different names or different types do not conflict; they coexist as separate fields. Figure 2 shows this case

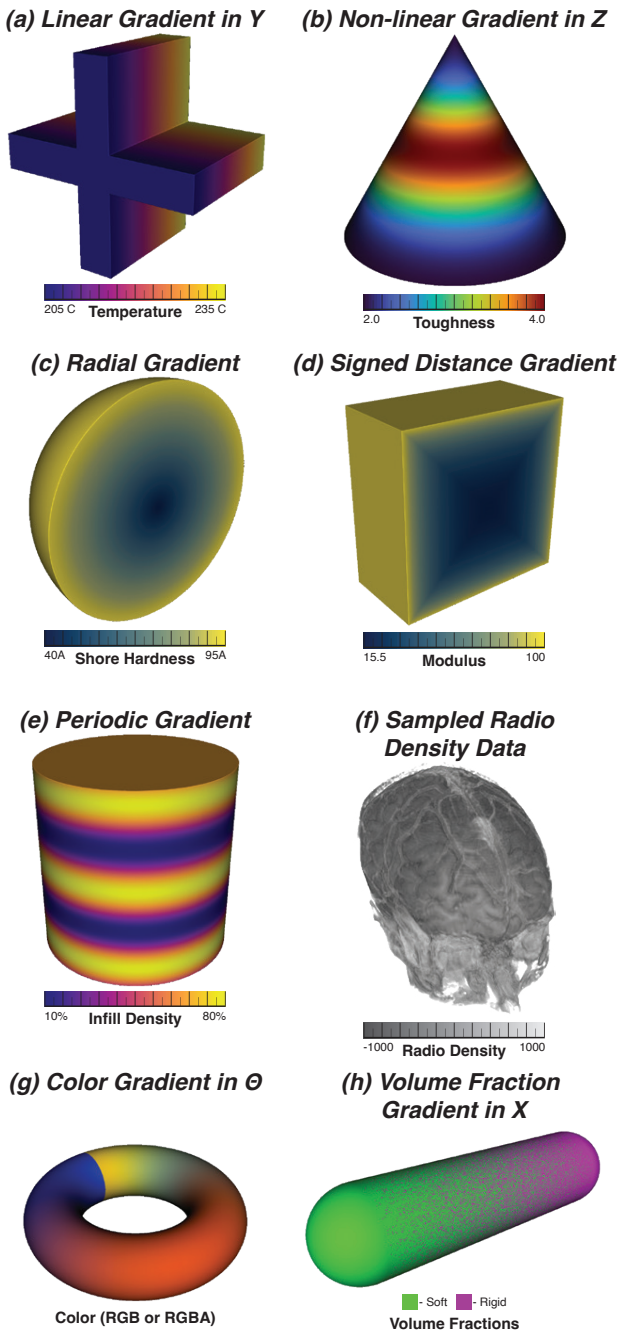


Figure 1: Representative attribute fields expressible through the proposed field representation. Analytic attributes can vary in Cartesian coordinates (a), (b), and (h); spherical coordinates (c); cylindrical coordinates (g); signed distance from the boundary (d); or periodic functions (e). Sampled attributes, such as CT-derived radiodensity (f), enter the same framework through interpolated field evaluators. The volume-fraction example in (h) is visualized as discrete material assignments for clarity, but the underlying attribute is a continuous material-fraction field. These examples illustrate possible evaluators rather than an exhaustive set of supported attributes.

for two overlapping bars whose union produces an overlap region with multiple candidate values.

We formulate conflict resolution as a typed reduction over candidate attribute samples. For an attribute identified by name and type, let

$$\mathcal{V}_a(p) = \{v_1, v_2, \ldots, v_n\} \tag{9}$$

be the values sampled from contributing children at point p. A resolver for attribute a is a function

$$v_{\text{out}} = R_a(\mathcal{V}_a(p)), \tag{10}$$

that maps the candidate samples to a single value of the same type. The resolver is invoked only when multiple contributing children define the same named and typed attribute at p.

One resolver is child-order priority: the first child, in the order added to the node, that defines the queried attribute at p supplies the value. This rule is deterministic and valid for any attribute type. Other resolvers can be selected when they match the semantics of the field, including extrema for ordered scalar values, averaging for compatible scalar or vector fields, and custom user-defined operations. Material-fraction fields may use specialized resolvers when a mixture operation is intended. Any such resolver must return a valid material-fraction vector, preserving nonnegative components that sum to one.

Conflict rules define the meaning of heterogeneous composition. The same geometric union can yield different attribute fields depending on whether the overlap selects one child, preserves an extremum, averages compatible values, or applies a custom operation. By making these rules explicit and local to the design graph, the proposed representation gives attributes compositional semantics under the same hierarchy, transforms, and Boolean operations used to construct geometry.

3.2.2 Attribute Availability and Undefined Regions. Composition can also produce regions where a queried attribute is not defined. This is distinct from a conflict. Conflict resolution chooses among multiple values; availability records whether any value exists. Because attributes are attached to nodes in the design graph, an object can be geometrically valid while carrying different attribute sets in different regions. If $\mathcal{V}_a(p)$ (as defined in Eq. 9) is empty, then attribute a is undefined at p.

Undefined attributes are a normal intermediate state during modeling. They become errors only when a downstream translation model or compiler requires the missing attribute at a point where it is undefined. This distinction allows different parts of a source design to be authored at different semantic levels and made compatible later. A designer may provide a default field, introduce a parent-level override, or apply a translation model that derives the missing attribute before the design reaches a downstream consumer.

Figure 3 illustrates this workflow with a heterogeneous dog-bone specimen. The gripping tabs are specified directly as rigid volume fractions, while the central gauge region is specified by target modulus and toughness. During authoring, the tabs do not define modulus or toughness, and the central region does not yet define volume fractions. Before compilation, the pair of mechanical attributes in the gauge region is lowered into volume fractions by a translation model, as described in Section 3.3. The tabs and gauge region then expose compatible material-fraction fields and can be combined into one material-recipe design.

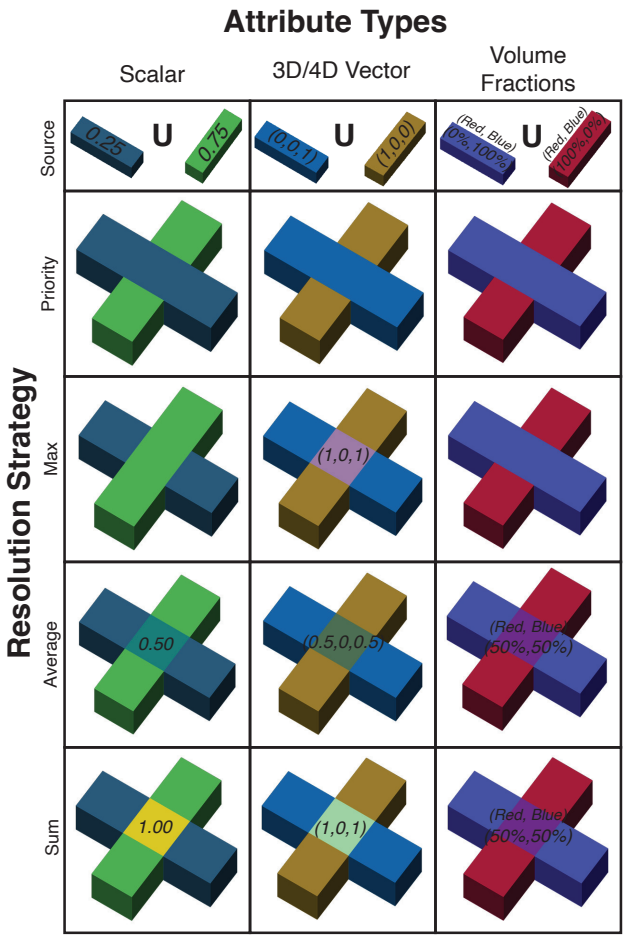


Figure 2: Attribute conflict resolution for overlapping child nodes. The source row shows two independently authored objects before union. Subsequent rows show the same geometric union resolved using priority, maximum, average, and a custom summation operation. Columns show scalar, 3D/4D vector, and material volume-fraction attributes. Conflict resolution is defined per attribute type; if no resolver is specified, child-order priority is used. The geometry is unchanged across rows, while the resolved overlap value depends on the selected semantic rule.

Making availability explicit separates geometric validity from attribute completeness. A composed object may be valid as geometry even when some attributes are unavailable in some regions. Only the attributes required by a later translation or compiler must be defined over the domain consumed by that downstream operation.

3.3 Lowering Attributes for Fabrication

The preceding sections define how attributes are represented and composed. We now define how an authored attribute becomes

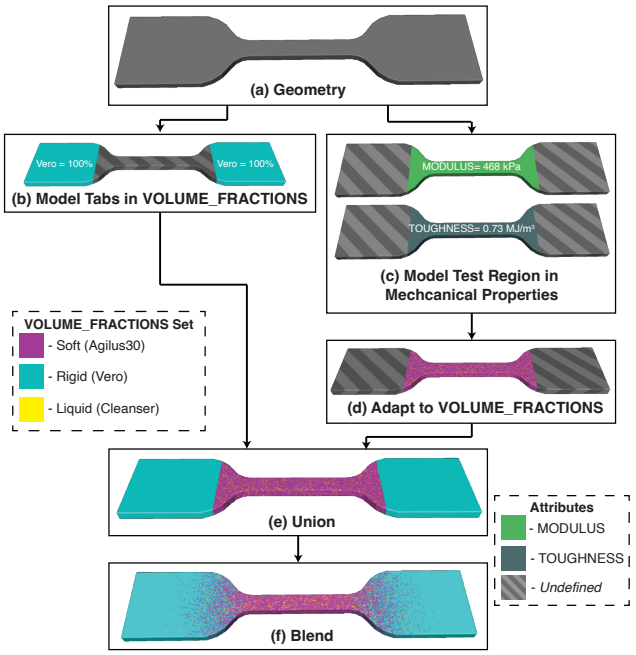


Figure 3: Region-specific attribute authoring in a single dog-bone design. The tabs are authored directly as rigid volume fractions, while the central gauge region is authored using target modulus and toughness. These choices intentionally leave some attributes undefined in some regions during authoring. Before compilation, the mechanical attributes in the gauge region are translated into volume fractions, making the regions compatible for union and blending. Undefined attributes are valid intermediate states unless they remain unresolved when required by a downstream translation or compiler.

useful to a particular fabrication process. The key distinction is between what an attribute means and how a backend realizes it.

Attributes play three roles in the lowering workflow. *Source attributes* describe measured, simulated, or externally generated data, such as CT-derived radiodensity or OpenVDB density and temperature grids. *Design-intent attributes* describe the desired object property or behavior, such as Shore hardness, modulus, toughness, or density. *Realization attributes* describe the material or process state required by a fabrication backend, such as volume fractions, temperature, flow rate, or slicer-level settings. These categories describe how an attribute is used in a workflow, not a fixed property of the name. RGBA color may be authored as visual intent, imported as source data, or consumed directly by a color-printing backend. Similarly, volume fractions often act as a realization attribute, but designers may also author them directly when fabricating calibration coupons used to measure material behavior and fit new translation models.

An attribute translation model is a field operator that derives one or more new named attributes from existing attributes. This differs from the conflict resolvers in Section 3.2.1: resolvers reduce multiple candidate values of the same attribute, while translations create new attributes from different inputs. For example, a translation may lower shore hardness into volume fractions, map radiodensity into RGBA color, or compute temperature and flow rate from a target density field. The translation node preserves the child geometry and exposes the original attributes together with any translated outputs.

Formally, let a translation model M_j define a local mapping from input attribute samples to output attribute samples. For an output attribute β, the translated evaluator is

$$A_\beta(p, \phi(p)) = M_j\Big(\{A_{\alpha_i}(p, \phi(p))\}_{i=1}^{k}\Big), \quad p \in \Omega, \tag{11}$$

where M_j is the analytical, empirical, calibrated, or user-defined translation model and $A_{\alpha_1}, A_{\alpha_2}, \ldots, A_{\alpha_k}$ denotes the set of input attributes. A model may produce multiple output attributes (e.g., $A_{\beta_1}, A_{\beta_2}, \ldots$) such that each output is evaluated as a field over the same spatial domain. This definition of translation models allows functional composition, mapping, or remapping of attributes that is propagated only when the output is queried.

Many fabrication translations encode inverse design relationships. The designer specifies a desired property field, and the translation computes the material recipe or process state expected to realize that property on a given fabrication system. For example, a calibrated material model can lower modulus and toughness into volume fractions for a multi-material inkjet workflow [Smith and MacCurdy 2024]. Other translations connect source data to design intent before fabrication-specific lowering occurs. A CT-derived radiodensity field may be mapped directly to RGBA color for visual anatomical printing, translated into mechanical attributes that are later lowered into material fractions, or converted into radiopaque material compositions for imaging-compatible phantom fabrication. In each case, translation models act as functional lowering rules between semantic fields, allowing the same source attribute to support multiple fabrication pathways.

3.3.1 Attribute Resolver. Designers may insert translation nodes manually when the intended fabrication path is known. However, manual insertion can make a source design unnecessarily backend-specific. A design authored in terms of shore hardness should not need to contain separate hard-coded translation chains for every material system and printer it might later target. To avoid this coupling, we use an attribute resolver that selects and inserts translation nodes from a registered translation graph.

The translation graph is a dependency graph over named, typed attributes and registered translation models. It is distinct from the design graph: the design graph defines spatial composition, while the translation graph defines how attributes can be lowered into other attributes. As shown in Figure 4, nodes represent single attributes or paired attribute requirements, such as modulus+toughness. Directed edges represent translation models. Each edge records its required input attributes, output attributes, model constructor, and optional workflow tags. Workflow tags restrict a translation to a particular material system, calibration context, or backend workflow, such as `via F.PLA` (foaming PLA) versus `via F.TPU` (foaming TPU).

Algorithm 1. Resolve compiler-required attributes

Input: Source design D, available attributes $\mathcal{S}$, compiler-required attributes $\mathcal{R}$, registered translations $\mathcal{M}$, workflow tags $\mathcal{G}$
Output: Adapted design D' exposing all attributes in $\mathcal{R}$

$D' \leftarrow D$;
$\mathcal{A} \leftarrow \mathcal{S}$;
while $\mathcal{R} \not\subseteq \mathcal{A}$ **do**
  $\mathcal{E} \leftarrow \{M \in \mathcal{M} \mid \text{inputs}(M) \subseteq \mathcal{A} \ \wedge \ \text{tags}(M) \text{ match } \mathcal{G}\}$;
  if $\mathcal{E} = \emptyset$ **then**
    return error: required attribute cannot be resolved;
  Select $M^* \in \mathcal{E}$ that advances an unresolved required dependency;
  Wrap D' with the translation node constructed by M^*;
  $\mathcal{A} \leftarrow \mathcal{A} \cup \text{outputs}(M^*)$;
return D';

Algorithm 1 shows this process. Given a source design and a target compiler, the resolver performs dependency resolution over the available attribute set. It begins with the attributes already exposed by the design and the attributes required by the compiler. A translation is eligible when all of its input attributes are available and its workflow tags are compatible with the requested workflow. Applying the translation adds its output attributes to the available set. The resolver continues until all required realization attributes are available, then wraps the original source design with the selected translation nodes. The result is not a new representation; it is the same queryable field object augmented with the operators needed to expose the requested realization attributes.

This dependency-resolution step is the lowering stage of the proposed workflow. The source design may contain high-level intent or source data, while the compiler declares the realization attributes it needs. The resolver inserts only the translation models needed to satisfy those requirements. If no compatible translation path exists, the design cannot be compiled for that backend without additional authored attributes or registered models.

3.4 Fabrication Compilation

The preceding sections define how source designs are represented, composed, and lowered into realization attributes. Compilation is the final stage: a compiler consumes a resolved field object and emits the format required by a target fabrication or rendering workflow. In this work, compilers are not the primary novelty. We adapt established voxel, toolpath, and slicer-project export strategies into a common interface so that each compiler can declare the attributes it requires and sample the same resolved spatial representation.

In our proposed method, each compiler declares a set of required attributes:

$$\mathcal{R}_b = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_q\}, \tag{12}$$

where b denotes the target workflow. The resolver described in Section 3.3.1 inserts translation nodes until the design exposes all attributes in $\mathcal{R}_b$. Then a compiler defines the mapping from the set of required attributes to workflow-specific output, denoted O_b:

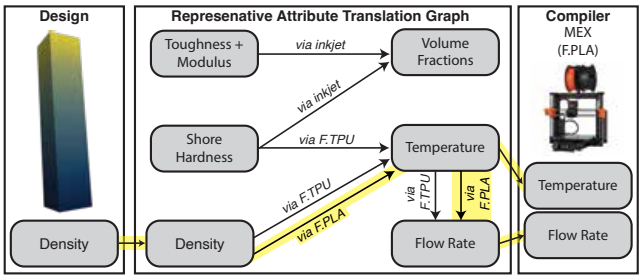


Figure 4: Attribute translation graph used to satisfy compiler-required attributes. The source design provides density, while the selected material-extrusion compiler for foaming PLA requires temperature and flow rate. Nodes represent typed attributes or paired attribute requirements, and directed edges represent registered translation models. Workflow tags, such as `via F.PLA`, `via F.TPU`, and `via inkjet`, restrict the resolver to translations compatible with the selected fabrication toolchain. The highlighted route shows the selected lowering path used to expose the attributes required by the compiler.

$$C_b : (\phi, \mathcal{R}_b) \rightarrow O_b, \tag{13}$$

The compiler then samples geometry and attributes according to the target workflow: voxels for material jetting, toolpath locations for material extrusion, or sub-mesh regions for slicer-project control.

3.4.1 Inkjet Compilation. Voxel-based inkjet and material-jetting workflows require a discrete printable material assignment at each sampled voxel or droplet location. We adapt two inkjet compiler strategies into the attribute interface defined above. Both consume resolved volume fractions or RGBA color fields and convert continuous spatial attributes into material-specific voxel stacks.

The stochastic compiler, adapted from OpenVCAD [Wade et al. 2024], samples the resolved material distribution independently at each voxel and assigns one printable material according to that distribution. This approach is simple and readily parallelizable, but sampling each location independently can produce local statistical imbalances, resulting in visible clusters of one material despite a smoothly varying target composition [Bacca Rodriguez et al. 2008]. The structured compiler instead applies three-dimensional error diffusion inspired by the work of Brunton et al. [Brunton et al. 2015; Brunton and Urban 2023]. It assigns one material per voxel while propagating quantization error through the voxel grid, which improves local agreement between the requested continuous field and the accumulated printed material distribution. As shown in Table 2, this improvement increases export time.

3.4.2 Material Extrusion Compilation. Material extrusion workflows require different compilation strategies because fabrication output is organized around paths, slicer regions, settings, and process-state changes rather than independent voxel assignments. We adapt two prior MEX compiler families into the proposed attribute interface: Gradient-Informed Toolpath Planning [Wade et al. 2025b] and

Slicer Project Control [Wade et al. 2026]. Both consume resolved spatial attributes, but they target different levels of machine control.

Gradient-Informed Toolpath Planning compiles attributes directly into process-annotated G-code. The original method encoded process states, such as temperature, indirectly as material volume fractions [Wade et al. 2025b]. We adapt the compiler to consume resolved temperature and flow rate fields directly during path generation, allowing spatial attributes to influence path construction and process scheduling without this intermediate encoding. This compiler is useful when the fabrication result depends on non-instantaneous process states, such as nozzle temperature for foaming filaments or mixture ratio for in-hotend filament mixing.

Slicer Project Control, detailed in the included pre-print, [Wade et al. 2026], targets workflows where a conventional slicer performs final toolpath generation. The compiler discretizes resolved fields into sub-meshes or slicer-control regions and exports configured project files in which each region carries the appropriate slicer-level setting. This method supports attributes such as infill density, temperature, flow rate, and RGB color while preserving access to mature slicer functionality. We use it in the results to compare cross-compiled outputs produced from the same resolved design.

These MEX compilers therefore serve complementary roles. Gradient-informed slicing provides direct control over path generation and process-state scheduling, whereas Slicer Project Control uses slicer project files to route spatial attributes through existing slicing workflows.

3.5 Cross-Compilation Across Fabrication Toolchains

Cross-compilation follows from the staged organization of the proposed workflow. Its value is that fabrication intent can be preserved independently of any particular machine or process, allowing designers to retarget, compare, or replace fabrication toolchains without rebuilding the design around each toolchain's process-specific controls. A design is authored once as geometry plus named, typed attributes. Translation models then lower those attributes into the material, process, or slicer-control fields required by a selected fabrication backend. Finally, compiler modules emit the machine-facing artifact for that backend. Unlike conventional workflows in which design intent is embedded directly in printer- or slicer-specific parameters, this structure organizes computational fabrication as a programming-oriented, multi-backend compilation workflow: designers author source-level fabrication intent, while target-specific compilers determine how that intent becomes printable instructions.

Figure 5 summarizes the complete method. The modeled object remains the source representation. Attribute translation provides the intermediate lowering layer, using either measured calibration data, inverse process models, or functional mappings. The compiler layer then consumes only the attributes required by the selected target and emits an output in the appropriate representation: voxel material stacks for inkjet printing, process-annotated G-code for gradient-informed material extrusion, or configured `.3MF` projects for slicer-based workflows. Because these stages remain separate, the same source design can be retargeted without rewriting the design in the vocabulary of each printer or slicer.

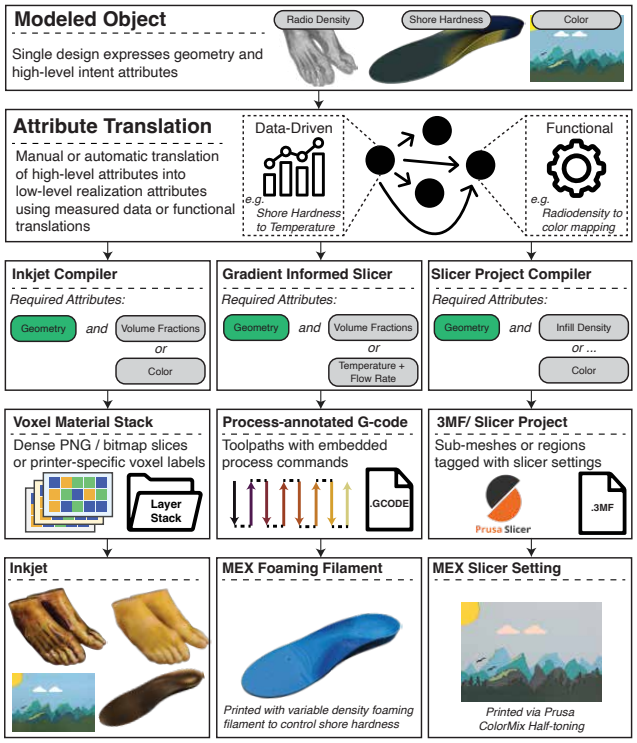


Figure 5: Overview of the proposed multi-backend compilation workflow for heterogeneous fabrication. A modeled object expresses geometry and high-level spatial attributes. Attribute translation lowers these fields into realization attributes. Backend compilers then request the attributes required by their target workflow and emit machine-facing outputs. Separating source modeling, attribute translation, and backend compilation enables the same design to be retargeted across fabrication systems.

4 Results

The results evaluate whether source designs authored in intent-level attributes can be lowered, compiled, and fabricated across distinct processes. We first specify the calibrated translation models used as reusable lowering rules in later examples. We then fabricate a set of diverse sampled volumetric attributes from simulation, medical, and industrial CT data using color inkjet compilation. These examples progress from single-process visual prints to a presurgical planning example that compiles the same sampled source data into visual, mechanical, and radiographic material-jetting outputs. Finally, we use cross-compilation examples to test whether a single source design can be realized through different material systems and fabrication toolchains, first for Shore-hardness fields and then for full-color fields. Together, these results show how attribute-level specification can preserve design intent while exposing the different constraints and artifacts introduced by each compiler and process.

Detailed calibration procedures, fitted coefficients, lookup tables, and supporting validation data are provided in the supplementary information.

4.1 Calibrated Attribute Translation Models

First, we describe the three translation models used to fabricate our results. Two are adopted from prior work, where they were developed for specific fabrication workflows rather than as components of a generalized attribute-modeling method. The third is a new inkjet material-jetting calibration introduced here. In our system, all three models serve as reusable translation layers that lower high-level physical-property fields into the material recipes or process parameters consumed by different compilers.

For mechanical-property specifications, we adapt the digital material model of Smith et al. [Smith and MacCurdy 2024] as a translation from modulus and toughness to volume fractions. The output volume-fraction vector specifies mixtures of Vero, Agilus30, and the printer's non-curing cleaning fluid. Smith et al. established an inverse design mapping between target mechanical properties and printable material recipes, and showed that the reachable property range overlaps with soft-tissue mechanical properties. This translation is used later for the CT scan of feet example in Fig. 10.

For material extrusion with foaming TPU, we adopt the calibrated process models of the pre-print included in this submission [Wade et al. 2026]. This workflow uses two linked translations. The first maps target shore hardness to nozzle temperature. The second maps temperature to a compensatory flow rate. Together, these fields form the process-attribute pair consumed by the MEX compiler to control foaming behavior while maintaining geometric accuracy. In the cross-compilation results, this model pair provides the material-extrusion realization of the same Shore-hardness intent used for inkjet.

We also introduce a new shore hardness to volume fractions model for inkjet with Vero and Agilus30. Calibration specimens were printed at prescribed Vero volume fractions from 0% to 35%, with the remaining fraction assigned to Agilus30. Five specimens were printed for each mixture. The measured calibration range spanned 40.8 to 91.0 Shore A. Because the inkjet compiler requires material fractions, we fit the inverse relationship directly, yielding a translation from target shore hardness to the volume fractions consumed by the compiler. This model provides the material-jetting lowering path used in the Shore-hardness cross-compilation result. Figure 6 shows the fitted relationship; full fitting details, coefficients, and error statistics are provided in the supplementary information.

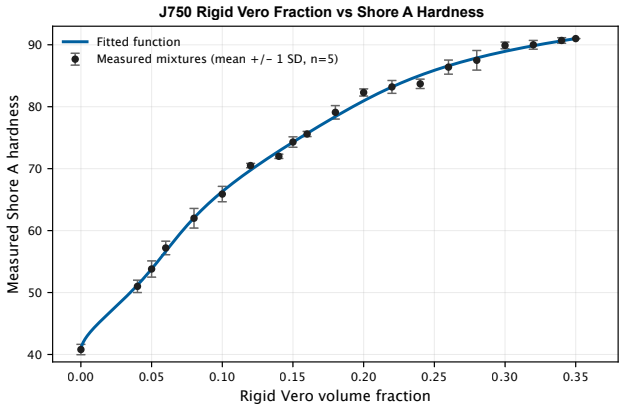


Figure 6: Inkjet Shore A hardness translation model for Vero/Agilus30 material jetting mixtures. Calibration specimens were printed with prescribed Vero volume fractions, with the remaining fraction assigned to Agilus30. Points show measured Shore A hardness values with error bars indicating one standard deviation across five specimens per mixture. The fitted inverse model defines the translation from target shore hardness to the volume fractions required by the inkjet material compiler.

4.2 Sampled Volumetric Attributes

Sampled volumes can serve as source attributes in the same lowering and compilation workflow as analytic fields. Volumetric datasets often carry spatial information on native grids, including simulation fields, graphics volumes, and CT scan data. We import these data as sampled attributes, interpolate them over the object domain, translate them to target attributes, and compile the resolved fields at the resolution required by the selected fabrication toolchain.

4.2.1 Volumes and OpenVDB. Figure 7 demonstrates this workflow with OpenVDB volumes from simulation-style datasets [JangaFX 2026; OpenVDB 2026]. Scalar density fields, and paired density–temperature fields, are mapped to RGBA color attributes and compiled using the stochastic color inkjet compiler. These examples show that volumetric data commonly used in graphics and simulation can become fabrication source attributes without first converting the volume into segmented regions or surface meshes. This example highlights the method's ability to treat sampled volumetric fields as first-class design attributes for fabrication.

4.2.2 CT Scans. Figures 8 and 9 apply the sampled-attribute workflow to CT data from medical and industrial sources. In Fig. 8, a brain CT volume is imported as a sampled radiodensity source attribute and mapped to RGBA color for color inkjet fabrication, preserving continuous variation through the tumor, surrounding tissue, and anatomical gradients. In Fig. 9, radiodensity from a CT scan of an Apple Vision Pro is mapped to color and opacity, exposing internal wires, circuit boards, cameras, and mechanical components as volumetric visual features. In both cases, the compiler operates directly on continuous scan-derived attributes rather than requiring the data to be reduced to segmented regions, labeled surfaces, or exterior geometry. These examples highlight the method's ability to treat CT volumes as fabrication-ready attribute sources across different scan domains.

The previous CT examples compile sampled radiodensity into visual attributes for color inkjet printing. In contrast, Figure 10 demonstrates how one sampled CT source can drive multiple fabrication targets through different attribute translation pathways. Starting from a single CT scan of a pair of feet, we generate three physical realizations through distinct attribute translation pathways. The first maps radiodensity to RGBA color for visual inkjet printing. The second maps radiodensity to mechanical attributes,

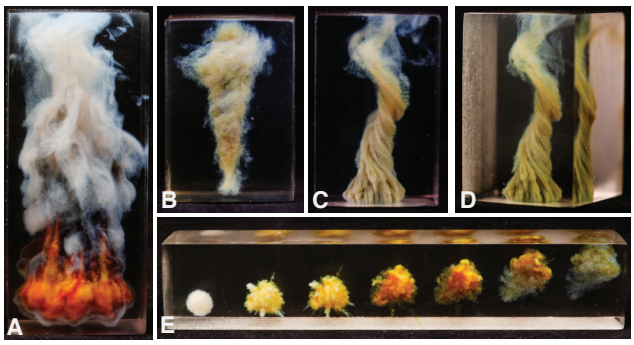


Figure 7: Fabrication of OpenVDB volumes as sampled source attributes. (A) Fire volume printed from density and temperature fields mapped to RGBA color. (B–D) Tornado and dust-devil volumes printed from density fields mapped to color and opacity. (E) Explosion sequence printed from density and temperature fields. Each sampled volume is compiled using a stochastic color inkjet compiler, demonstrating that simulation-style volumetric data can be resampled into printer-resolution material assignments.

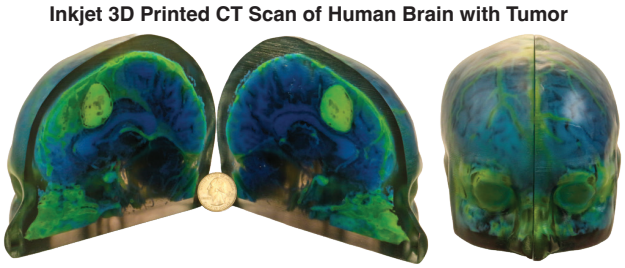


Figure 8: Inkjet 3D print of a human brain CT volume with a visible tumor. The CT data is imported as a sampled radiodensity attribute and mapped to RGBA color for stochastic color inkjet compilation. The printed result preserves continuous radiodensity gradients through the volume rather than reducing the scan to segmented surface meshes.

modulus and toughness, using anatomical-region property targets consistent with the ranges reported by Smith et al. [Smith and MacCurdy 2024] and HU-to-anatomy mappings [Schneider et al. 2000]. These mechanical attributes are then lowered to material volume fractions using Smith's model and compiled for material jetting. The third pathway maps the radiodensity field into radiopaque material compositions using the RadioMatrix/Vero material relationship reported by Stratasys [Stratasys 2025]. This translation converts the sampled CT attribute into a spatial mixture of radiopaque and non-radiopaque PolyJet materials, producing a physical phantom whose internal contrast is expressed under CT imaging. The printed object appears largely opaque under visible light, but a subsequent microCT scan reveals attenuation gradients that reproduce the structure of the original source volume. Together, these examples

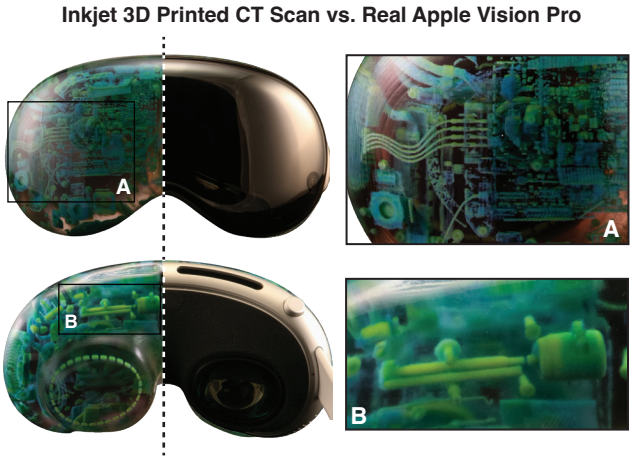


Figure 9: Volumetric inkjet print of an Apple Vision Pro CT scan compared with the physical artifact. The normalized radiodensity volume is mapped to color and opacity attributes, allowing internal electronic and optical structures to remain visible in the printed model. Inset (A) highlights internal wires, circuit boards, and a camera. Inset (B) shows a motor and lead screw used to adjust the interpupillary distance of the displays. An interactive side-by-side comparison is available in the online project gallery.

demonstrate that a single sampled source attribute can be lowered into different fabrication intents, including visual appearance, mechanical behavior, and imaging response.

4.3 Cross-Compilation Across Fabrication Toolchains

Cross-compilation demonstrates an advantage of separating design intent from fabrication-specific realization. We evaluate whether the same source attributes can be realized through different physical mechanisms, compiler requirements, and output formats. This separation is valuable not only across fundamentally different fabrication processes, but also across variants of the same process. For example, material-jetting systems may differ in available materials, voxel encodings, calibration relationships, supported mixing strategies, and machine-facing file formats, particularly across manufacturers and printer generations. By retaining attributes at the source level, these differences can be handled through target-specific translation models and compilers rather than by reformulating the design for each machine.

First, we cross-compile Shore hardness fields to inkjet, where hardness is realized by Vero/Agilus30 material fractions, and to material extrusion, where hardness is realized by foaming TPU process control. Second, we cross-compile an RGB color field to material-jetting and filament-based color workflows. These examples test whether source designs can remain expressed in intended spatial attributes while translation models and compilers resolve the target-specific realization details.

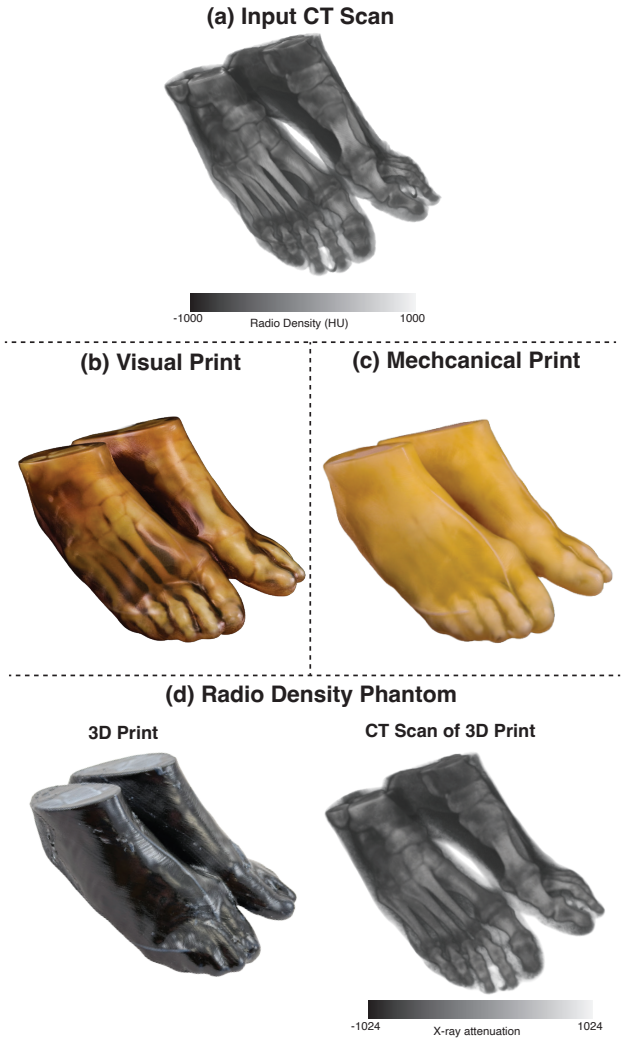


Figure 10: CT-derived multi-attribute fabrication from the same paired-feet scan. (a) Source CT volume imported as a sampled radiodensity attribute. (b) Inkjet print mapped to visual properties through RGBA color. (c) Inkjet print mapped to mechanical properties, including modulus and toughness, which are translated into material volume fractions. (d) Radiopaque PolyJet phantom generated by translating the CT radiodensity field into VeroBlack/RadioMatrix material mixtures. The left image shows the printed object under visible light, while the right image shows the reconstructed microCT attenuation volume, demonstrating preservation of spatial radiodensity variation. The example demonstrates multi-stage translation from one sampled volumetric source attribute into distinct fabrication outputs. An interactive](https://matterassembly.org/OpenVCAD-Docs/v3/gallery.htmlinteractive) side-by-side comparison is available in the online project gallery.

4.3.1 Shore-Hardness Cross-Compilation. Figure 11 evaluates Shore-hardness cross-compilation from a single high-level attribute specification to two fabrication workflows. The source design for the calibration specimen was specified once as a linear shore hardness field ranging from 65A to 85A over a 100 mm × 30 mm × 10 mm bar. The gradient start and end positions were offset 12 mm from the specimen edges along the x-axis, allowing measurements to be taken within the geometric constraints of the Shore A protocol. Eleven locations were sampled at regular intervals along the gradient.

For inkjet, the translation graph lowered shore hardness to volume fractions, and the stochastic inkjet compiler converted the resulting Vero/Agilus30 material fractions into discrete voxel assignments. For material extrusion, the same shore hardness field was lowered using the foaming TPU models detailed in the included pre-print [Wade et al. 2026] to compute temperature and compensatory flow-rate fields.

Both fabricated bars reproduced the designed hardness profile closely. The plotted values in Fig. 11a are the average measurements at each location. Mean absolute error against the requested hardness profile was 0.3A for inkjet and 0.5A for foaming TPU material extrusion. This agreement shows that the same design-intent field can be resolved into different realization attributes while preserving the requested functional gradient.

Figure 11b applies the same workflow to a full-scale insole. The source design specifies a spatial shore hardness field with a compliant 65A heel region, stiffer 85A arch and lateral-midfoot support regions, and an intermediate 75A region elsewhere, with smooth transitions between zones. From this single source field, the system computes the workflow-specific attributes needed for each compiler: temperature and flow rate for foaming TPU material extrusion, and soft/rigid material fractions for inkjet. Together, the calibration bar and insole demonstrate cross-compilation from one functional specification to distinct fabrication processes, separating modeling intent from translation and compiler-specific realization.

4.3.2 Color Cross-Compilation. Figure 12 evaluates color cross-compilation using a mountain-scene image as a compact benchmark for comparing visual reproduction across fabrication toolchains. The rendered source design was imported as an RGB color attribute and extruded into a planar 3D object. The same source design was compiled across two printing modalities and four color reproduction workflows. For material jetting, we evaluated structured error-diffusion dithering and stochastic compiler output. For fused filament fabrication on a five-toolhead system, we used the Prusa ColorMix and OrcaSlicer FullSpectrum results reported in the pre-print included in this submission [Wade et al. 2026]. We reproduce these FFF conditions here as comparison points for evaluating cross-compilation across fabrication modalities.

We quantified color fidelity by photographing each sample under fixed camera settings, lighting conditions, and alignment. Each photograph was registered to the digital reference and compared in CIELAB space using CIEDE2000 color difference, ΔE_{00}. Before computing ΔE_{00}, both the reference and sample images were Gaussian filtered in CIELAB space with $\sigma = 12$ pixels. This metric therefore emphasizes regional color reproduction rather than droplet-, bead-,

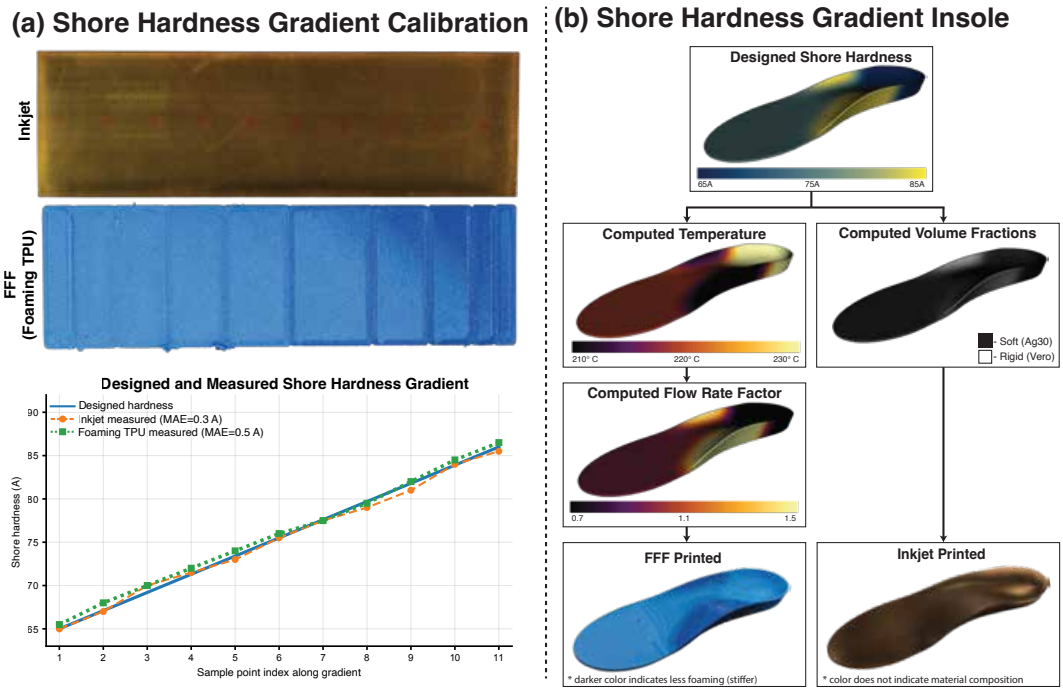


Figure 11: Shore-hardness cross-compilation. (a) Linear 65A–85A calibration bar fabricated by inkjet and foaming TPU material extrusion, with measured profiles compared against the requested hardness field. Foaming TPU results are reported in the pre-print included in this submission [Wade et al. 2026] and are provided for comparison. (b) Application-scale insole compiled from one shore hardness field to workflow-specific realization attributes for foaming TPU material extrusion and inkjet.

or pixel-scale variation. We separately measured print-induced high-frequency artifacts using an excess texture residual. This residual isolates high-frequency variation after removing the low-frequency color component, then subtracts the corresponding residual of the reference image so that the reported value emphasizes texture introduced by fabrication rather than texture already present in the source image.

The error maps in Fig. 12 show clear workflow-dependent tradeoffs. Inkjet produced the lowest regional color errors, with mean $\Delta E_{00} = 3.19$ for structured dithering and $\Delta E_{00} = 3.14$ for stochastic output. These workflows therefore achieved similar regional color accuracy. However, structured dithering produced lower excess texture than stochastic output, with mean texture residuals of 0.76 and 0.95, respectively. This indicates that deterministic error diffusion better suppresses local grain in this example, while stochastic compilation provides a simpler and lower-cost computational alternative evaluated further in Section 4.4.

The MEX outputs reported in the pre-print included in this submission [Wade et al. 2026] produced larger color errors and higher texture residuals than the material-jetting outputs, reflecting the lower spatial resolution, bead-based deposition, and more limited local color control of the evaluated filament workflows. Prusa ColorMix achieved a mean ΔE_{00} of 11.73 and a mean texture residual of 3.04, while OrcaSlicer FullSpectrum produced a mean ΔE_{00} of 16.63 and a mean texture residual of 6.69. We include these previously reported measurements to expose process-dependent tradeoffs rather than to present them as independent experimental replications. Material jetting provided higher regional color fidelity and lower texture for this benchmark, while FFF provided a lower-cost and more accessible route to full-color fabrication through slicer-based filament halftoning.

Across the Shore-hardness and color demonstrations, cross-compilation preserves attribute-level intent while exposing workflow-dependent realization quality. The hardness examples show that functional intent can be realized across different material mechanisms with small measured errors. The color examples show that visual intent can be retargeted across different color reproduction workflows, while preserving measurable differences in fidelity and texture. Together, these results demonstrate that source designs can be portable across fabrication toolchains without implying that all workflows realize the design with identical accuracy, resolution, or artifacts.

4.4 Compilation Cost

We measured export time to assess whether resolved attribute fields can be compiled at fabrication-relevant scales. Each benchmark reports the wall-clock time required to take a resolved design and export it to the corresponding output format. Resolver time was negligible relative to export time, which is dominated by sampling the resolved geometry and attributes into the target representation, quantizing or discretizing fields as required by the compiler, and writing the output files. Benchmarks were run as single executions on an Apple M5 MacBook Pro using 10 CPU threads.

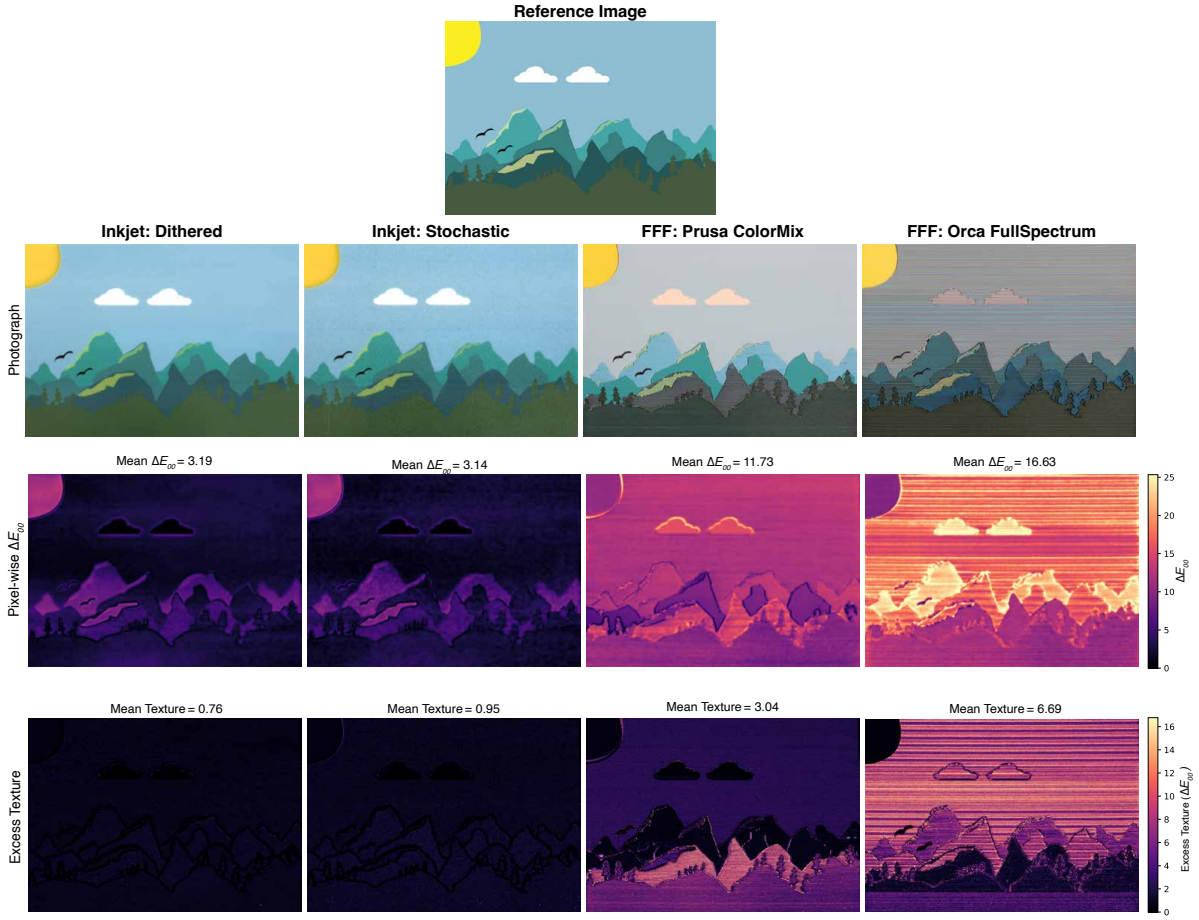


Figure 12: Color error and texture residuals for the mountain scene compiled across material-jetting and FFF workflows. Row three reports ΔE_{00} after Gaussian filtering in CIELAB space, emphasizing regional color accuracy. Row four reports excess texture residuals, emphasizing print-induced high-frequency variation. The material-jetting workflows produced the lowest regional color errors and texture residuals, while the FFF workflows introduced larger color deviations and stronger layer- or halftone-induced texture. Mean values are reported in each panel. The FFF: Prusa ColorMix and FFF: Orca FullSpectrum photographs, error maps, texture maps, and quantitative measurements were originally reported the pre-print included with this submission[Wade et al. 2026] and are reproduced here as comparison points for evaluating cross-compilation across fabrication modalities.

Table 2 reports bounding-box size, total sampled points, export time, and sample throughput for the examples used in the results. The benchmarks consist of the objects from the results section. The demonstrated exports range from $2.17e7$ to $3.66e10$ samples. Stochastic inkjet compilation scales to tens of billions of samples for large CT-derived prints, including the paired-feet and Apple Vision Pro examples. Smaller OpenVDB volumes export in approximately one to thirteen minutes. The paired-feet mechanical print is marginally slower than the paired-feet color print at the same sample count because the mechanical workflow evaluates a more expensive translation path before material assignment.

The mountain-scene benchmark isolates the cost of compiler choice because the inkjet stochastic and dithered outputs use the same object size and sample count. Stochastic inkjet export required 3.3 min for $1.64e9$ samples, while dithered inkjet export required 5.6 min for the same sample count. This difference reflects the tradeoff observed in the color-quality results: structured dithering reduced local texture artifacts, but its error-diffusion procedure introduced additional export cost. The MEX color workflows sampled far fewer points because they operate at slicer-project resolution rather than inkjet voxel resolution. Their throughput values are therefore not directly comparable to inkjet throughput; these exports are dominated by mesh construction and `.3MF` project generation rather than dense voxel sampling.

These measurements are provided only as general references for computational scale, not as direct performance comparisons. Earlier programmable multi-material fabrication systems addressed different problem formulations and did not support the cross-compilation

Table 2: Export-time benchmarks for the demonstrated examples. *Samples* denotes the number of spatial evaluations used by the compiler, independent of whether the target output is a voxel stack, process-annotated toolpath, or slicer-project representation. Throughput is reported in millions of samples per second. These benchmarks primarily characterize whether the compiler supports practical fabrication workflows; in the demonstrated cases, export time is dominated by the number of spatial samples evaluated rather than by the geometric size of the object alone.

Example	Compiler	Bounding box (mm)	Samples	Export (min)	Samples/s (10^6)
Fire, Fig. 7A	Inkjet–Stochastic	$100 \times 42 \times 43$	1.86e9	11.4	2.72
Tornado, Fig. 7B	Inkjet–Stochastic	$49 \times 33 \times 34$	5.79e8	2.0	4.83
Dust devil, Fig. 7C	Inkjet–Stochastic	$58 \times 38 \times 38$	8.57e8	2.6	5.56
Explosion, Fig. 7E	Inkjet–Stochastic	$157 \times 30 \times 25$	1.20e9	13.3	1.51
Brain CT, Fig. 8	Inkjet–Stochastic	$154 \times 111 \times 110$	1.95e10	28.20	11.50
Apple Vision Pro CT, Fig. 9	Inkjet–Stochastic	$183 \times 95 \times 178$	3.20e10	22.2	24.00
Feet CT, color, Fig. 10B	Inkjet–Stochastic	$159 \times 131 \times 169$	3.66e10	45.6	13.40
Feet CT, mechanical, Fig. 10C	Inkjet–Stochastic	$159 \times 131 \times 169$	3.66e10	49.4	12.30
Insole, inkjet, Fig. 11	Inkjet–Stochastic	$289 \times 104 \times 30$	9.44e9	16.6	9.49
Insole, MEX, Fig. 11	MEX–Gradient-informed	$290 \times 105 \times 31$	1.19e8	1.2	1.72
Mountains, Fig. 12B	Inkjet–Stochastic	$149 \times 106 \times 10$	1.64e9	3.3	8.32
Mountains, Fig. 12C	Inkjet–Dithered	$149 \times 106 \times 10$	1.64e9	5.6	4.87
Mountains, Fig. 12D	MEX–ColorMix	$149 \times 106 \times 10$	2.17e7	17.9	0.02
Mountains, Fig. 12E	MEX–FullSpectrum	$149 \times 106 \times 10$	2.17e7	17.4	0.02

of a common attribute-based source design across multiple fabrication backends. OpenFab reported full-volume synthesis for 0.12B–14.2B voxels at 300 DPI, including a 1.8B-voxel butterfly in 33 min and a 1.2B-voxel marble table in 25 min [Vidimče et al. 2013]. The closest inkjet-scale examples in Table 2, including the 1.64B-sample mountain object and the 1.20B-sample explosion volume, fall within a similar sample-count regime. The larger CT-derived examples extend to tens of billions of samples. Although differences in representation, implementation, hardware, and supported workflows preclude direct benchmarking, these references indicate that the proposed workflow operates at fabrication-relevant scales while additionally supporting typed attributes, sampled source data, modular translation models, and multiple compiler targets.

5 Conclusion

Heterogeneous additive manufacturing needs representations that preserve design intent without forcing designers to author printer-specific material recipes, process parameters, or slicer settings directly. This paper establishes typed attribute fields as a design-intent layer for heterogeneous fabrication. Geometry is represented as an implicit object domain, while visual, measured, mechanical, material, process, and slicer-level quantities are represented as named, typed spatial fields over that domain. This separation lets designers specify what an object should express, while translation models and compiler modules determine how that intent is realized for a selected fabrication toolchain.

The central contribution is a staged workflow for heterogeneous design and fabrication. Typed attributes provide a common spatial interface for heterogeneous information. Composition rules define how these fields behave under hierarchy, transforms, Boolean operations, and overlapping regions. Translation models lower source or intent attributes into realization attributes when valid calibrated, empirical, analytical, or user-defined mappings exist. Compiler modules then request the attributes they require and sample the resolved fields into process-specific outputs, including voxel material stacks, process-annotated G-code, and slicer-project files. Together, these components separate source design, attribute realization, and machine-specific output.

The results show the value of this separation. Sampled CT and OpenVDB volumes enter the same workflow as analytic fields and can be translated into visual, mechanical, or radiographic attributes for fabrication. We also demonstrate cross-compilation: one shore hardness field is realized through both inkjet material fractions and foaming-TPU process control, and one RGB color field is realized through both material-jetting and filament-based color workflows. These examples show that the source design can remain stable while the realization attributes, compiler outputs, and fabrication mechanisms change. They also show that portability does not erase process-specific behavior; accuracy, texture, resolution, cost, and fabrication artifacts remain measurable consequences of the selected workflow.

Our results and examples reflect available models and compilers, but the method is extensible to additional model and compiler types. New applications require material data, process calibrations, analytical mappings, or learned models that relate intent attributes to realizable material or process fields. They also require compiler modules for additional fabrication processes. Future work should therefore expand translation-model libraries, add compiler targets beyond the inkjet and material-extrusion workflows demonstrated here, and accelerate high-resolution compilation with GPU implementations. These extensions would make the approach applicable to broader domains, including pre-surgical planning models, soft

robotics, functionally graded materials, and multifunctional structures.

We provide the representation, translation framework, compiler interface, and demonstrated examples as an open-source Python package implementation. The goal is to make the system extensible: researchers can add attribute types, sampled-data sources, calibrated mappings, and fabrication compilers as new materials and processes become available. By organizing heterogeneous fabrication around typed spatial intent rather than fixed material recipes, this work provides a foundation for reusable design specifications, measurable process-specific realization, and reproducible research in multi-material and functionally graded additive manufacturing.

Acknowledgments

This material is based upon work supported by the Charles Stark Draper Laboratory, Inc. under Contract No. N00030-24-C-6001. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Strategic Systems Programs.

References

J. Bacca Rodriguez, G.R. Arce, and D.L. Lau. 2008. Blue-Noise Multitone Dithering. *IEEE Transactions on Image Processing* 17, 8 (Aug. 2008), 1368–1382. doi:10.1109/TIP.2008.926145

Cole Brauer and Daniel M. Aukes. 2020. Automated Generation of Multi-Material Structures Using the VoxelFuse Framework. *Proceedings - SCF 2020: ACM Symposium on Computational Fabrication* (Nov. 2020). ISBN: 9781450381703. doi:10.1145/3424630.3425417

Alan Brunton, Can Ates Arikan, and Philipp Urban. 2015. Pushing the Limits of 3D Color Printing: Error Diffusion with Translucent Materials. *ACM Transactions on Graphics* 35, 1 (Dec. 2015), 1–13. doi:10.1145/2832905

Alan Brunton and Philipp Urban. 2023. Cuttlefish: Pushing the Limits of Graphical 3-D Printing. *IEEE Computer Graphics and Applications* 43, 5 (Sept. 2023), 114–121. doi:10.1109/MCG.2023.3298173

Thore M. Bücking, Emma R. Hill, James L. Robertson, Efthymios Maneas, Andrew A. Plumb, and Daniil I. Nikitichev. 2017. From medical imaging data to 3D printed anatomical models. *PLOS ONE* 12, 5 (May 2017), e0178540. doi:10.1371/journal.pone.0178540

Desai Chen, David I. W. Levin, Piotr Didyk, Pitchaya Sitthi-Amorn, and Wojciech Matusik. 2013. Spec2Fab: a reducer-tuner model for translating specifications to 3D prints. *ACM Transactions on Graphics* 32, 4 (July 2013), 1–10. doi:10.1145/2461912.2461994

A. R. Damanpack, André Sousa, and M. Bodaghi. 2021. Porous PLAs with Controllable Density by FDM 3D Printing and Chemical Foaming Agent. *Micromachines* 12, 8 (July 2021), 866. doi:10.3390/mi12080866

Oskar Elek, Denis Sumin, Ran Zhang, Tim Weyrich, Karol Myszkowski, Bernd Bickel, Alexander Wilkie, and Jaroslav Křivánek. 2017. Scattering-aware texture reproduction for 3D printing. *ACM Transactions on Graphics* 36, 6 (Dec. 2017), 1–15. doi:10.1145/3130800.3130890

Magdalene Fogarasi, James C. Coburn, and Beth Ripley. 2022. Algorithms used in medical image segmentation for 3D printing and how to understand and quantify their performance. *3D Printing in Medicine* 8, 1 (Dec. 2022), 18. doi:10.1186/s41205-022-00145-9

Pablo González-López, Artem Kuptsov, Cristina Gómez-Revuelta, Jaime Fernández-Villa, Javier Abarca-Olivas, Roy T. Daniel, Torstein R. Meling, and Juan Nieto-Navarro. 2024. The Integration of 3D Virtual Reality and 3D Printing Technology as Innovative Approaches to Preoperative Planning in Neuro-Oncology. *Journal of Personalized Medicine* 14, 2 (Feb. 2024), 187. doi:10.3390/jpm14020187

Ahmed Hosny, Steven J. Keating, Joshua D. Dilley, Beth Ripley, Tatiana Kelil, Steve Pieper, Dominik Kolb, Christoph Bader, Anne-Marie Pobloth, Molly Griffin, Reza Nezafat, Georg Duda, Ennio A. Chiocca, James R. Stone, James S. Michaelson, Mason N. Dean, Neri Oxman, and James C. Weaver. 2018. From Improved Diagnostics to Presurgical Planning: High-Resolution Functionally Graded Multimaterial 3D Printing of Biomedical Tomographic Data Sets. *3D Printing and Additive Manufacturing* 5, 2 (2018), 103–113. doi:10.1089/3dp.2017.0140

Nicholas Jacobson, Erik Carerra, Lawrence Smith, Lorna Browne, Nicholas Stence, Alison Sheridan, and Robert MacCurdy. 2022a. Defining Soft Tissue: Bitmap Printing of Soft Tissue for Surgical Planning. *3D Printing and Additive Manufacturing* 9, 6 (Dec. 2022), 461–472. doi:10.1089/3dp.2021.0141

Nicholas M. Jacobson, Jane Brusilovsky, Robert Ducey, Nicholas V. Stence, Alex J. Barker, Max B. Mitchell, Lawrence Smith, Robert MacCurdy, and James C. Weaver. 2023a. The Inner Complexities of Multimodal Medical Data: Bitmap-Based 3D Printing for Surgical Planning Using Dynamic Physiology. *3D Printing and Additive Manufacturing* 10, 5 (Oct. 2023), 855–868. doi:10.1089/3dp.2022.0265

Nicholas M. Jacobson, Erik Carerra, Aaron Treat, Megan McDonnell, David Mathes, and Christodoulous Kaoutzanis. 2023b. Hybrid modeling techniques for 3D printed deep inferior epigastric perforator flap models. *3D Printing in Medicine* 9, 1 (Sept. 2023), 26. doi:10.1186/s41205-023-00181-z

Nicholas M. Jacobson, Lawrence Smith, Jane Brusilovsky, Erik Carrera, Hayden McClain, and Robert MacCurdy. 2022b. Voxel Printing Anatomy: Design and Fabrication of Realistic, Presurgical Planning Models through Bitmap Printing. *Journal of Visualized Experiments* 180 (Feb. 2022), 63214. doi:10.3791/63214

JangaFX. 2026. Free VDB Volumes Created With EmberGen. https://jangafx.com/software/embergen/download/free-vdb-animations. Accessed: 2026-06-22.

Takashi Kamio, Madoka Suzuki, Rieko Asaumi, and Taisuke Kawai. 2020. DICOM segmentation and STL creation for 3D printing: a process and software package comparison for osseous anatomy. *3D Printing in Medicine* 6, 1 (Dec. 2020), 17. doi:10.1186/s41205-020-00069-2

Marius Kintel. 2010. OpenSCAD - The Programmers Solid 3D CAD Modeller.

V. Kumar, D. Burns, D. Dutta, and C. Hoffmann. 1999. A framework for object modeling. *Computer-Aided Design* 31, 9 (Aug. 1999), 541–556. doi:10.1016/S0010-4485(99)00051-2

Vinod Kumar and Debasish Dutta. 1997. Solid model creation for materially graded objects. *1997 International Solid Freeform Fabrication Symposium* (1997).

Mohammadreza Lalegani Dezaki, Mahdi Bodaghi, Ahmad Serjouei, Shukri Afazov, and Ali Zolfagharian. 2023. Soft Pneumatic Actuators with Controllable Stiffness by Bio-Inspired Lattice Chambers and Fused Deposition Modeling 3D Printing. *Advanced Engineering Materials* 25, 6 (March 2023), 2200797. doi:10.1002/adem.202200797

Wenbin Li, Austin J. Martin, Benjamin Kroehler, Alexander Henderson, Tieshu Huang, Jeremy Watts, Gregory E. Hilmas, and Ming C. Leu. 2018. Fabricating Functionally Graded Materials by Ceramic On-Demand Extrusion with Dynamic Mixing. doi:10.26153/TSW/17109

Enze Liao, Yaling Zhang, Yu Su, Chenyang Zhang, Chengzhen Geng, Changlin Li, Xiaoyan Liu, Yu Liu, and Ai Lu. 2024. Active mixing of two-component viscoelastic silicone ink at molecular level for spatiotemporally controlled 3D/4D printing of cellular silicones. *Polymer* 302 (May 2024), 127053. doi:10.1016/j.polymer.2024.127053

Giselle Hsiang Loh, Eujin Pei, David Harrison, and Mario D. Monzón. 2018. An overview of functionally graded additive manufacturing. *Additive Manufacturing* 23 (Oct. 2018), 34–44. doi:10.1016/j.addma.2018.06.023

William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics* 21, 4 (Aug. 1987), 163–169. doi:10.1145/37402.37422

Thu Huong Luu, Christian Altenhofen, Tobias Ewald, André Stork, and Dieter Fellner. 2019. Efficient slicing of Catmull–Clark solids for 3D printed objects with functionally graded material. *Computers & Graphics* 82 (Aug. 2019), 295–303. doi:10.1016/j.cag.2019.05.023

Thu Huong Luu, Christian Altenhofen, André Stork, and Dieter Fellner. 2022. GraMMaCAD: Interactively Defining Spatially Varying FGMs on BRep CAD Models. In *Volume 2: 42nd Computers and Information in Engineering Conference (CIE)*. American Society of Mechanical Engineers, St. Louis, Missouri, USA, V002T02A014. doi:10.1115/DETC2022-89552

Khairani Arlinda Mandaliana, Tri Harsono, and Riyanto Sigit. 2019. 3D visualization and reconstruction of lung cancer images using marching cubes algorithm. In *2019 International Electronics Symposium (IES)*. IEEE, 143–147.

William Martin and Elaine Cohen. 2001. Representation and extraction of volumetric attributes using trivariate splines: a mathematical framework. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*. ACM, Ann Arbor Michigan USA, 234–240. doi:10.1145/376957.376984

Timothy S Newman and Hong Yi. 2006. A survey of the marching cubes algorithm. *Computers & Graphics* 30, 5 (2006), 854–879.

Thomas Klaus Nindel, Tomáš Iser, Tobias Rittig, Alexander Wilkie, and Jaroslav Křivánek. 2021. A gradient-based framework for 3D print appearance optimization. *ACM Transactions on Graphics* 40, 4 (Aug. 2021), 1–15. doi:10.1145/3450626.3459844

OpenVDB. 2026. OpenVDB Download. https://www.openvdb.org/download/. Accessed: 2026-06-22.

Jason M. Ortega, Melody Golobic, John D. Sain, Jeremy M. Lenhardt, Amanda S. Wu, Scott E. Fisher, Lemuel X. Perez Perez, Adam W. Jaycox, James E. Smay, Eric B. Duoss, and Thomas S. Wilson. 2019. Active Mixing of Disparate Inks for Multimaterial 3D Printing. *Advanced Materials Technologies* 4, 7 (July 2019), 1800717. doi:10.1002/admt.201800717

Alexander Pasko, V Adzhiev, R Cartwright, E Fausett, A Ossipov, and V Savchenko. 1999. HyperFun project: a framework for collaborative multidimensional F-rep modeling. In *Eurographics/ACM SIGGRAPH Workshop Implicit Surfaces' 99*. Workshop Implicit Surfaces 1999, 59–69.

Alexander Pasko, Valery Adzhiev, Benjamin Schmitt, and Christophe Schlick. 2001. Constructive Hypervolume Modeling. *Graphical Models* 63, 6 (Nov. 2001), 413–442.

doi:10.1006/gmod.2001.0560
Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. 1995. Function representation in geometric modeling: concepts, implementation and applications. *The visual computer* 11 (1995), 429–446. doi:10.1038/s41467-017-01210-2
Joshua S. Pelz, Nicholas Ku, William T. Shoulders, Marc A. Meyers, and Lionel R. Vargas-Gonzalez. 2021. Multi-material additive manufacturing of functionally graded carbide ceramics via active, in-line mixing. *Additive Manufacturing* 37 (Jan. 2021), 101647. doi:10.1016/j.addma.2020.101647
Tobias Rittig, Denis Sumin, Vahid Babaei, Piotr Didyk, Alexey Voloboy, Alexander Wilkie, Bernd Bickel, Karol Myszkowski, Tim Weyrich, and Jaroslav Křivánek. 2021. Neural Acceleration of Scattering-Aware Color 3D Printing. *Computer Graphics Forum* 40, 2 (May 2021), 205–219. doi:10.1111/cgf.142626
Yuhi Sasaki, Masahito Takezawa, Seungki Kim, Hiroshi Kawaharada, and Takashi Maekawa. 2017. Adaptive direct slicing of volumetric attribute data represented by trivariate B-spline functions. *The International Journal of Advanced Manufacturing Technology* 91, 5-8 (July 2017), 1791–1807. doi:10.1007/s00170-016-9800-0
Benjamin Schmitt, Alexander Pasko, and Christophe Schlick. 2008. Constructive Hypervolume Modeling Using Extended Space Mappings. In *Heterogeneous Objects Modelling and Applications*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Alexander Pasko, Valery Adzhiev, and Peter Comninos (Eds.). Vol. 4889. Springer Berlin Heidelberg, Berlin, Heidelberg, 167–192. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-540-68443-5_7
Wilfried Schneider, Thomas Bortfeld, and Wolfgang Schlegel. 2000. Correlation between CT numbers and tissue parameters needed for Monte Carlo simulations of clinical dose distributions. *Physics in Medicine and Biology* 45, 2 (Feb. 2000), 459–478. doi:10.1088/0031-9155/45/2/314
Y.K. Siu and S.T. Tan. 2002a. Modeling the material grading and structures of heterogeneous objects for layered manufacturing. *Computer-Aided Design* 34, 10 (Sept. 2002), 705–716. doi:10.1016/S0010-4485(01)00200-7
Y.K. Siu and S.T. Tan. 2002b. 'Source-based' heterogeneous solid modeling. *Computer-Aided Design* 34, 1 (Jan. 2002), 41–55. doi:10.1016/S0010-4485(01)00046-X
Lawrence T. Smith and Robert B. MacCurdy. 2024. Digital Multiphase Composites via Additive Manufacturing. *Advanced Materials* 36, 34 (Aug. 2024), 2308491. doi:10.1002/adma.202308491
Stratasys. 2025. *RadioMatrix Lab Report.* White Paper. Stratasys Ltd. https://cdn-dev.goengineer.com/stratasys-radiomatrix-lab-report-white-paper.pdf
Denis Sumin, Tobias Rittig, Vahid Babaei, Thomas Nindel, Alexander Wilkie, Piotr Didyk, Bernd Bickel, Jaroslav Křivánek, Karol Myszkowski, and Tim Weyrich. 2019. Geometry-aware scattering compensation for 3D printing. *ACM Transactions on Graphics* 38, 4 (Aug. 2019), 1–14. doi:10.1145/3306346.3322992
Daniele Tammaro, Massimiliano Maria Villone, and Pier Luca Maffettone. 2022. Microfoamed Strands by 3D Foam Printing. *Polymers* 14, 15 (Aug. 2022), 3214. doi:10.3390/polym14153214
Arnau Valls-Esteve, Núria Adell-Gómez, Albert Pasten, Ignasi Barber, Josep Munuera, and Lucas Krauel. 2023. Exploring the Potential of Three-Dimensional Imaging, Printing, and Modeling in Pediatric Surgical Oncology: A New Era of Precision Surgery. *Children* 10, 5 (May 2023), 832. doi:10.3390/children10050832
Kiril Vidimče, Alexandre Kaspar, Ye Wang, and Wojciech Matusik. 2016. Foundry: Hierarchical material design for multi-material fabrication. *UIST 2016 - Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Oct. 2016). ISBN: 9781450345316. doi:10.1145/2984511.2984516
Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. 2013. OpenFab: a programmable pipeline for multi-material fabrication. *ACM Transactions on Graphics* 32, 4 (July 2013), 1–12. doi:10.1145/2461912.2461993
Charles Wade, Devon Beck, and Robert MacCurdy. 2025a. Implicit Modeling for 3D-printed Multi-material Computational Object Design via Python. In *Proceedings of the ACM Symposium on Computational Fabrication*. ACM, Cambridge MA USA, 1–16. doi:10.1145/3745778.3766659
Charles Wade, Devon Beck, and Robert MacCurdy. 2025b. Implicit toolpath generation for functionally graded additive manufacturing via gradient-informed slicing. *Additive Manufacturing* 112 (Aug. 2025), 104963. doi:10.1016/j.addma.2025.104963
Charles Wade, Devon Beck, and Robert MacCurdy. 2026. Functionally Grading the Slicing Process by Compiling Design Intent into Slicer Projects. *arXiv preprint arXiv:2607.25326* (2026).
Charles Wade, Graham Williams, Sean Connelly, Braden Kopec, and Robert MacCurdy. 2024. OpenVCAD: An open source volumetric multi-material geometry compiler. *Additive Manufacturing* 79 (2024), 103912. doi:10.1016/j.addma.2023.103912
Udaranga Wickramasinghe, Edoardo Remelli, Graham Knott, and Pascal Fua. 2020. Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, Anne L. Martel, Purang Abolmaesumi, Danail Stoyanov, Diana Mateus, Maria A. Zuluaga, S. Kevin Zhou, Daniel Racoceanu, and Leo Joskowicz (Eds.). Vol. 12264. Springer International Publishing, Cham, 299–308. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-030-59719-1_30
Adeeb Zoabi, Idan Redenski, Daniel Oren, Adi Kasem, Asaf Zigron, Shadi Daoud, Liad Moskovich, Fares Kablan, and Samer Srouji. 2022. 3D Printing and Virtual Surgical Planning in Oral and Maxillofacial Surgery. *Journal of Clinical Medicine* 11, 9 (April 2022), 2385. doi:10.3390/jcm11092385

6 Appendix A: Code and Data Availability

All examples presented in this work, together with the Python and C++ implementations, are available at https://matterassembly.org/openvcad. OpenVCAD supports Windows, macOS, and Linux and can be installed using `pip install OpenVCAD`.