Implicit Slicing for Functionally Tailored Additive Manufacturing

Conference Paper · August 2016

DOI: 10.1115/DETC2016-59638

3 authors:

John Steuben
United States Naval Research Laboratory
40 PUBLICATIONS 88 CITATIONS

Athanasios Iliopoulos
United States Naval Research Laboratory
104 PUBLICATIONS 511 CITATIONS

John G Michopoulos
United States Naval Research Laboratory
226 PUBLICATIONS 1,383 CITATIONS

Some of the authors of this publication are also working on these related projects:

- The Science of Customizing Deposition Processes for Novel Materials View project
- Thin film characterization View project

All content following this page was uploaded by John Steuben on 09 October 2018.
ABSTRACT
One crucial component of the additive manufacturing software toolchain is a class of geometric algorithms known as “slicers.” The purpose of the slicer is to compute a parametric toolpath defined at the mesoscale and associated g-code commands, which direct an additive manufacturing system to produce a physical realization of a three-dimensional input model. Existing slicing algorithms operate by application of geometric transformations upon the input geometry in order to produce the toolpath. In this paper we introduce an implicit slicing algorithm that computes mesoscale toolpaths from the level sets of heuristics-based or physics-based fields defined over the input geometry. This enables computationally efficient slicing of arbitrarily complex geometries in a straightforward fashion. The calculation of component “infill” is explored, as a process control parameter, due to its strong influence on the produced component’s functional performance. Several examples of the application of the proposed implicit slicer are presented. It is demonstrated–via proper experimentation–that the implicit slicer can produce a mesoscale structure leading to objects of superior functional performance such as greatly increased stiffness and ultimate strength without an increase of mass. We conclude with remarks regarding the strengths of the implicit approach relative to existing explicit approaches, and discuss future work required in order to extend the methodology.

INTRODUCTION
Additive manufacturing (AM), also known as layered manufacturing, rapid prototyping, or less formally as 3D printing, is an increasingly important family of fabrication techniques for the production of a wide variety of components. These fabrication techniques are characterized by successive additions of material to a domain, as opposed to the repeated subtractions that are employed by most traditional fabrication technologies [1]. Recent years have seen a surge of interest in additive manufacturing technology from a broad number of engineering and manufacturing disciplines. Besides the emergence of inexpensive consumer grade systems, this interest is primarily driven by the opportunity of the relative freedom from geometric constraints provided by additive manufacturing methods; geometries that are difficult or impossible to produce by conventional means are often readily achievable via AM.

At the present time, a variety of AM technologies exist. Common techniques include stereolithography [2], Fused Deposition Modeling (FDM) [3], Selective Laser Sintering (SLS) [4–6], Electron Beam Melting (EBM) [7], and Direct Metal Deposition (DMD) [8,9]. The mechanical details of these processes vary considerably, but they share a common software toolchain, sometimes referred to as the “digital thread.” A block diagram of the AM-specific digital thread concept is shown in Fig. 1. It should be noted that the conventional usage of the term “digital thread” in the AM community refers only to CAD modeling, STL representation, and transmission to the 3D-printer. What is presented in Fig. 1 represents our own extended definition of the term “digital thread.” Because of the highly integrated nature of the additive manufacturing software implementations produced by AM hardware manufacturers, most of the individual components of the digital thread (usually those that are executed on the runtime system of the 3D-printer itself) are frequently invisible and not accessible by the end-users.

As Fig. 1 shows, the AM-specific digital thread is subdivided into three major stages; a design environment, a preprocessing environment, and a manufacturing environment. The
digital thread begins in the design environment, and originates from a Computer-Aided Design (CAD) model produced by a designer. The ultimate goal of the additive manufacturing process is to produce this model within acceptable constraints on accuracy, time, cost, and other parameters. Within the design environment, this geometry is converted to a triangular mesh form, typically encoded as a stereolithography (STL) or an additive manufacturing file (AMF), or similar file. It is important to note that this conversion preserves approximate geometric information regarding the original model only. Any other ancillary information encoded within the original model is lost in this process. Moving to the preprocessing stage, the position of the resultant mesh within the build volume of the additive manufacturing process is determined by a layout optimization routine. In practice, a collection of many meshes is packed into the build volume of the machine in order to reduce per-unit production costs, using a method such as that presented in [11]. The mesh (or collection of meshes) is then processed by an algorithm known as “slicer.” The purpose of the slicer is to subdivide the mesh(es) into a series of distinct layers, and to compute the numeric control (NC) commands issued to the additive manufacturing machine in order to produce the distinct toolpaths making up each layer. The build layout and slicer tools are often combined into a single commercial software product, that largely behaves as a “black box.” Once the toolpath has been produced and the proper G-code sequences have been generated, the motion control software and hardware systems present in the manufacturing environment are used to drive the additive manufacturing machine in order to produce the output object.

The various stages of the additive manufacturing digital thread have been studied and developed for decades, and in many senses have reached a high level of development sophistication [12, 13]. One important shortcoming of the current state of the art is the aforementioned loss of design information that occurs at the interface between the design environment and the preprocessing environment. This effectively reduces the additive manufacturing process to a purely geometric exercise. While this is acceptable for established uses of additive manufacturing such as the production of “look and feel” prototypes, it is of primary importance to ensure that the AM-produced parts can satisfy their functional performance requirements in terms of exhibiting specific functional properties. Examples of functional properties include yield and ultimate strengths, elastic anisotropy constants, residual strains, and thermal or electrical conductivities. Initial efforts documenting such activities can be found in [14, 15].

The primary focus of this paper is the development of a new type of slicing algorithm that enables control of the mesoscale topology of the toolpaths in a manner that enables control of the functional performance of the final part. The motivation for this development is to reduce the deficiency of design data in the preprocessing domain, and facilitate ongoing efforts to develop functionally imbued additively manufactured objects. In order to do so, we adopt a fundamentally different approach to solving the slicing problem. Unlike existing algorithms, that operate on the basis of explicit geometric transforms applied to the input geometry, we employ a novel implicit method based on the computation of level sets of field functions. We explore the use of field functions defined upon these regions in order to re-introduce design intent into the preprocessing environment. In particular, we develop a methodology by which the results of a physics-based performance specification that can be driven by a Finite Element Analyses (FEA) may be used to dictate the computation of toolpaths in order to improve functional performance fields of interest such as strain and stress distributions, generated within additively manufactured components.
BACKGROUND

Slicers are a class of algorithms in the domain of computational geometry that are used to convert input 3D geometry into a series of motion commands (a “toolpath”) for an additive manufacturing machine. The slicer is required to both process the input geometry into a suitable toolpath for additive manufacturing, and export this toolpath as a series of numeric control (NC) commands (e.g. G-code sequences) that are subsequently conveyed to the additive manufacturing hardware. As the second stage of this process depends heavily on the specific additive manufacturing process and device employed, we restrict our discussion to that of toolpath generation. Figure 2 demonstrates a hypothetical toolpath generated by a slicing algorithm.

Slicers and Associated Terminology

Beyond the similarities to existing NC toolpath generation, there are several unique aspects of additive manufacturing slicing that render it a distinct research field. A review of early work in this vein may be found in [16–19]. Various improvements to these early algorithms, generally termed “adaptive” slicing can be found in [20–25]. Efforts to improve the results of slicing, for instance by eliminating voids in the infill portion of the toolpath, are explored by [26, 27]. Recent research into slicing, for instance the work of [28, 29], has primarily been focused on increasing computational performance. Additionally, some work including [18, 29] has focused on “direct” slicers, that bypass the model triangularization step and convert the input CAD geometry directly into toolpaths. Most recently, image-space approaches to slicing have emerged [30, 31], that operate on point-cloud or volumetric image data in a reverse-engineering context. One such slicer utilizes General-Purpose Graphics Processing Unit (GPGPU) computing [32]. Additionally, slicers intended to produce objects with spatially-varying stiffness properties have also been developed [33].

A review of the available literature reveals that existing approaches may be broadly characterized as explicit transformations applied to input geometry that is explicitly defined (although the work of [30, 31] considers implicit input geometry). Figure 3 illustrates the variable definitions associated with slicing algorithms. We denote the input geometric domain \( \Omega \in \mathbb{R}^3 \), and its boundary \( \Gamma \). A series of slicing planes \( \mathcal{P} \) are intersected with \( \Omega \) in order to create planar subdomains \( \Omega_i = \Omega \cap \mathcal{P} \) that are bounded by \( \gamma_i \). The subdomains and boundaries associated with the \( i^{th} \) slicing plane \( \mathcal{P}_i \) are thus denoted \( \Omega_i \) and \( \gamma_i \), respectively. The explicit slicers discussed earlier can be represented by the form

\[
g = \bigcup_{i=0}^{n_l} \bigcup_{j=0}^{n_o} F_j(\gamma_i)
\]

where \( g \) represents the singleton set containing the output toolpath, and \( F_j \) are the various geometric transformations applied to the boundary \( \gamma_i \) of each layer. The number of layers is \( n_l \) and the number of transformation operations per layer is \( n_o \). To provide a convenient framework for elucidating the similar and dissimilar aspects of explicit and implicit slicers, we introduced the operator \( \cup \) to express a special combination of the union operator for bags (in the set-theoretic context) with an ordering optimization from the time efficiency perspective. That is to say that \( \cup \) orders the results of the \( F_j(\gamma_i) \) transformations so that \( g \) may be produced by the additive manufacturing system in minimal time. This optimization is beyond the scope of the present work, but further information on the topic can be found in [34].

As seen in Eq. (1), The previously mentioned slicing algorithms typically operate on \( \gamma \) only, due to its direct availability by computing simple triangle-plane intersections. While this is a sensible choice when the purpose of slicing is viewed from a strictly geometric viewpoint, it is problematic from the standpoint of producing components that imbue a functional purpose. The functional responses of interest, for example mechanical stress, are defined as fields over \( \Omega \) and thus over \( \omega \). In order to develop a slicing procedure that incorporates these fields, and thus allows the re-introduction of design intent into the digital thread, we turn towards an implicit method detailed in the next section.

PROPOSED METHODOLOGY

One method by which relevant functional fields may be incorporated into the slicing procedure is by the adoption of an implicit methodology. We consider the formulation

\[
g = \bigcup_{i=0}^{n_l} \bigcup_{j=0}^{n_o} \bigcup_{k=0}^{n_k} H_j(x) = c_k, x \in \omega_k
\]

where \( H_j \) are the field functions, \( x \) is a position vector of a point in the volume \( \Omega \) defined in \( \mathbb{R}^3 \), and \( c_k \) is a discrete value of the field functions defining the \( k^{th} \) level set. Our implementation of this formulation is represented by the multi-step procedure.
FIGURE 4: Sequence of operations for the implicit slicer.

shown in Fig. 4. The steps outlined in this figure are detailed in the sections below.

Model Parsing and Preparation
In the present work a standard STL file is used for input, due to its widespread use in the additive manufacturing industry. The algorithms presented in the following sections are trivially adaptable to other input types. The only precondition placed on the STL input files is that it must describe a manifold or finite set of manifolds, without gaps or degenerate geometric features.

The input STL file explicitly defines \( \Gamma \) as a set of triangular facets \( (F_j) \), each defined by three vertices in \( \mathbb{R}^3 \).

\[
\Gamma = \bigcup_{j=0}^{n_f} F_j = \bigcup_{j=0}^{n_f} \{ \mathbf{v}_{j1}, \mathbf{v}_{j2}, \mathbf{v}_{j3} \} \tag{3}
\]

where \( n_f \) is the number of facets and \( \mathbf{v} \) are the vectors describing the vertex coordinates. Because of the limitations of the STL file format, that does not contain any long range ordering of the facets nor any inter-facet connectivity information, \( \Gamma \) is known only in a piecewise, discontinuous fashion.

Computing Layer Boundaries
The subdomain boundaries \( \gamma \) are computed directly from \( \Gamma \), by sequential intersections with the slicing planes \( P_i \).

\[
\gamma = \Gamma \cap P_i = \bigcup_{j=1}^{n_f} P_i \cap F_j \tag{4}
\]

The triangle-plane intersection algorithm of [35] is utilized to compute these intersections. Upon the completion of these intersection tests, \( \gamma \) is a set of line segments without organization. This renders it difficult to directly reconstruct \( \omega \) from \( \gamma \); without further organization \( \gamma \) contains no information regarding the topology of \( \omega \). This difficulty is illustrated in Fig. 5, that shows that some subsets of \( \gamma \) define outer boundaries of \( \omega \), while others define internal “holes.”

To enable the recovery of \( \omega \), \( \gamma \) is partitioned into contiguous ordered subsets denoted \( \gamma_j \) as shown in Fig. 5. This is trivially equatable to the problem of finding the connected components of a graph. We parse the lines and edges of \( \gamma \) to an edge-list graph and apply the classical algorithm of [36], that is based on depth-first searching. While this algorithms operates in linear time on any graph, it is particularly fast in the present application because the graph defined by \( \gamma \) exclusively contains vertices of order two only. At the conclusion of this operation, the \( \gamma_j \) are cohesively ordered loops of line segments. From these loops, we proceed to reconstruct the corresponding domain.

Computing Layer Domains
Once \( \gamma \) has been properly organized, the reconstruction of the domain \( \omega \) may be achieved using several well-known algorithms. In the present work, we utilize constrained Delaunay triangulation for this purpose. This approach is given in detail by [37, 38], implemented in several high-quality freely available software libraries, [39,40], and also available in commonly-used computer algebra tools such as Mathematica and Matlab. The result of computing the \( \omega \) corresponding to the \( \gamma_j \) of Fig. 5 is shown in Fig. 6. The procedure outlined in this, and the previous, section is repeated for every slicing plane in order to compute all elements of \( \omega \). Once this process has been completed, these domains are defined as a set of planar facets \( F \). These domains are used to bound the implicit field functions and compute the slicer toolpath, as described in the next section.
Implicit Field Computation

Once the boundaries and domains for each layer have been defined, the additive manufacturing machine toolpath is determined by the calculation of level sets of implicit functions upon these domains and boundaries. Equation (2) gives the most general form of this procedure, in practice a decomposed form is used

\[ g = g_{pr} \bigcup g_{in} \bigcup g_{other} \bigcup \ldots \]  

(5)

where again \( \bigcup \) indicates an operator representing the optimized interleaved sequencing of toolpath segments. \( g_{pr} \) and \( g_{in} \) are associated with the perimeter and infill portions of the toolpath, as seen in Fig. 2. The \( g_{other} \) term reflects other toolpath components, such as support material, that are commonly required but often manufacturing technology specific. In the present work, we focus on the perimeter and infill toolpaths. For simplicity, we examine the case where the perimeters and infill are dictated each dictated by one field function and an associated set of levels

\[ g_{pr} \equiv \bigcup_{i=0}^{n_i} \bigcup_{k=0}^{n_k} H_{pr}(x) = c_{ki} x \in \omega_i \]  

(6)

\[ g_{in} \equiv \bigcup_{i=0}^{n_i} \bigcup_{k=0}^{n_k} H_{in}(x) = \hat{c}_{ki} x \in \omega_i \]  

(7)

In general, several different fields could be weighted and combined to weigh different design functional objectives in order to produce either \( g_{pr} \) and \( g_{in} \). In the following sections, we discuss the formulation of these functions, and the associated selection of values for \( c_{ki} \) in greater detail.

Computing Perimeter Shells

The most fundamental requirement of the slicer is that the output toolpath must replicate the geometry of the input model. Naively interpreted, this suggests that

\[ H_{pr}(x) = c_{0i} x \in \gamma_i \]  

(8)

Since the shape of \( \gamma \) must be strongly encoded in \( H_{pr} \), the signed distance transform is the most logical selection for this function. With this choice,

\[ H_{pr}(x) = \begin{cases} 
\min \| x - x_{yi} \|, & x_{yi} \in \gamma_i \quad x \in \omega_i \\
- \min \| x - x_{yi} \|, & x_{yi} \in \gamma_i \quad \text{otherwise}
\end{cases} \]  

(9)

Figure 7 illustrates the perimeters generated by this approach when applied to a portion of the domain seen in Fig. 6. Four perimeters are computed, with \( c_k = \{0.0, 0.5, 1.0, 1.5\} \) mm. It is seen that the implicit formulation can handle cases of vanishing and subdividing inner perimeters without any difficulty. In practice, it may not be desirable to run the outermost perimeter at \( H_{pr}(x) = 0.0 \). For instance, in FDM additive manufacturing, the polymer extrusion has a width \( \varepsilon \). In this case, a sensible scheme is to choose \( c_k = \varepsilon \{1/2, 3/2, \ldots \} \).

Computing Volumetric Infill

While the form of \( H_{pr} \) is necessarily dictated by \( \gamma \), the infill function \( H_{in} \) is not similarly constrained. This is the principal advantage of the implicit methodology, as the infill may be dictated by an arbitrary field of functional or design interest. Several examples are given in Fig. 8. Parts (a) and (b) of Fig. 8 show simple closed-form functions that replicate infill strategies commonly employed by existing commercial slicers. Parts (c) and (d) show alternate closed-form selections, that demonstrate the flexibility of the implicit approach. Part (e) shows an infill pattern based on a stochastic pseudo-random number generator, and part (f) shows an infill pattern based on the solution to Poisson’s equation with a Dirichlet boundary condition on \( \gamma \).

The infill patterns of Fig. 8 are representative of one layer \( \omega_i \) of the entire toolpath. It is typical to modulate the infill pattern between layers, for instance to rotate the linear infill by 90° between layers. In Fig. 8 it is assumed that the \( \omega_i \) are parallel to the xy-plane. In this frame of reference, the modulation would take the form of a functional dependence of \( H_{in} \) on \( z \).

The only restriction on \( H_{in} \) is that it must be a function that

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

Approved for public release; distribution is unlimited.
returns a real-valued result. This opens the possibility of using the infill to couple the performance properties of the additively manufactured component to some field of engineering or design interest. For example, the stress or strain fields computed by FEA could be used as the basis for the infill geometry. This is illustrated in greater detail in the ‘Experimental Validation’ section. However, there are several other details of the toolpath generation that must be discussed first.

**Computing Contours of \( H_{in} \)** Due to the widespread use of contour plots in a great variety of computer tools, as well as the usage of level sets in simulation methods such as [41], algorithms for computing level sets and contours of functions/data are very well developed. Various fast algorithms for the calculation of contours have been developed [42–44], and there have been some recent improvements in such algorithms [45]. These algorithms are embodied in virtually all modern computer algebra tools, as well as in libraries for most computer programming languages.

Any of the existing algorithms for computing contour plots may be employed to calculate the contours of \( H_{in} \) in order to produce the corresponding toolpath components. In the present work, the following method is employed. Recalling the definition of \( \omega \), the layer domains are each encoded as a set planar facets \( \hat{F} \) in \( \mathbb{R}^3 \). We take \( H_{in} \) as a transform

\[
T : \mathbb{R}^2 \rightarrow \mathbb{R}^3, T(x) = (x_1, x_2, H_{in}(x_1, x_2)), x \in \omega
\]

This transform is applied to the vertex coordinates of the facets in \( \omega \), effectively carrying the transformation of \( \hat{F} \) to \( F \). The procedure for intersecting slicing planes with \( F \) is employed to compute the contours in \( \mathbb{R}^3 \) in precisely the same fashion as the original computation of \( \gamma \). The slicing planes are positioned at the values of \( c_k \). The resultant series of line segments are then projected back into the slicing plane via

\[
T : \mathbb{R}^3 \rightarrow \mathbb{R}^2, T(x) = (x_1, x_2), x \in \omega
\]

This process is illustrated in Fig. 9.
Collecting Layers & Output
Once the perimeter and infill toolpath components have been computed individually for each layer, they must be assembled into a single cohesive NC program. The first stage of this process is to ensure that the infill and perimeter toolpaths for each layer are compatible and do not overlap. This is achieved by pruning the infill contours and discarding any portions of the infill that fall outside of the contracted domain

\[ \omega' = x[H_{pr}(x)] > \max(c_k) + \varepsilon \]  

(12)

where \( \varepsilon \) is a small constant that introduces a minor overlap or clearance between infill and perimeters. This variable accommodates the physics of some additive manufacturing processes; for instance in FDM it is typical to overlap the infill and perimeters slightly in order to realize improved structural performance. The sequencing of the contours within each layer is a general combinatorial optimization problem that is constrained by the physics of the additive manufacturing process that is being targeted by the slicer. Large difference in machine movement speed, as well as different times associated with stopping, repositioning, and restarting the deposition of material/energy, dictate that optimal toolpaths for different additive manufacturing technologies will be structured in different fashions. For the purposes of demonstrating the implicit approach in the present work, we use a simple scheme for ordering the toolpath components, as opposed to solving the full optimization problem. For every layer:

1. Begin with the outermost perimeter, traverse it in a clockwise fashion, starting/ending at a randomly selected point.
2. Select the nearest point on the next perimeter, traverse in clockwise fashion. Repeat for all perimeters.
3. Select the endpoint of an infill curve that is nearest to the start/end point of the final perimeter. Traverse it to the other endpoint.
4. Select the endpoint of an untraversed infill curve that is closest to the current endpoint. Traverse this new infill curve. Repeat until all infill curves are traversed.

Once these steps have been completed for every layer, the final output toolpath is simply the sequence of these layers from first to last (typically from lowest to highest in the z-direction). A process-specific, possibly machine-specific parser must be used to translate the purely numerical toolpath into a format understood by the additive manufacturing machine, but such developments are beyond the scope of the present work.

Support Structure
Many components that are additively manufactured require so-called “support material.” This support material is deposited by the additive manufacturing process outside of the component domain, in order to prevent the sagging or curling of the component itself under gravitational or other forces during the manufacturing process. The proposed implicit methodology may also be used to compute the toolpaths associated with the support material, in a very simple fashion. The support material domain \( \omega_s \) for any layer is defined as the difference between a layer’s region \( \omega_l \) and the union of all layer regions above \( \omega_l \). The distance transform may be used to compute perimeters for this support region, and any function may be used to compute the support region infill, in a fashion identical to that described previously.

Demonstration Results
A prototype of the implicit slicer described in the previous section was implemented in the Mathematica symbolic mathematics system [46]. A laptop computer with a dual-core processor and 4 GB of memory was used to execute this program and conduct the slicing procedure. Two specific test geometries were sliced using this implementation and hardware in order to demonstrate the capabilities of the implicit slicing methodology.

Test Case 1: Engine Cylinder Head
The goal of the first test problem is to apply the implicit slicer to a highly complex input geometry, and to demonstrate that by using a linear function for \( H_{pr} \) toolpaths equivalent to those produced by an explicit slicer are produced. The input geometry selected is a sufficiently complex model of an automobile engine cylinder head [47]. The geometry of this model is shown in Fig. 10. The implicit slicer was applied with a layer thickness \( \delta = 0.2 \) mm, and \( H_{pr}(x) = x_c + x_y \) with \( x_c \) and \( x_y \) corresponding to the Cartesian coordinates of the slicing planes. The values of \( c_k \) for the perimeters were \{0, 0.3\} and the corresponding values for the infill were \{-50, -49, ..., 50\}. In order to validate the results produced by the implicit slicer, an existing explicit slicer was applied to the same geometry with equivalent settings. In this case the open-source “Slic3r” [48] software package was used. It was selected due to its widespread use in AM by FDM. A compari-
The results of the implicit and explicit slicers are shown in Fig. 11. The particular layer seen in this comparison was chosen as it exhibits the highest topological complexity of the slices. A magnified portion of this comparison is shown in Fig. 12. A comparison of Figs. 11 and 12 reveals that the implicit methodology produces an output that is equivalent to that produced by the explicit slicer. However, some minor differences are noted. First, the explicit slicer resorts to concentric shells to fill narrow gaps. Second, it can be seen that the explicit slicer is joining the infill segments in an attempt to optimize the toolpath. Examination of the implicit output, clearly indicates that similar provisions could be made.

**Test Case 2: Stanford Dragon**

To demonstrate the capability of the implicit slicer to generate support structure a second test case is considered, utilizing the geometry of the well-known Stanford Dragon which is shown in Fig. 13. In this case, it is clear that the generation of support material is required for satisfactory production using FDM. Using the same frame of reference as the previous test problem, the implicit slicer is applied with a layer thickness of $l = 0.5$ mm. The values of $c_k$ for the perimeters were $\{0, 0.2, 0.4\}$ mm and the corresponding values for the infill were $\{-20, -19, \ldots, 20\}$ mm. The “wave” infill of Fig. 8(c) was arbitrarily selected for the main body of the component.

The support material for this model was generated with a single perimeter at $c_k = 0.2$. This creates a small gap between the support material and the main body of the component, and also prevents the generation of support material for shallow-angle overhangs. The infill of the support material was generated using the “eggcrate” infill of Fig. 8(d), at levels of $\{-0.5, 0.0, 0.5\}$ mm. The output produced by the implicit slicer using these settings is shown in Fig. 14.

The results of shown in Fig. 14 indicate that the implicit slicer may be used to produce objects with infill defined by any field function of interest. In the next section we employ this ability in order to incorporate functional fields of design interest into the additive manufacturing process.

**FUNCTIONAL TAILORING AND EXPERIMENTAL VALIDATION**

The examples given in the previous section demonstrate that the implicit slicing methodology is capable of producing highly non-uniform toolpaths. The purpose of this section is twofold; first to demonstrate that the implicit methodology may be used to produce components with significantly improved physical properties and performance for the purpose of functional tailoring; and second to validate this capability by comparing its performance response for various ways of applying the slicer to affect the mesoscale, via experimental data produced from physical experimentation.

A “dogbone” test specimen intended to be subjected to uniaxial tension loads is used for this purpose. The input geometry
of the test specimen is shown in Fig. 15, along with the associated $\gamma$ and $\omega$. The test specimens are 100 mm tall, 30 mm wide, and 5 mm thick. The gage area used for strain measurement is 10 mm wide and 20 mm tall, centered on the test specimen. The “COMSOL Multiphysics” FEA software tool [49] was used to solve the structural mechanics PDE governing the physics of the system under the influence of a 1 kN tension load. In addition a clamping load of 1 kN (total) was applied to each end of the specimens, in an area coinciding with the application of tensile forces, as shown in Fig. 15. The regions of force application are 12 mm tall and cover the full width of the specimen. The clamping load magnitude was estimated according to our prior experience with the system employed for physical testing. Fig. 16 shows the resulting von-Mises stress distribution ($\sigma_{vm}$).

We first compute a purely rectilinear grid infill, as used in many existing slicers, as

$$H_{lin}(x,y,z;\theta) = x \sin(\theta) + y \cos(\theta) \left(-1(z/l)^2\right), \quad (13)$$

where $x$ and $y$ are the Cartesian coordinates within each slice, $z$ is the Cartesian coordinate normal to the slices, $l_i$ is the layer thickness, and $\%$ indicates the arithmetic modulo operator. This function creates a grid of parallel diagonal lines, rotated by $\theta$ from the $x$-axis for each slice’s infill, which flip about the $y$ axis every other layer. The linear infill is subsequently modulated according to the value of $\sigma_{vm}$ in order to produce infill that is denser in high-stress regions. As is common in existing slicers, a value of $\theta = \pi/4$ is used. The infill function is composed from $\sigma_{vm}$ and $H_{lin}$, taking the form

$$H_{in}(x,y,z) = \sigma_{vm}(x,y,z) H_{lin}(x,y,z;\pi/4) k. \quad (14)$$

where $k$ is a scalar parameter that controls the relative infill density. A visualization of the field functions $\sigma_{vm}$, $H_{lin}$, and $H_{in}$ is given in Fig. 16.

Figure 17 shows the final toolpath design produced by the implicit slicer. Also shown is a physical realization of this toolpath, produced on a consumer FDM 3D-printer. In order to provide a comparison, the implicit slicer was also used to produce dogbone test specimens using a different modulation pattern, with $H_{in}(x,y,z) = \sigma_{vm}(x,0,0) H_{lin}(x,y,z;\pi/4) k$. Finally, reference specimens were produced using the unmodulated, uniform infill function $H_{in} = H_{lin}$. The density parameter $k$ was tuned so that all three types specimens were of identical mass.

Three replicates of each specimen photographed in Fig. 17 were additively manufactured from ABS polymer using a desktop FDM 3D printer. Each was printed sequentially, using an
identical set of instructions sent to the additive manufacturing machine so as to minimize differences between replicates. The specimens were gripped and loaded in tension until failure using a small uniaxial test frame. Full-field techniques [50, 51] were used to measure the specimen strains during testing. The experimental apparatus and setup are shown in Fig. 18.

The results of the testing are shown in Fig. 19. It is seen that the properties of the specimens featuring stress-modulated infill are markedly different than those of the specimens with conventional linear infill and the same mass. The ultimate failure stress is increased by 45%, and the apparent elastic modulus increased by more than 100%. These data are given in detail in Table 1. These results clearly demonstrate that the implicit slicer may be used to alter the properties of an additively manufactured object in order to produce enhanced performance with respect to its intended function. It should also be noted that the stress-modulated specimens failed outside the gauge area, typically at or near the grips. This indicates that the ultimate strength of the specimens may be under-represented by the data of Fig. 19. This may indicate that additional performance increases may be realized by augmenting the formulation of Eq. (14) with an additional component to compensate for such failure modes. Finally, the selection of $\theta = \pi/4$ is effectively arbitrary; in the future it will be beneficial to examine in more detail the effects of this parameter as well as potential infill densification in the regions within the grips.

TABLE 1: Results from testing of additively manufactured specimens. All figures represent the average results of three replicate experiments. Parenthetical values indicate relative performance versus the Linear infill case.

<table>
<thead>
<tr>
<th>Infill Type</th>
<th>Ultimate Stress (MPa)</th>
<th>Apparent Elastic Modulus (GPa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>13.7</td>
<td>0.79</td>
</tr>
<tr>
<td>Stress A</td>
<td>19.4</td>
<td>1.24</td>
</tr>
<tr>
<td></td>
<td>(+ 41.6%)</td>
<td>(+ 57.0%)</td>
</tr>
<tr>
<td>Stress B</td>
<td>19.8</td>
<td>1.71</td>
</tr>
<tr>
<td></td>
<td>(+ 44.5%)</td>
<td>(+ 116%)</td>
</tr>
</tbody>
</table>

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States. Approved for public release; distribution is unlimited.
CONCLUSIONS

We have developed and demonstrated a new approach for toolpath generation and slicing for AM. A review of existing slicing methodologies has shown that prior approaches may be categorized as explicit slicers, as they operate by means of explicit transformations applied upon the input geometry. In contrast to this, the present paper introduces an implicit approach in order to address the loss of non-geometric information in the digital thread, and to better convey the intent of designers throughout the additive manufacturing process. The introduced methodology allows for a straightforward implementation that avoids the complexity associated with traditional polygon offsetting. It also allows great flexibility in the specification of the additive manufacturing toolpath, and for the generation of toolpaths based on physical fields of interest for the purpose of controlling the mesoscale structure in manner that enables functional tailoring of the to be manufactured parts.

The results demonstrated in this work show that the implicit slicer offers several important advantages. The first test case, on the engine head geometry, demonstrates that the methodology can be applied to generalized, and even very complex components, such as those that are increasingly produced using additive manufacturing systems in practice. The second test case, on the Stanford Dragon, demonstrates the ability of the implicit slicer to generate support structure. The third and last test problem, with dogbone specimen, demonstrates that the infill toolpath may be generated based on the solution to a partial differential equation capturing the physics relevant to the original design intent such as the stress behavior of the system under tension. The results of an FEA simulation, that captures the explicitly specified intended use of the dogbone specimen, are successfully used to compute the infill toolpath geometry. Quantitative assessment of the implicit slicer via actual mechanical testing of different variations of the mesoscale structure shows that significantly improved mechanical properties and component performance may be achieved by use of this methodology to incorporate the results of an external structural mechanics simulation.

In the test cases presented, the toolpaths produced by the implicit slicer do not represent optimal structures. Rather, the purpose of these exercises is to demonstrate the great flexibility offered by the implicit methodology, and the possibility of including relevant physics or multi-physics results into the production of additively manufactured components. This in turn opens the door to the production of components with performance properties that are tailored towards the intended use of the component.

It must also be noted that the present work is exploratory in nature, and that there is a great deal of future work that is required to bring the implicit methodology to the state of maturity exhibited by its explicit counterparts. Our future plans include:

1. Development of strategies in order to generate production time-optimized toolpaths.
2. Development of a methodology for producing functionally optimized components. This may require the development of constitutive models of additively manufactured objects which reflect the highly nonuniform and hierarchical nature of these processes.
3. Expansion of the implicit methodology to include features such as bridge and gap fill routines, thin section detection, and other features present in current explicit slicers. The development of routines for producing “support” or scaffolding material is particularly important.
4. Exploration of the best methods for automatically determining the contour intervals. For example, it may be necessary to limit the gradient of the infill function in order to avoid over-dense contours, such as those seen in the second test problem. Conversely, it may be desirable to increase the density of the contours in a localized sense in order to satisfy additional performance constraints.
5. Developing a methodology for either discrete or gradual translations between different fields for different portions of the toolpath. For example, to ensure that the outer surface of the component is solid when a sparse infill pattern is used.
6. The development of processes and machine-specific interpreters to translate the output toolpath into NC codes for a variety of additive manufacturing hardware.

The present work illustrates many of the advantages of adopting an implicit slicing methodology. Augmented by the future work outlined above, this methodology may significantly improve the state of the additive manufacturing digital thread. By re-introducing design intent into the toolpath generation process, it is likely that the implicit methodology will allow the additive manufacture of complex components that are better suited to their intended purpose. Additionally, it may be possible to
achieve important improvements in production time, raw material, and energy consumption that will in turn reduce the cost of additively manufactured structures and components.

ACKNOWLEDGMENT
The authors acknowledge support for this work by the Office of Naval Research through the Naval Research Laboratory’s core funding, as well as the National Research Council’s Research Associateship Program. This work was also supported in part by the National Institute of Standards and Technology through Interagency Agreement NIST IAA-2018-0011-0.

REFERENCES


