Solid Freeform Fabrication 2024: Proceedings of the 35th Annual International
Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

An Interpolative Slicing Algorithm for Continuously Graded Stiffness in Viscous Thread Printed Foams

MASA NAKURA*, VIVEK SARKAR*, and DANIEL REVIER, University of Washington, USA
BRETT EMERY and JEFFREY I. LIPTON†, Northeastern University, USA

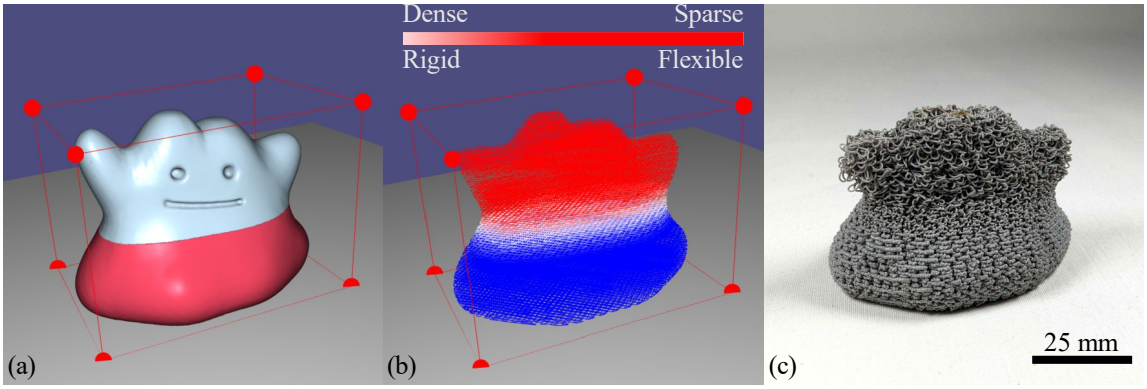

Fig. 1. A three-part illustration of our method for creating multiple stiffness foams using viscous thread instability printing (VTP). (a) A screenshot of a 3D model in the GUI of our application. (b) The corresponding printing tool path for the model, colored by the amount of material to be dispensed. (c) A photograph of the final printed structure, demonstrating the successful realization of a graded elastic structure.

Foams, essential for applications from car seats to thermal insulation, are limited by traditional manufacturing techniques that struggle to produce graded stiffness, a key feature for enhanced functionality. Here, we introduce a novel slicing algorithm for producing heterogeneous foams through viscous thread printing (VTP). Our slicer generates a single, global toolpath for the entire foam volume while modulating the viscous thread's self-interactions along this path to program stiffness. The slicer integrates multiple meshes into a unified print space and interpolates the print speed and height based on specified mesh parameters to program the desired stiffness variations. Using both qualitative samples and quantitative compression tests, we demonstrate that our slicer can (1) generate foam stiffnesses spanning an order of magnitude, (2) achieve millimeter precision in stiffness control, and (3) continuously vary stiffness between regions of constant stiffness using arbitrary functional forms.

Additional Key Words and Phrases: foams, slicers, 3d printing, geometry

*Both authors contributed equally to this research.
†Corresponding author.

Authors' addresses: Masa Nakura, mnakura@cs.washington.edu; Vivek Sarkar, viveksar@uw.edu; Daniel Revier, drevier@uw.edu, University of Washington, Seattle, WA, USA; Brett Emery, emery.b@northeastern.edu; Jeffrey I. Lipton, j.lipton@northeastern.edu, Northeastern University, Boston, MA, USA.

1 INTRODUCTION

Foams are ubiquitous in everyday life with applications in insulation, padding[8], filtration[6] and more[20] and derive their material properties from the interaction of a microstructure and the properties of the base material[1]. In traditional foam manufacturing, the microstructure is produced from the interactions of gases with liquid materials that solidify to form stochastic structures[1]. When done in bulk, this produces a structure whose material properties homogenize relatively quickly and can be varied substantially. However bulk foams can only be composed through lamination which creates the possibility for stress concentrations[20], whereas the ideal would incorporate continuous and gradual transitions between regions.

Here we show a method of creating smooth and continuous 3D printed foams with spatially controlled material properties. We achieved this by developing a custom printing method and slicer that is able to generate unique foam microstructures and smoothly transition between them. This allows us to make controllable gradients of different functions such as linear or logarithmic transitions between regions. The resulting system uses common, desktop 3D printing technology and material sets (e.g., thermoplastic polyurethane or TPU) and can vary from standard lattice infills into coiled microstructures to provide a transition between stochastic and ordered lattice frameworks. We use this new slicer to show how to produce regions of decreased stiffness to make living hinges, custom orthotics, and graded mechanical properties in objects orders of magnitude larger than cell size. The main contributions of this work are:

- A method for creating multiple stiffness foams using viscous thread instability printing (VTP).
- An algorithmic slicing approach that produces multi-stiffness foams from a single tool path.
- An example of direct spatial control of stiffness using VTP for joints, padding and controlling material properties.

2 RELATED WORK

2.1 Explicitly Defined 3D Printed Foams

The production of foams in additive manufacturing requires either direct extrusion of a pre-foamed microstructure [3, 14, 19], or the production of a network of materials and voids. These voids can be explicitly created as geometric features or can be implicitly created by the printing process. Direct foam extrusion has had the widest adoption with applications in construction for the production for casting forms and insulation [3].

The creation of intricate cellular structures through traditional 3D printing requires an explicit and detailed definition of geometry. [11] delves into the use of resin-based printing to fabricate open-cell foams with graded material stiffness. The authors employ a computational method based on a Voronoi cell structure to generate foams with spatially dependent elastic behavior. They construct a network of beams along the Voronoi cells' boundaries, using cell shape and size to influence local stiffness. The system can be used to produce anisotropic stiffnesses in the material [12]. Their method successfully yields significant elastic differences between areas of prescribed stiffness, achieving a varied elastic stiffness on the order of magnitude from 0.5 to 2.5 MPa. While it is feasible to design and print such structures, the method demands a machine resolution significantly finer than the cell size to achieve the desired homogenized effect. This requirement inherently limits the scale of structures that can be produced.

In another approach, [22] introduces a technique for printing graded foams by dispensing liquid shell-gas droplets and photopolymerizing the shell. They create both open- and closed-cell foams by using either oxygen or nitrogen as the gas. They also achieve graded material properties by altering the cell shape and size. However, their technique demands the use of high-precision printing techniques. The machine stage must be fine enough to produce the bubbles, which form on the order of hundreds of microns. Furthermore, the authors note that the printing speed and dispensing

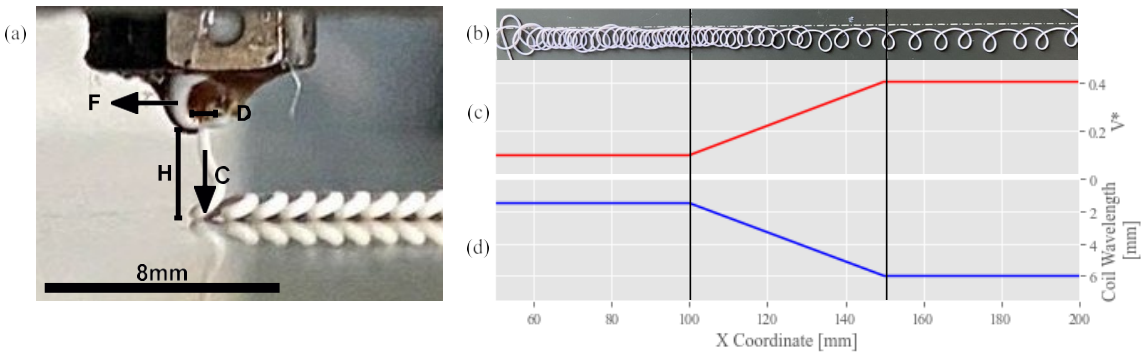

Fig. 2. The VTP printing process. We see that a higher V^* results in less dense coiling. (a) VTP printing in action, with all associated variables labeled. (b) A line of coiling going from a low V^* to a high one. (c) The graph of the changing V^*. (d) The graph of the coil wavelength.

pressure affect the filament width and shape, indicating the need for precise control. They are able to produce varied elastic stiffness of the order of magnitude from 10 to 400 kPa.

All of these methods underscore the need for high-resolution printers to achieve the desired homogenization effect, limiting their application to small-scale structures. The computational complexity of generating the explicit structure further exacerbates this limitation. In contrast, our approach allows for the definition of a coarse toolpath, akin to a traditional 3D printing infill, and leverages the physics of Viscous Thread Printing (VTP) to implicitly generate fine cellular structure. This distinction underscores the novelty and potential advantages of our method, particularly in terms of scalability.

2.2 Viscous Thread Instability and Printing

Viscous thread instability (VTI) is a well-characterized phenomenon that has intrigued scientists for over a century [2, 16–18, 21]. VTI, which occurs when a viscous fluid is extruded at steady-state from a height much larger than the nozzle it is leaving, is governed by the dimensionless parameters V^* (dimensionless speed) and H^* (dimensionless height). These parameters are defined as follows:

$$V^* = \frac{F}{C}$$

$$H^* = \frac{H}{\alpha D}$$

where F is the nozzle's translation speed, C is the speed of the material as it leaves the nozzle, H is the nozzle's print height, α is the material and process dependent die swell constant, and D is the nozzle diameter [23]. These variables are illustrated in Figure 2. The coil behavior changes as V^* and H^* are modified, with the coil behavior having an intertwined effect from both. For example, increasing V^* leads to fewer coils per unit length, while an increasing H^* leads to larger.

While VTI in one dimension is a well-characterized phenomenon[4, 13], the study of what happens when multiple threads are layered is still an active area of research. The pioneering work by [10] demonstrated the use of VTI to produce foams with tunable stiffness. Viscous thread instability printing (VTP) is a foam 3D printing process, which relies on the instability of a viscous thread hitting a moving surface, buckling and weaving to create a self-intersecting

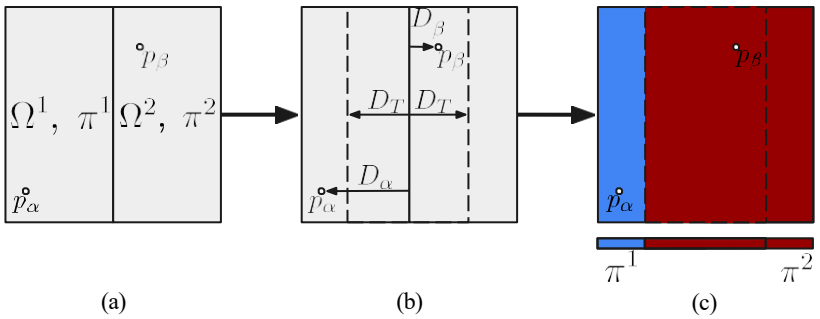

Fig. 3. 2D overview of point interpolation. (a) Ω^j and π^j are identified for query points p_α and p_β. (b) Relevant intersection region is determined by comparing D_i to D_T. p_β is contained in the intersection region, while p_α is not. (c) Finally, p_β is assigned π_i using the weighting function from Appendix A, while p_β is assigned π^1.

pattern. This self-interaction produces cells that are on the same size scale as the print process. Because it relies on viscous extrusion, it is compatible with a wide range of materials including glass[4], corn dough[9], silicone[10], TPUs and nylon[5]. The authors focused on modifying V^* and H^* to produce alternating loops and translating coils, which produce the greatest amount of interconnectedness between strands.

More recent efforts have been made to characterize the stiffness of VTP foams as a function of density, resulting in some understanding of homogeneous properties[5]. In this case, the authors focused on modifying V^* and H^* to produce different sizes of alternating and translating coils. The authors successfully manufactured homogenized parts that result in a cellular structures that follow the well-established scaling laws of cellular materials. Until now there has not been a method of producing graded VTP foams. Expanding on this previous work, our method bridges this gap and allows for continuous, single-print VTP parts that can obtain gradients throughout the structure.

3 VISCOUS THREAD PRINTED FOAM GENERATION

In order to generate spatially varying foams, we assign a V^*/H^* pair, $\pi = \{V^*, H^*\}$, to each point inside the 3D model space. To do this, the user must divide the domain into a set of non-overlapping sub-domains ($\Omega^j \in$ M) that share part of their boundary with another domain. Each mesh has a prescribed π^j, and we interpolate the region surrounding their shared boundaries to find the specific π_i values at a given point x_i. Note, we use a superscript/subscript distinction to denote the difference between mesh-wise and point-wise quantities, where superscript corresponds to meshes and subscript corresponds to points. We then use a traditional slicer library (libslic3r) [15] to generate the single toolpath and feed it into our own algorithm to associate all points in the toolpath with a V*/H* pair. We can then modify the toolpath provided by libslic3r to create one that will produce the desired VTP foam based on user parameters.

3.1 Interpolating V^*/H^*

Viscous Thread Printing (VTP) operates as a constant extrusion process, requiring a single, uninterrupted tool path to fill the entirety of $\bar{\Omega}$. To do this we implement a few design constraints: 1) meshes must be non-overlapping but share a boundary with at least one other mesh and 2) meshes can share a boundary with at most one other mesh. An implication of these two constraints is that the union of all meshes, denoted as $\bar{\Omega} = \bigcup_M \Omega^j$, must form a single, continuous domain. We then slice $\bar{\Omega}$ in a traditional manner and return a single tool path across the unioned domain. Finally, we refine the tool path into smaller segments and query points along the tool path using our interpolation

scheme to determine the π_i at each point. The end result is a 3D-printable file, called GCode, with extrusion values that match the continuous space of graded V^*/H^* values.

3.1.1 Mesh Union and Intersections. The Ω^j are defined to be non-overlapping with a shared boundary. This is done to have a clearly defined surface inside of $\bar{\Omega}$ that will serve as the mid-point of our V^*/H^* interpolation scheme. In theory, touching and non-overlapping domains will have a continuous and void free union; however, the triangular meshes common in 3D printing introduce floating point error which may be enough to introduce void artifacts on the interior of a unioned mesh. Thus, we expand all of the submesh faces along their normal by a small amount ϵ (on the order of 10^{-6}) to ensure some overlap, allowing us to robustly determine $\bar{\Omega}$ through an algorithm outlined below in algorithm 1. The mesh intersections are computed in a similar manner via first augmentation and then a mesh boolean intersection operation. The augmentation by small ϵ ensures that meshes will overlap only with their adjacent counterparts and not meshes an appreciable distance away. Once again, because ϵ is small compared to mesh size the intersection meshes approximate shared boundaries between meshes well.

3.1.2 Interpolation Scheme. To determine the π for every point along the tool path in $\bar{\Omega}$ we first look at the general case of interpolation. We first associate a π^j with each Ω^j, which define regions of constant V^*/H^*. We then query a point $p_i \in \bar{\Omega}$ to determine π_i. This is done by first calculating the signed squared distance functions first to each Ω^j to determine which mesh p_i belongs to. Because the meshes are assumed to be non-overlapping but have an interfacing boundary we can guarantee p_i will be uniquely mapped to a single pi^j up to the submesh boundary.

We then calculate a signed distance D_i between p_i and each intersection mesh. We cannot use the same squared distance sign as before, however, because there may be cases where a point is closer to an unshared boundary than a shared one, which would lead to an inaccurate distance calculation from the intersection. Each intersection is associated with two submeshes Ω^j and Ω^k, a transition length D_T, and a weighting function $w(x; s)$ that defines how π varies within the transition region. A transition region has orientation and is denoted by $\Omega^{j\to k}$ where the orientation of $j \to k$ informs the direction from Ω^j to Ω^k. Thus, a point is said to be in a transition region if $\frac{\|D_i\|}{D_T} < \frac{1}{2}$, where the sign of D_i determines if p_i is in Ω^j or Ω^k. Once again, p_i can be uniquely determined to belong to a single transition region due to the assumption that mesh boundaries may only be shared by one other mesh.

Because the transition region is associated with an orientation between two meshes we can define a mapping $w^{j\to k} : [-\frac{1}{2}, \frac{1}{2}] \to [0, 1]$ as a weighting function between Ω^j to Ω^k across a length D_T. Specifically, for the points inside Ω^j D_i is negative, and for the points in Ω^k D_i is positive. The user may freely choose the direction of interpolation function across the mesh boundary. More formally,

$$w(x) = \begin{cases} 0 & x \leq -\frac{1}{2} \\ w(x;s) & -\frac{1}{2} < x < \frac{1}{2} \\ 1 & \frac{1}{2} < x \end{cases}$$

where $x = \frac{D_i}{D_T}$ and s is a parameter specific to the weighting function. We can then determine the VTP parameters by

$$\pi_i = \pi^i \cdot (1 - w(x;s)) + \pi^j \cdot w(x;s) \tag{1}$$

The weighting functions used in this work are provided in Appendix A.

3.2 Implementation

We built a custom VTP slicer program utilizing libslic3r [15] for initial toolpath generation and libigl [7] for the GUI and some geometry operations, with all other algorithmic implementation details being our own. The VTP slicer is able to import meshes of multiple formats, assign a V^* and H^* to each mesh, and define the transitions between these different meshes. The users are also given an option to view intersection and can select a specific one, which will be highlighted on the GUI. The transition can be defined in terms of what function it takes on, the direction of that transition, and the length of the transition. We also are able to define the other miscellania involved in 3D printing such as layer height, line spacing, etc.

As it stands, our slicer is capable of processing parts intended to have multiple densities, as long as the following constraints hold:

- As discussed in the previous section, the meshes must interface to form a continuous domain, with a mesh sharing a boundary with at most one other mesh.
- $\bar{\Omega}$ must only produce one polygon when sliced with a z-oriented plane. The reason for this is that introducing multiple polygons on a single layer would necessitate a discontinuous toolpath or undesired strands outside the geometry definition.

3.2.1 Mesh Augmentation. As mentioned above, touching and non-overlapping domains should have a continuous and void free union; however, floating point errors may be enough to introduce undesired artifacts in the unioned mesh. To augment the mesh and ensure well-formed boolean operations are possible we expand the mesh by a small distance ϵ. Since this ϵ is small (approximately 2-3 orders of magnitude smaller than the mesh itself), we determine our computed $\bar{\Omega}$ approximates the entire domain well without distorting the original design intent of the shape.

Algorithm 1: Mesh Boolean Operation

Input : Two meshes M_1 and M_2, expansion distance ϵ, boolean operation B
Output : Single mesh M_3 representing union or intersection

$F_{1,2} \leftarrow FaceNormals(M_{1,2})$
$V_{1,2} \leftarrow Vertices(M_{1,2})$
for $i \leftarrow 1$ **to** V **do**
  $sa \leftarrow AverageSurfaceNormal(V[i], F)$
  $V[i] \leftarrow ShiftAlongVector(sa, \epsilon)$
end for
return $B(V_1, F_1, V_2, F_2)$

3.2.2 Toolpath Refinement. After we compute the union of all the submeshes, we feed it into libslic3r in order to get a toolpath that covers the whole mesh. A toolpath in this case is nothing more than a list of segments, which indicate what path the nozzle of the FDM printer is to follow. We can only vary printer behaviour across different segments, so in order for changes in V^* or H^* to be perceivable, we need these segments to be short. The toolpath segments we receive from libslic3r are rather coarse however, with each segment spanning the entire part, and its endpoints lying on the boundary of $\bar{\Omega}$. We therefore run a refinement operation on it so that the toolpath has the necessary resolution. We do this by discretizing each toolpath segment into multiple segments of a length specified by the user. This gives us a list of small toolpath segments with which we can associate a $V^*$and H^*. The query point we use to determine the V* and H* of each segment is its end point, which we feed into our interpolation function.

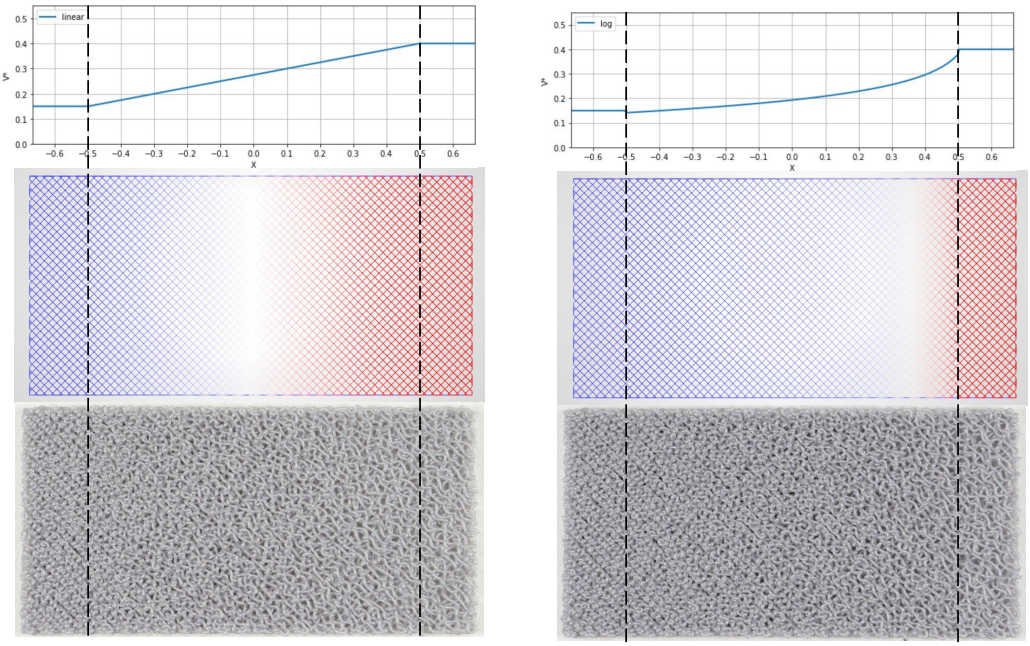

Fig. 4. Two examples of different transition functions affecting V^*. On the left we have a linear transition while on the right we have a logarithmic transition, both over 75 mm.

3.2.3 Toolpath Interpolation. To implement the interpolation function (Equation 1), we first find the associated mesh of a point p_i using the minimum signed distance from p_i to each Ω^j, denoted as $D^j = \min_i \text{SDF}(p_i, \Omega^j)$, where $D^j < 0$ implies $p_i \in \Omega^j$. Similarly, we want to find all D_i, the signed distance from point p_i to an intersection. However, using a signed distance function will always result in a non-negative distance since the intersections are slim meshes with virtually no volume, and any points inside the intersection mesh can be assumed to have a distance of zero to its boundary. For weighting functions with derivative functions symmetric around the y axis, the workaround is quite straightforward because the direction of the intersection does not matter when applying the interpolation. In this case, we can fix the direction of the interpolation function by always calculating w_i for $p_i \in \Omega^j$ using a positive D_i. However, for other types of functions such as the logarithm, the orientations of the meshes matter when applying the interpolation. Therefore, the algorithm must keep track of the directions for each intersections and manually adjust the signs of D_i based on the direction and the mesh associated with p_i. With D_i and the direction identified, V^*/H^* can easily be obtained using Equation 1 and the weighting functions defined in Appendix A. When all the points on the toolpath are characterized, the slicer is ready to translate the V* and H* values into GCode variables.

3.2.4 GCode Generation. Once we have determined the V^*/H^* of each point by inputting it into our algorithm, we can finalize the position of the toolpath in space and generate the remaining variables needed to fully characterize a the print. These parameters ultimately are translated into G-code commands that the FDM printer can execute, which we now describe.

A G-code command G1 instructs the printer to move from its current position (x_0, y_0, z_0) to a new position (x, y, z) at a certain speed f, while extruding Δe length of material. In a G-code file this looks like G1 Xx Yy Zz EΔe Ff, where G1

Algorithm 2: FindVH

Input : Query Point p, List I of intersections, Set Ω of submeshes , List Π of all V*/H* pair π

Output : π_p

// Find Inside Mesh, containing p
for $i \leftarrow 1$ **to** Ω **do**
    $ssd \leftarrow SignedSquaredDistance(p, \Omega_i)$
    if $ssd < 0$ **then**
        $iM \leftarrow i$ // Inside Mesh
    end if
end for
$D \leftarrow UnsignedDistances(p, I)$ // Distance to each intersections
$\pi_p \leftarrow \Pi[iM]$
for $i \leftarrow 1$ **to** I **do**
    $d \leftarrow D[i]/TransitionLength(I[i])$ // Scaled Distance to Intersection
    if $d < \frac{1}{2}$ **then**
        $oM \leftarrow getOutsideMesh(I[i], iM)$ // The non-inside mesh bordering interception
        $startFromInside \leftarrow (iM == I[i]_{start\ Mesh})$
        $\pi_p \leftarrow InterpolationFunction(d, startFromInside, \Pi[iM], \Pi[oM])$
    end if
end for
return π_p

and the capital letters are character literals, and the lowercase letters represent numerical values for the printer to parse. The x and y positions are determined by the object's infill toolpath, whereas z, e and, f are values to be solved for.

The first value we need is the distance by which we need to raise each point in order to achieve the desired coiling behaviour. We can calculate this value by using the associated H* of each point, using the following formula, where dz is the layer height and D_N is the nozzle diameter:

$$z_{offset} = \alpha D_N H^* - dz$$

Since libslic3r gives us the toolpath directly on top of each layer with no offset, we simply add z_{offset} to the current z-value of each point to get the final position of the nozzle.

The remaining values to generate for each G-code command are Δe, the amount of filament extruded in millimeters over each segment, and f , the speed at which the printer head travels over the segment in millimeters per minute. Using the V* associated with the point, we can derive these values using the following formulas:

$$\Delta e = \frac{L \ A_T}{V^* A_F}$$

$$f = V^* C \frac{A_F}{A_T}$$

where A_T is the cross-sectional area of thread being extruded accounting for die swell, A_F is the cross-sectional area of filament being fed into the extruder, C is the speed of material leaving the nozzle, and L is the distance between p_i and p_{i-1}. For p_0, the previous point selected is printer specific, generally being close to the origin.

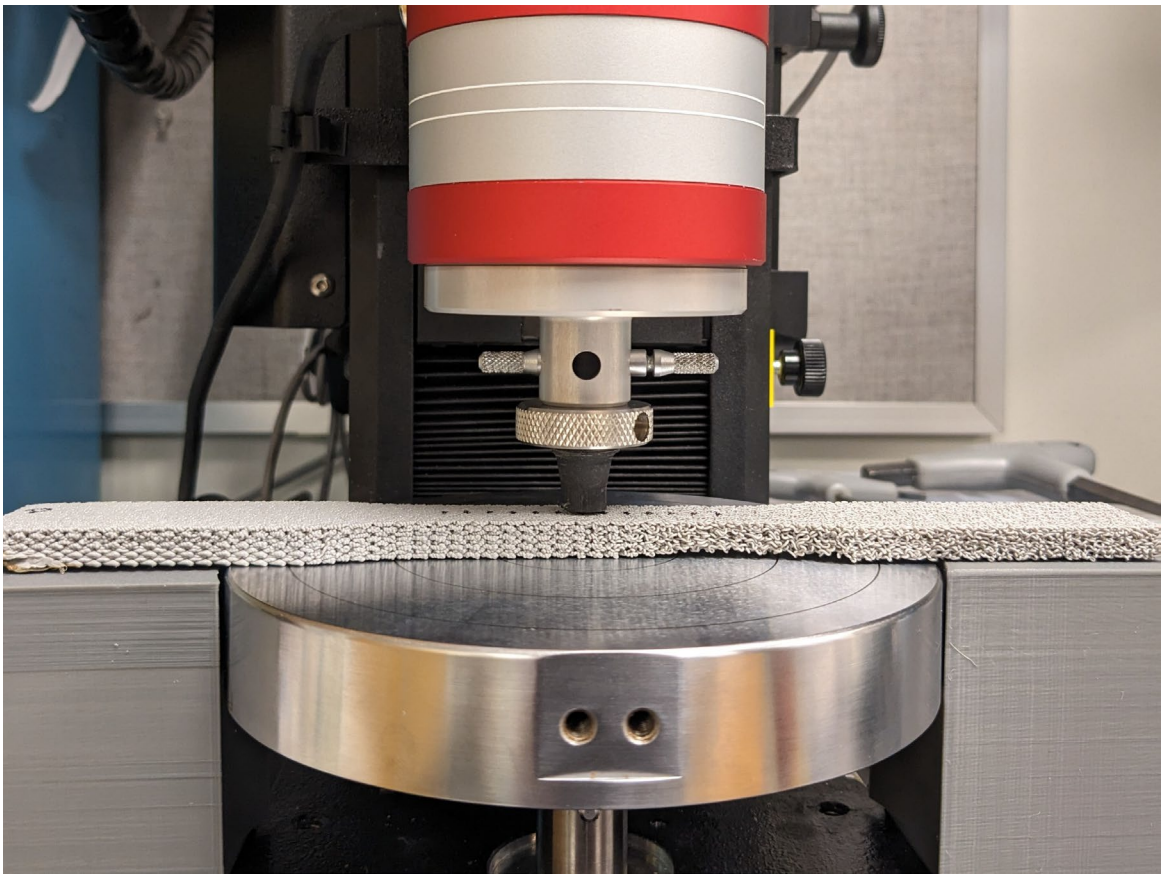

Fig. 5. A probing the compressive modulus featuring a logarithmic transition between V^*s.

Once we have these values, we then have all of the information that characterizes the behaviour of the printer, and can write the G-code that will produce the specified part.

4 GRADED VTP FOAMS

4.1 VTP Foam Characterization and Control

The data gathered from the printed structures provide a quantitative demonstration of our method's ability to control the modulus through graded VTP parameters. We performed compression testing of V^* transitions along a long, dogbone-like structure every 5 mm, shown in Figure 5. A linear V^* gradient (dashed blue curve) with a transition length of 40 mm was used across the 60 mm neck length as shown in the bottom plot of Figure 6. This represents programming of V^* as a function of position along the axial length of the dogbone. V^* values at the ends were 0.15 on the left and 0.4 on the right. Notably, a linear V^* gradient resulted in an exponential decrease of compressive modulus left to right (solid blue curve and measurements top of Figure 6). We fit this data an exponential curve shown as a solid blue line ($E(x) = 4.79e^{-3.11x}$, R^2 : 0.987).

To further validate our method, we adjusted our approach in response to these findings. We took our findings from the linear V^* modulus measurements and implemented a logarithmic sweep of V^* (dashed red curve bottom of Figure 6) using the fitted parameters from the initial linear V^* gradient. Due to measurement error in the exponential fit, the V^* values are discontinuous as the ends of the transition region and are 0.1415 on the left and 0.3795 on the right. Despite this error, the logarithmic V^* successfully achieved a linear decrease in modulus across the length ($E(x) = -24.15x + 11.47$, R^2 : 0.977) (solid red curve and measurements, top Figure 6), as opposed to the initial exponential decrease from the linear V^*.

This result, as illustrated in Figure 6, confirms that the modulus of the printed material can be manipulated by adjustment of the VTP parameters. The data show that our method provides a robust and flexible approach to controlling the mechanical properties of 3D printed materials.

4.2 Applications

We tested our approach in a variety of applications that use multi-modulus and graded modulus structures, as demon- strated in Figure 7.

First, we generated a cube with low-modulus living hinges (V^*=0.15 or 0.4, transition length zero), shown in its unfolded state, and in its folded state, which exemplifies the potential for creating complex, foldable structures. Low-modulus hinges allow for easy folding and unfolding, which could be beneficial in applications such as soft robotics, packaging or deployable structures. The ability to print structures with varying stiffness within a single print can significantly enhance the functionality of these systems. For instance, areas of the structure that need to be rigid can be printed with a high-modulus material, while areas that need to flex or fold can be printed with a low-modulus material. This could allow for the creation of deployable structures with complex folding patterns or robots that have shock absorbing features built into the frame.

The foot orthotic (V^*=0.15-0.4, linear transitions, 10 mm), designed with low modulus in areas of high stresss such as the ball and heel of the foot, exemplifies the potential to create customized and comfort-enhancing products using our method. The ability to vary the modulus within a single print allows the creation of personalized cushioning objects that can provide targeted support. This application has significant implications for the field of orthotics, where custom-fit devices can greatly improve user comfort and functionality. Beyond orthotics, this method could also be applied to create custom cushioning for wheelchair seats and sockets for prostheses, improving comfort and reducing the risk of pressure sores or skin lesions.

The linear and logarithmic transitions between areas of different moduli on the specimens (V^* from 0.15 to 0.4, linear/log transitions, 40 mm transition length) demonstrate our method's ability to create graded structures with continuous material property transitions. This capability is crucial in applications requiring gradual property transitions, such as load-bearing structures or components subjected to varying stress conditions. For instance, aerospace and automotive industries could leverage this for components that need to withstand different pressure, temperature, or impact conditions. Similarly, in biomedical engineering, graded materials could be useful for designing implants or prosthetics with varying tissue-mimicking properties.

5 CONCLUSION

In this work, we have introduced a novel slicing algorithm that recognizes the boundaries between multiple meshes and facilitates gradient transitions between them. Specifically tailored for the viscous thread printing (VTP) process, this slicer is able to adjust the print height and speed to control the coiling behavior and thereby the foam microstructure,

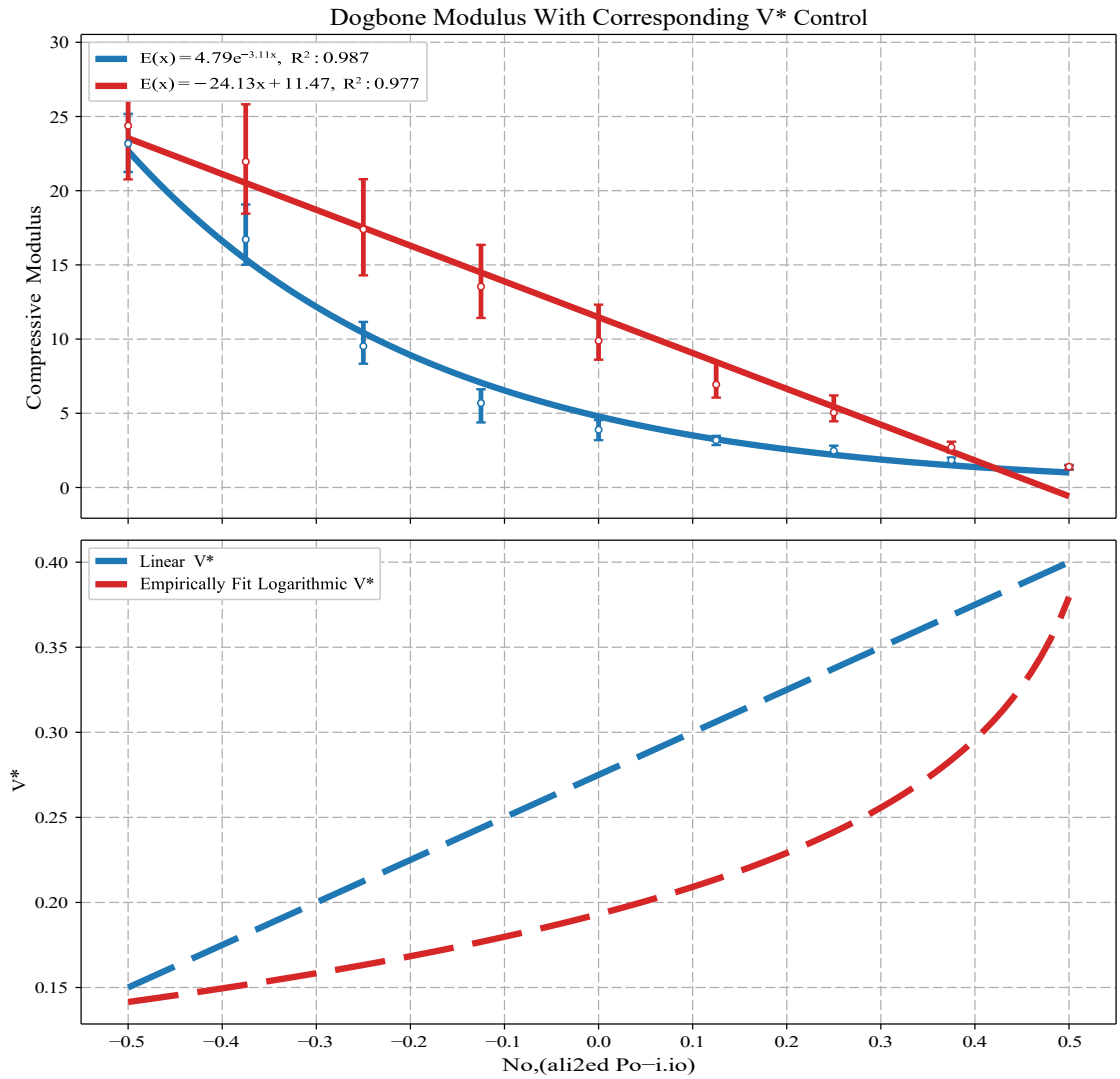

Fig. 6. Compressive modulus measured along the specimens. The specimen with a linear V^* (blue dashed curved) was printed first and the modulus was characterized as a function of space (blue solid curve with measured values). The log-linear fit was applied to this data to arrive at the values needed to invert the behavior. A logarithmic change in V^* was applied (red dashed curve) and the corresponding modulus was fit to a linear curve (red solid curve with measured values). This shows that the exponential relationship between V^* and modulus can be accounted for.

which enables the creation of foams with spatially varied mechanical properties. It allows for a seamless transition from standard printing to stochastic foam structures, integrating foam and non-foam areas into a single print. We demonstrated our spatial control of stiffness by creating qualitative examples, but also quantitatively demonstrating the arbitrary control of stiffness as a function of space. By enabling the creation of multiple stiffness foams using VTP and

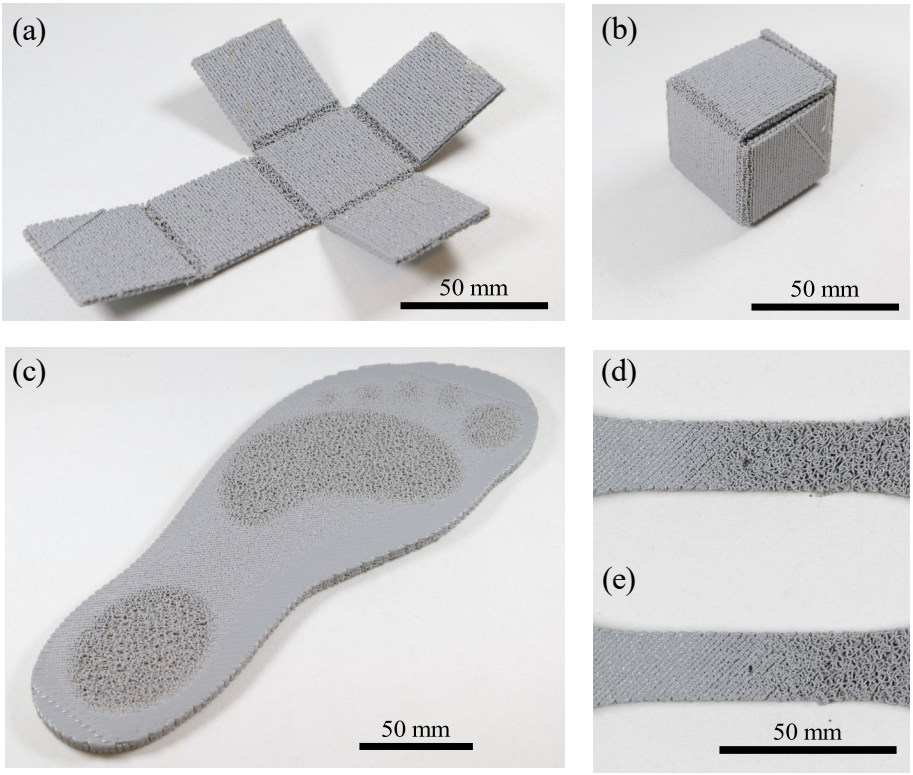

Fig. 7. Examples of multi-modulus printing. (a) A cube with low-modulus living hinges in the unfolded state and (b) in the folded state. (c) A foot orthotic with low-moduli in high stress areas (e.g., ball and heel of foot), V^* range 0.15-0.4. (d) Top down shots of linear (d) and logarithmic (e) transitions between areas of different moduli on the specimen, V^* range 0.15-0.4.

facilitating direct spatial control of stiffness, we have opened up new possibilities for the design and manufacture of advanced materials and components.

A WEIGHTING FUNCTIONS

The weighting functions presented in this work are listed here as $w(x\,;s)$ where x is the distance from the transition intersection normalized by the transition length D_T and s is a transition function specific parameter set by the user. The meaning of parameter s is unique to the specific weighting function and is called out in each section, or if a function does not use s it is omitted. All weighting functions guarantee C^0 continuity with the π^j associated with the ends of the transition and other constraining equations are listed if used. The one exception is the empirically derived logarithmic function which was fit to data and not analytically derived.

Linear.

$$w(x) = x + \frac{1}{2} \tag{2}$$

Cubic.

$$w(x;s) = 4 \cdot (1-s)x^3 \quad + sx + \frac{1}{2} \tag{3}$$

where the control parameter s sets the slope of the function at $x = 0$

$$w'(x; s) = s$$

A special case of the cubic function is $s = 1.5$ which also guarantees C^1 continuity at the ends of the transition region.

Experimentally Fit Logarithmic. The logarithmic function was derived by inverting the log-linear fit of the measured modulus data $E(x)$. Because of error in the measurement, the V^* do not align perfectly at the boundary conditions $x \in \left[-\frac{1}{2}, \frac{1}{2}\right]$.

$$w(x) = \frac{1}{b} \cdot \ln\left(\frac{E(x)}{a}\right) + \frac{1}{2} \tag{4}$$

where

$$E(x) = 23.91x + 13.26$$
$$a = 4.79$$
$$b = -3.11$$

Cube Root.

$$w(x) = \sqrt[3]{x} + \frac{1}{4} \tag{5}$$

Sinusoidal.

$$w(x) = \frac{1}{2}\sin(\pi x) + \frac{1}{2} \tag{6}$$

REFERENCES

[1] Michael F Ashby. 2006. The properties of foams and lattices. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 364, 1838 (2006), 15–30.

[2] George Barnes and Richard Woodcock. 1958. Liquid rope-coil effect. *Am. J. Phys.* 26, 4 (April 1958), 205–209.

[3] Patrick Bedarf, Alessandro Dutto, Michele Zanini, and Benjamin Dillenburger. 2021. Foam 3D printing for construction: A review of applications, materials, and processes. *Automation in Construction* 130 (2021), 103861.

[4] P-T Brun, Basile Audoly, Neil M Ribe, Tom S Eaves, and John R Lister. 2015. Liquid ropes: a geometrical model for thin viscous jet instabilities. *Physical review letters* 114, 17 (2015), 174501.

[5] Brett Emery and Daniel Revier. 2022. Applied viscous thread instability for manufacturing 3D printed foams. In *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication*. 1–2.

[6] LJ Gauckler, MM Waeber, C Conti, and M Jacob-Duliere. 1985. Ceramic foam for molten metal filtration. *Jom* 37 (1985), 47–50.

[7] Alec Jacobson, Daniele Panozzo, et al. 2023. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

[8] Shruthika Kandukuri, Atharva Kashyap, and Jeffrey Lipton. 2022. Vibration Reduction Using Material Jetted Parts for Sander Grips. In *2022 International Solid Freeform Fabrication Symposium*.

[9] Jeffrey I Lipton, Meredith Cutler, Franz Nigl, Dan Cohen, and Hod Lipson. 2015. Additive manufacturing for the food industry. *Trends in food science & technology* 43, 1 (2015), 114–123.

[10] Jeffrey I Lipton and Hod Lipson. 2016. 3D printing variable stiffness foams using viscous thread instability. *Scientific reports* 6, 1 (2016), 29996.

[11] Jonàs Martínez, Jérémie Dumas, and Sylvain Lefebvre. 2016. Procedural voronoi foams for additive manufacturing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

[12] Jonàs Martínez, Haichuan Song, Jérémie Dumas, and Sylvain Lefebvre. 2017. Orthotropic k-nearest foams for additive manufacturing. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

[13] Stephen W Morris, Jonathan HP Dawes, Neil M Ribe, and John R Lister. 2008. Meandering instability of a viscous thread. *Physical Review E* 77, 6 (2008), 066218.

[14] Joseph T Muth, Patrick G Dixon, Logan Woish, Lorna J Gibson, and Jennifer A Lewis. 2017. Architected cellular ceramics with tailored stiffness via direct foam writing. *Proceedings of the National Academy of Sciences* 114, 8 (2017), 1832–1837.

[15] Alessandro Ranellucci et al. 2023. Slic3r: Open source 3D printing toolbox. https://github.com/slic3r/Slic3r.

[16] Lord Rayleigh. 1892. XVI. On the instability of a cylinder of viscous liquid under capillary force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 34, 207 (Aug. 1892), 145–154.

[17] N M Ribe. 2004. Coiling of viscous jets. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 460, 2051 (Nov. 2004), 3223–3239.

[18] N M Ribe, M Habibi, and Daniel Bonn. 2006. Stability of liquid rope coiling. *Phys. Fluids* 18, 8 (Aug. 2006), 084102.

[19] Benito Roman-Manso, Joseph Muth, Lorna J Gibson, Wolfgang Ruettinger, and Jennifer A Lewis. 2021. Hierarchically porous ceramics via direct writing of binary colloidal gel foams. *ACS Applied Materials & Interfaces* 13, 7 (2021), 8976–8984.

[20] Paul Stevenson. 2012. *Foam engineering: fundamentals and applications*. John Wiley & Sons.

[21] S Tomotika and Geoffrey Ingram Taylor. 1935. On the instability of a cylindrical thread of a viscous liquid surrounded by another viscous fluid. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences* 150, 870 (June 1935), 322–337.

[22] Claas Willem Visser, Dahlia N Amato, Jochen Mueller, and Jennifer A Lewis. 2019. Architected polymer foams via direct bubble writing. *Advanced materials* 31, 46 (2019), 1904668.

[23] Hyunwoo Yuk and Xuanhe Zhao. 2018. A new 3D printing strategy by harnessing deformation, instability, and fracture of viscoelastic inks. *Advanced Materials* 30, 6 (2018), 1704028.