JOM, Vol. 72, No. 1, 2020
https://doi.org/10.1007/s11837-019-03825-w

An Algorithm to Generate Synthetic 3D Microstructures from 2D Exemplars

TRISTAN N. ASHTON,[1] DONNA POST GUILLEN ,[1,5] and WILLIAM H. HARRIS[2]

1.—Idaho National Laboratory, 2351 N. Boulevard Ave., Idaho Falls, ID 83415, USA. 2.—Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139, USA. 5.—e-mail: Donna.Guillen@inl.gov

The inverse problem of constructing 3D microstructures from 2D data is an area of active research within the materials science community. This paper presents the implementation of a robust, computationally efficient algorithm: the Hierarchical Algorithm for the Reconstruction of Exemplars (HARE), written in Python to reconstruct 3D features in a given microstructure from up to three orthogonal 2D exemplars and using nearest-neighbor matching to reproduce feature qualities, such as shape, size, and distribution. HARE's feature sampling implements histogram reweighting to avoid both over- and undersampling. A neighborhood voting scheme allows each pixel to provisionally affect its neighbors according to its weight. The algorithm is presently configured for two-phase materials and is being extended to accommodate multiple phases. HARE is a convenient and robust base from which to generate statistically representative synthetic microstructures for use in multiscale modeling or machine-learning applications to support advanced manufacturing and materials discovery.

List of Symbols

$\vec{D}_I$	Diameter of Ith particle in microstructure
$\vec{r}$	Vector denoting the position of each pixel
$\vec{R}_I$	Spatial position of Ith particle in microstructure
$\vec{u}$	Neighborhood vector denoting pixel positions
$\widetilde{N}_{\vec{r}}$	Neighborhood in the exemplar
$N_{\vec{s}}$	Neighborhood in the reconstruction
$E(\vec{s})$	Error function
$g(\vec{r})$	Phase ID value of the pixel at $\vec{r}$ in the exemplar
$H(\widetilde{N}_{\vec{s}_j}(\vec{s}))$	Position histogram of phase ID values in neighborhoods in the exemplar
K	Exemplar side dimension (number of pixels)
L	Reconstruction side dimension (number of pixels)
$m(\vec{s})$	Phase ID value of the pixel at $\vec{s}$ in the reconstruction
t	Computational time
$\omega'_{\vec{s},\vec{s}_j}$	Weight of a pixel's vote when updating phase ID values
p	Number of phases
θ	Probability that each pixel of the exemplar is sampled in the reconstruction

INTRODUCTION

Recent years have seen an accelerating expansion of computational resources available to the materials science research community, due in large part to advances in parallel computing architectures and the resulting increases in processing speeds. These advances have prompted renewed focus in the area of predictive computational modeling of materials, in which novel materials are designed digitally to produce a prescribed functionality and performance. This approach reduces both the time and resources necessary for material design, and opens the frontier for multiscale modeling to connect emergent properties with nano-, micro-, and macroscale features. Key materials processes occur at microstructural scales and are inherently three-dimensional in nature.[1] Richard Feynman, in his celebrated 1959

(Published online October 15, 2019)

lecture, "There's Plenty of Room at the Bottom," imagined the possibilities that could be achieved by manipulating and controlling matter at such small scales.[2] However, the development of such predictive frameworks is challenging for at least two reasons. First, as a classic inverse problem, the space of possible material microstructures to explore to achieve the desired properties is extremely large, which makes the location of a global optimum difficult without specialized searching algorithms. Secondly, predictive models must be verified through comparison with experimental measurements of fabricated materials. This requires that as-fabricated materials be digitally reconstructed in some manner so that the candidate predictive model can be performed on them. The method described here is much less resource-intensive than obtaining 3D material microstructures by means of electron backscatter diffraction with successive serial sectioning.[3] The first challenge has been addressed with a wide variety of techniques,[4] such as modified Markov Chain Monte Carlo sampling and similar Monte Carlo methods,[5–7] and statistical correlation functions,[6,8–11] largely in the context of molecular modeling and computational chemistry, and is not explored here. This paper focuses on solutions to the second challenge, namely, digital reconstruction of three-dimensional microstructures.

As described by Turner et al.[12], the primary aim of predictive materials modeling is the development of processing–microstructure–property–performance relationships to address the inverse problem. For example, consider a binary composite in which a matrix phase contains spherical inclusions of a precipitate phase B of varying diameter. One could completely specify this microstructure with two ordered configuration-space vectors of the form $\vec{R}_I$ where $I = 1, 2, 3, \ldots, n$ specifying the location of the nth inclusion particle, and the vector $\vec{D}_I$ stores their diameters. Finally, boundary conditions (i.e., periodic in all directions) must be specified, assuming that the configuration–space vectors above are meant to constitute a representative sample of the material. Note that this information could equivalently, though more compactly, be stored within a single higher-dimensional vector for actual computational applications, but that might obscure the present discussion. With this information, the microstructure is known deterministically.

Such an exact specification of a microstructure is of limited utility in achieving microstructure–property relationships because such results are not generalizable. Suppose the aforementioned composite is measured to have an unexpectedly high electrical conductivity. Is that due to the absolute positions of the inclusions or their relative spacing? The distribution of diameters? The length of the longest continuous path through the matrix? These questions cannot be answered readily by studying a single microstructure. Turner et al.[12] and others have instead supported the probabilistic or statistical approach to microstructure characterization, in which the bulk properties of a material are assumed to be uniquely and completely specified by the configurational statistics of the microstructure.[12,13] This has led to numerous efforts to build both 2D and 3D solid textures from 2D inputs,[14–17] some of which have emerged as leaders in the rapidly-growing materials development field.[15,16]

This statistical approach is preferable for a number of reasons. First, if two materials are known to be statistically equivalent relative to one descriptor (inclusion radius distribution, for example), yet are still measured to have different properties, one can readily hypothesize alternative statistical descriptors that may give rise to the discrepancy (interparticle spacing distribution, for example). Moreover, one would not be burdened with keeping track of the absolute position of every inclusion particle during this comparison, as required in a deterministic approach, and statistical descriptions are readily amenable to sampling of smaller subsets while simultaneously keeping track of the uncertainties that might arise from such sampling.

The development of a robust reconstruction algorithm that can use three or less 2D inputs to rigorously produce a fully quantifiable microstructure is desirable, but faces two issues common to inverse problems: that exact analytical solutions either may not exist or may exist in local optima that fail to match the desired criteria, or that multiple solutions are possible without indication as to the superiority of one solution over the others. Although general circumventions of these issues often do not exist, it is possible to mitigate them in varying degrees by a selection of dynamic, ideally self-correcting, adjustments to the solution space. Such adjustments can include n-point spatial correlations,[18] which are useful in determining the importance of specific spatial structures to the effective microstructure properties; minimization of objective error functions, often employed in Monte-Carlo methods[5]; reweighting schemes that uniformly sample features within the reconstruction to fairly represent all points within the microstructure[15]; and randomization conditions within the algorithm to nudge the convergence out of a potential local minimum.[19]

METHODS

A robust, computationally efficient algorithm written in Python, the Hierarchical Algorithm for the Reconstruction of Exemplars (HARE), was created to reconstruct volumetric microstructures using a single 2D exemplar for isotropic materials or two to three orthogonal 2D exemplars for isotropic or anisotropic materials. This work builds upon the hybrid MATLAB/C++-based algorithm of Turner and Kalidindi[12] with a code written entirely in

Python to provide extensibility and harness the power of modern open-source scientific computing libraries, toolkits and machine-learning algorithms, as well as automated DREAM.3D pipelines.

The study of the mathematical and computational principles necessary for synthetic microstructure generation is an active area of research.[20] A number of approaches have been developed, largely in the context of research focusing on graphics and image-processing algorithms. In particular, Kopf et al.[15] developed an algorithm for 3D texture synthesis based on the input of exemplars. Although this method was developed primarily for applications in computer graphics, it has been readily assimilated into the materials research community to address the longstanding issue of synthetic microstructure generation, to which it is particularly well suited; when a material is characterized, this is often done via optical or scanning electron microscopy of a polished flat surface of the material. The resulting dataset is a 2D matrix of color values, to which the method of Kopf et al.[15] is applicable. The method becomes simpler if a threshold is performed on the initial color value input dataset, so that the exemplars are segmented into two phases such that each pixel of the 2D cross-section can take only one of two discrete values (0 or 1). In the discussion that follows, it will be assumed that this has already been done.

The generation of a statistically-equivalent 2D slice from an input exemplar is described first, after which the method will be extended to three dimensions. Consider the $K \times K$ exemplar shown in Fig. 1, where K is the side-length in units of pixels. As described above, each pixel is associated with a phase ID, which in the case of this binary composite is either 0 (white) or 1 (black). Let the vector $\vec{r} = (r_1, r_2)$ denote the position of each pixel and let the scalar-valued function $g(\vec{r})$ denote the phase ID of the pixel located at position $\vec{r}$; thus, $g(\vec{r})$ is a full description of the exemplar.

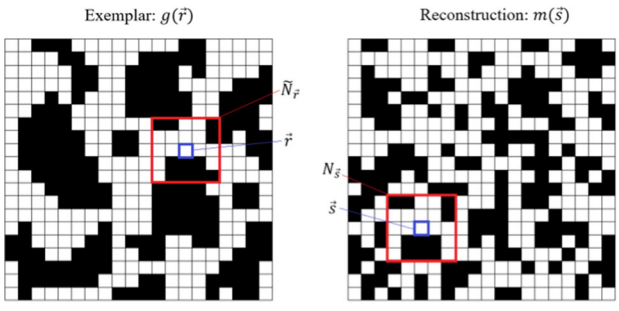

Fig. 1. The neighborhood vectors $\widetilde{N}_{\vec{r}}$ for each point in exemplar space $g(\vec{r})$ are matched with their nearest-neighbor $N_{\vec{s}}$ in the reconstruction $m(\vec{s})$. Although each neighborhood pair is unlikely to perfectly match, as shown here with an error of 1, FLANN is able to quickly find the best pairs that minimize the local error and, thus, by extension, the global error.

We wish to produce an $L \times L$-pixel microstructure that is statistically equivalent to the exemplar described by $g(\vec{r})$. Analogously to $g(\vec{r})$ and in keeping with standard notation in the literature, let $m(\vec{s})$ describe the phase ID values over all pixels in the reconstruction. To generate this statistically-equivalent reconstruction, Kopf et al.[15] introduced the concept of local-neighborhood similarity. Each pixel in $m(\vec{s})$ is centered on an $n \times n$-pixel neighborhood $N_{\vec{s}}$, as shown in Fig. 1 with n = 5. Starting with a randomized $m(\vec{s})$, we would like to iteratively make all neighborhoods $N_{\vec{s}}$ resemble the neighborhoods of $g(\vec{r})$.

To achieve this convergence between the reconstruction and the exemplar, each pixel of $m(\vec{s})$ must find the neighborhood $\widetilde{N}_{\vec{r}}$ of $g(\vec{r})$ that best matches its own local neighborhood $N_{\vec{s}}$. In Fig. 1, notice that $\widetilde{N}_{\vec{r}}$ differs from $N_{\vec{s}}$ only at the bottom right pixel. This confirms that, as required, $\widetilde{N}_{\vec{r}}$ is the closest-matching neighborhood of $N_{\vec{s}}$ in the exemplar. If all of the phase ID values of the pixels in the neighborhood $\widetilde{N}_{\vec{r}}$ equal those in $N_{\vec{s}}$, the reconstruction is in agreement with the exemplar. Otherwise, the two models are in disagreement. This discrepancy is characterized by the error function $E(\vec{s})$ and is defined for each pixel in the reconstruction $m(\vec{s})$.

$$E(\vec{s}) = \sqrt{\sum_{\vec{u}} \left(N_{\vec{s}}(\vec{u}) - \widetilde{N}_{\vec{r}}(\vec{u})\right)^2} \tag{1}$$

A new vector $\vec{u}$ is introduced here, ranging from (1,1) to (n,n), that denotes the position of pixels within each neighborhood. For example, in Fig. 1, $N_{\vec{s}}(\vec{u} = (5,5)) = 0$ and $\widetilde{N}_{\vec{r}}(\vec{u} = (5,5)) = 1$, where the origin of $\vec{u}$ is chosen as the top left corner of each neighborhood; this is the only discrepancy between the two neighborhoods, as mentioned previously. The sum over $\vec{u}$ allows Eq. 1 to capture the total discrepancy between the neighborhoods. $E(\vec{s})$ is minimized when $N_{\vec{s}} = \widetilde{N}_{\vec{r}}$, or when the $n \times n$ neighborhood $N_{\vec{s}}$ of the pixel at position $\vec{s}$ in the reconstruction is identical, in terms of phase ID values, to the best-matching $n \times n$ neighborhood $\widetilde{N}_{\vec{r}}$ in the exemplar. $E(\vec{s})$ is maximized when the neighborhoods are opposites, i.e., the black and white representation of the neighborhoods are inverted. The global error functional is shown in Eq. 2, and captures the total discrepancy between $m(\vec{s})$ and $g(\vec{r})$ by summing over all local neighborhood discrepancies:

$$E_{global} = \sum_{\vec{s}} E(\vec{s}) = \sum_{\vec{s}} \sqrt{\sum_{\vec{u}} \left(N_{\vec{s}}(\vec{u}) - \widetilde{N}_{\vec{r}}(\vec{u})\right)^2} \tag{2}$$

Equation 2 is described as a functional, rather than a function, because it is a function of $m(\vec{s})$, which is itself a function describing the microstructure. The target reconstruction seeks to minimize this global error. This is done in an iterative

fashion, alternating between a search phase and an optimization phase.[15] In the search phase, the closest matching neighborhoods $\widetilde{N}_{\vec{r}}$ are found for all $\vec{s}$. This is a nearest-neighbors search, and previous researchers in solid texture synthesis have generally found the approximate nearest-neighbor (ANN) algorithms implemented with the Fast Library for Approximate Nearest Neighbors (FLANN C++ library) to provide an acceptable compromise between speed and accuracy.[21,22] The details of the internal working of the ANN algorithms are not described here, but their purpose is to organize the total set of possible neighborhoods $\widetilde{N}_{\vec{r}}$ of the exemplar into a k-dimensional (or k-d) tree in such a way that the approximate nearest neighborhood of any $N_{\vec{s}}$ can be found with significantly less computational expenditure than if the exact nearest neighborhood had been sought.[21]

The 3D microstructure is generated by stochastically minimizing a reconstruction loss defined as the sum of element-wise differences between each pixel's local neighborhood and its corresponding nearest-neighbor patch in the parallel exemplar, summed over the three orthogonal exemplars. Thus, in a given plane, each stochastic update searches through the fixed space of all size N^2 patches in the corresponding exemplar. For a fixed neighborhood size, this search space increases quadratically with the exemplar size, while for a fixed exemplar size, the search space decreases quadratically with neighborhood size. However, this benefit comes at the expense of accuracy in the reconstructed microstructure over length scales larger than the neighborhood size.

In their review of the methods described above, Turner et al.[12] clarified the following point: each time the solid texture synthesis algorithm goes through an iteration, the values of each pixel in the reconstruction are updated in order to decrease the global error determined from Eq. 2. This update can be described as a weighted average of the "votes" each pixel receives from all neighborhoods to which it belongs.[20] Consider pixel A and three of its neighbors B, C, and D in the reconstruction $m(\vec{s})$, and focus on the update to the phase ID value of pixel A. Say that from the perspective of pixel B, its local error would decrease if pixel A were switched to the other phase, but, from the perspectives of pixel C and pixel D, pixel A should remain the same to minimize their local errors. If the weights of the "votes" of B, C, and D are equal, pixel A will remain unchanged in the next iteration of the reconstruction in pursuit of the minimization of global error, as calculated from Eq. 2. This mechanism is expressed as:[15]

$$m(\vec{s}) = \frac{\sum_{j=1}^{n^2} \omega_{\vec{s},\vec{s}_j} \widetilde{N}_{\vec{s}_j}(\vec{s})}{\sum_{j=1}^{n^2} \omega_{\vec{s},\vec{s}_j}} \tag{3}$$

Note that the notation in Eq. 3 has been modified slightly to accommodate the relatively large amount of information referenced without using too many subscripts. Consider the pixel at $\vec{s}$ in the reconstruction. It participates in n^2 neighborhoods, because it is within the neighborhood of each of the $n^2 - 1$ pixels that surround it in addition to its own. The index $j = 1, 2, \ldots n^2$ is the counter over those pixels, $\vec{s}_j$, each of which "votes", with the weight $\omega_{\vec{s},\vec{s}_j}$, as to what the phase ID $m(\vec{s})$ should be set to (either 0 or 1) during the next iteration such that its own local error is decreased. These votes are equal to $\widetilde{N}_{\vec{s}_j}(\vec{s})$, which denotes the phase ID of the pixel within $\widetilde{N}_{\vec{s}_j}$ that corresponds to the location of $\vec{s}$. This is best understood through an example.

Suppose that $m(\vec{s})$, as illustrated in Fig. 2, is the initial randomized "guess" for the reconstruction of the exemplar. Let $\vec{s}_y$, $\vec{s}_r$, and $\vec{s}_g$ denote the pixels marked by the yellow, red, and green stars in $m(\vec{s})$. We would like to know what the phase ID value of $\vec{s}_y$ (i.e., $m(\vec{s}_y)$) should be updated to in the next iteration in order to decrease the global error functional $E(m(\vec{s}))$. To do this, we will consider the individual votes of $\vec{s}_r$ and $\vec{s}_g$, although in reality all pixels in the yellow neighborhood will contribute if the neighborhood size is chosen to be 3×3. The best-matching neighborhood of $\vec{s}_r$ is outlined with red in the exemplar $g(\vec{r})$ and is denoted by $\widetilde{N}_{\vec{s}_r}$, and similarly for the green star with $\vec{s}_g$ and $\widetilde{N}_{\vec{s}_g}$. As mentioned previously, these best-matching neighborhoods were determined via the ANN search algorithm using the FLANN library.[22] Notice that the local error of $\vec{s}_r$ would be zero if the yellow star-marked pixel were white, or $m(\vec{s}_y) = 0$. Similarly, the pixel at $\vec{s}_g$ votes for $m(\vec{s}_y)$ to be 0. The weighted average of such votes is calculated over all pixels in the yellow-outlined neighborhood, and the final result is rounded to 0 or 1. The result will be the

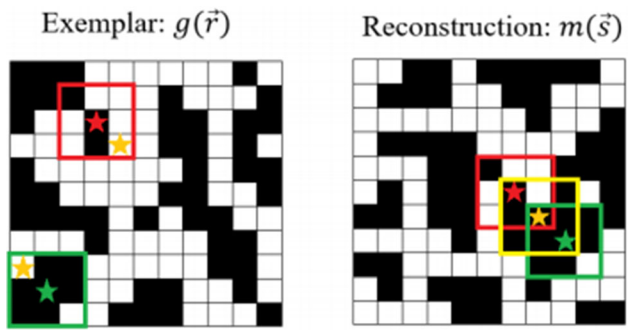

Fig. 2. The reconstruction is driven by the nearest neighbor search, matching neighborhoods in the exemplar to those in the reconstruction. The *red* and *green stars* and associated *boxes* indicate sample neighborhoods and their closest match in the reconstruction; the *yellow star* describes the subsequent voting process. The *red* and *green stars* both vote for the *yellow star* to turn *white*, and only pixels in the *yellow box* in the reconstruction (which are chosen by the nearest neighbor matching) can cast votes for the *yellow star* (Color figure online).

value of $m(\vec{s}_y)$ in the next update of the reconstruction. Note that this rounding operation is not always performed in solid texture synthesis algorithms, but doing so mitigates the possibility of obtaining a blurry reconstruction.[16] The reweighting prescription given in Eq. 4 is implemented after each iteration of the reconstruction

$$\omega'_{\vec{s},\vec{s}_j} = \frac{\omega_{\vec{s},\vec{s}_j}}{1 + \max[0, H(\widetilde{N}_{\vec{s}_j}(\vec{s})) - \theta]} \tag{4}$$

where H denotes the position histogram and $\theta = \frac{1}{K^2}$, which represents the (normalized) probability of each pixel of the exemplar being sampled in the reconstruction if all such pixels are sampled uniformly.[16] This reweighting permits a dynamic adjustment of the relative importance of each pixel to the reconstruction between each iteration, selectively choosing undersampled pixels wherever possible.

Histogram reweighting provides a mathematical bridge between microstructure reconstruction and statistical mechanics by unbiasing a statistical distribution away from its initial state. Consider a Monte Carlo simulation that, through random initialization, begins near a mestastable state that is not representative of the equilibrium properties of the system. Depending on the height of nearby energy barriers, the simulation may never escape its local minimum. To avoid this situation, one can artificially increase the energy of a particular configuration based on the number of times that configuration has been visited along the simulation's trajectory, thus biasing the trajectory away from them. Similarly, over-sampled patches in the exemplar can be down-weighted based on how frequently they are used in the reconstruction, thus ensuring representative sampling of all of the input data. This technique is also commonly used when training neural networks, particularly in the reverse phase of training-restricted Boltzmann machines.

The convergence is further accelerated by a hierarchical upsampling across several resolution levels, implemented after a set number of iterations. Exemplars can be successively downsampled to half the previous resolution to produce a coarse structure, and the lowest resolution is used as the first input exemplar. Initial use of this coarsest exemplar allows the reconstruction to capture large-scale features that might otherwise be missed with only one level of resolution, and circumvents the initial guess issue in inverse problems. Although it is always better to start with an input as close to the desired solution as possible, this can be a difficult requirement to fulfill due to the vast solution space available to a 2D-to-3D generalization; a randomized structure as an initial guess is sufficient for most purposes, and is where the reconstruction begins. The use of multiple levels then permits the initial guess of subsequent levels to use the output from the previous level, allowing the new guess to more closely match the input exemplars while simultaneously increasing the resolution between each level to capture more and more features. When downsampling the exemplars, every set of four pixels becomes a single pixel. The downsampled phase ID value (0 = black and 1 = white) is determined using a rounded average. When the average is calculated to be 0.5, the downsampled pixel is chosen randomly.

To allow for inevitable increases in reconstruction size, the algorithm was wrapped in an asynchronous multiprocessing scheme which divides the reconstruction into multiple sections, enabling a concurrent evaluation of all sections in each iteration while piping the result outside the reconstruction to be plotted on screen. This implementation dramatically sped up the reconstruction: in a reconstruction containing three levels of 40 iterations each, neighborhood sizes in each level of 5, 11, and 19, respectively, and a full reconstruction size of 256^3, the parallelization scheme reduced the runtime from about 37 h to just 8.5 h when each level is divided into six sections, a reduction of almost 80%.

The multiprocessing scheme is accompanied by a compensatory patching tool, serving as a communicator between separate processors to coordinate the actions of each section. A padded buffer at the edges of each section is matched with a similar buffer at the edge of the adjacent section(s) to coordinate the weighted phase ID votes and subsequent updated phase values. Without this patching, distinct divisions between sections are readily visible, rendering the reconstruction unusable for any physical application. However, this nominal defect was adapted to extend the algorithm into non-cubic volumes, namely the reconstruction of plate-like structures. Without the patching tool, the reconstruction can be considered as independently executing as many times as the number of sections determined by the user, with the caveat of stacking each section with the others to produce a cubic volume. The patching scheme is illustrated in Fig. 3. A plate reconstruction, then, can be extracted by cropping and tiling the two thin plate exemplars with themselves to match the dimensions of the largest exemplar and then running the algorithm as normal. In this case, however, we exploit the segregation that occurs without the patching by setting the number of sections equal to the number of tiles of the thin exemplars needed to match the large exemplar's dimensions, rounded up to the nearest integer. Each section is then sequestered from the others and saved to its own automatically generated directory; in this manner, we actually achieve multiple reconstructions of the plate for the same computational cost as one.

The generalization to multiphase materials would provide a useful avenue for the exploration of microstructures with known chemical or phase data. We have developed a prototype methodology

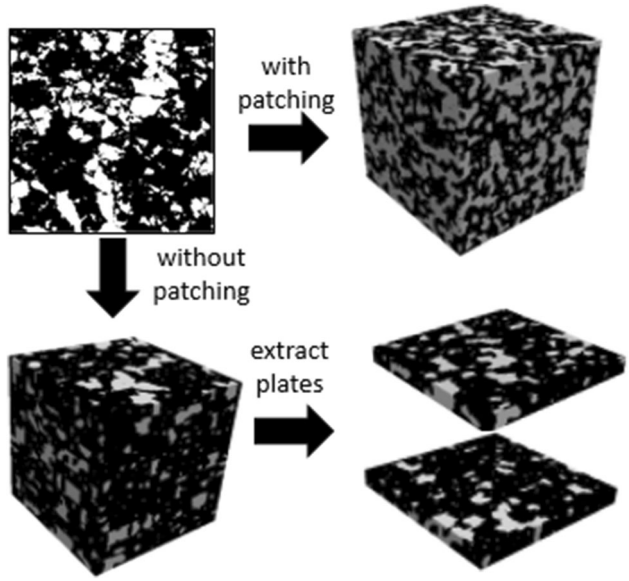

Fig. 3. The multithreading within the reconstruction must be accompanied by a patching scheme to properly coordinate the actions of each section with its neighbor(s) to produce microstructures that contain physically relevant data. However, the patching can be circumvented to produce non-cubic volumes such as plates that also carry statistical similarities to the exemplar, in effect producing a range of reconstructions at once (only two such plates are shown here, but many horizontal section boundaries can be seen on the *bottom-left cube*).

that characterizes spatial correlations between phase pairs within a multiphase material by reconstructing each pair individually. To effect maximum information preservation, the final reconstruction for each pair is superimposed into a 4D master array, from which a number of 3D nano- or microstructures can be extracted. Results from energy-dispersive x-ray spectroscopy (EDS) images of Si and Al in a simulated nuclear waste feed[23] are shown in Fig. 4. Although superimposed phase–pair reconstructions are only a rudimentary attempt to generate multiphasic microstructures, this implementation reduced the complexities of trinary (and greater) reconstructions despite increasing computational time. In its current implementation, the runtime t increases as

$$t = 0.5p^2 - 0.5p \tag{5}$$

where p is the number of phases, reflecting the significant increase in combinatorial pairs as the number of phases increases; however, we believe the trade-off between the greatly expanded informational scope versus the time investment to be worthwhile. When multiprocessing across multiple nodes is fully enabled (rather than partitioning a single node), the nodal count will likely negatively offset the p^2 term in Eq. 5. The reconstructions were executed on a 192-core Dell PowerEdge distributed large-memory system with 32 cores per node and 2.25 TB total memory and a LINPACK rating of 3.78 TFlops. In future versions of HARE, adapting this combinatorial approach to the exemplars would permit the reconstruction of RGB images as opposed to grayscale, significantly expanding the capabilities of the algorithm.

RESULTS AND DISCUSSION

Figure 5a illustrates the process of reconstructing a synthetic 3D microstructure of a $HfAl_3$-Al metal matrix composite from a segmented 2D optical microsocopy image.[3] The reconstruction algorithm is performed for 100 iterations to generate a reconstructed "microstructure" of size 50×50. The progression from the initial randomized reconstruction at the first and 100th iterations of the algorithm is shown in Fig. 5b. It can be seen that the general form of the exemplar is visible by the first iteration, suggesting that the rather small size of the exemplar allowed for rapid convergence of the microstructure. The following 99 iterations cleaned up the reconstruction further to produce a somewhat imperfect tiled pattern of the exemplar that nonetheless captures the overall shape of the exemplar.

The impetus for this algorithm centers on the motivation to produce a statistically similar reconstruction of the exemplars. To verify that the reconstructed volumes are indeed representative of the exemplars, we surveyed and analyzed a range of morphological statistics across several exemplar–reconstruction pairs. DREAM.3D software,[24–26] a tool that is well-suited for analysis of volumetric microstructures, can be used to tally features of the reconstructed microstructure for comparison to the exemplar. For a given reconstruction, DREAM.3D can be used to quantify characteristics, such as the volume fraction of phases, equivalent sphere diameter, distance between features, radius of neighboring features, neighboring features in contact or other parameters of interest. The mathematics of DREAM.3D's workings are available in a series of papers establishing the framework for the recognition and analysis of n-dimensional solids and their associated moment invariants.[27–31] It should be noted that, while DREAM.3D is also adept at microstructure synthesis according to morphological statistics, it currently only generates semi-regular polygonal and ellipsoidal features, which are not space-filling and thus can miss smaller irregular features. HARE is able to construct fully irregular features and performs quite well in tandem with DREAM.3D. DREAM.3D can provide both the 2D and 3D statistics of the exemplars and the microstructure, respectively, while a tool such as MATLAB can be subsequently employed to compare the statistical distributions.

The reconstruction of materials with different textures is illustrated in Fig. 6. The nature of these textures ranges from ideal and isotropic (Fig. 6a, b) to the reconstruction of a real anisotropic material from three orthogonal images (such as the SiC foam

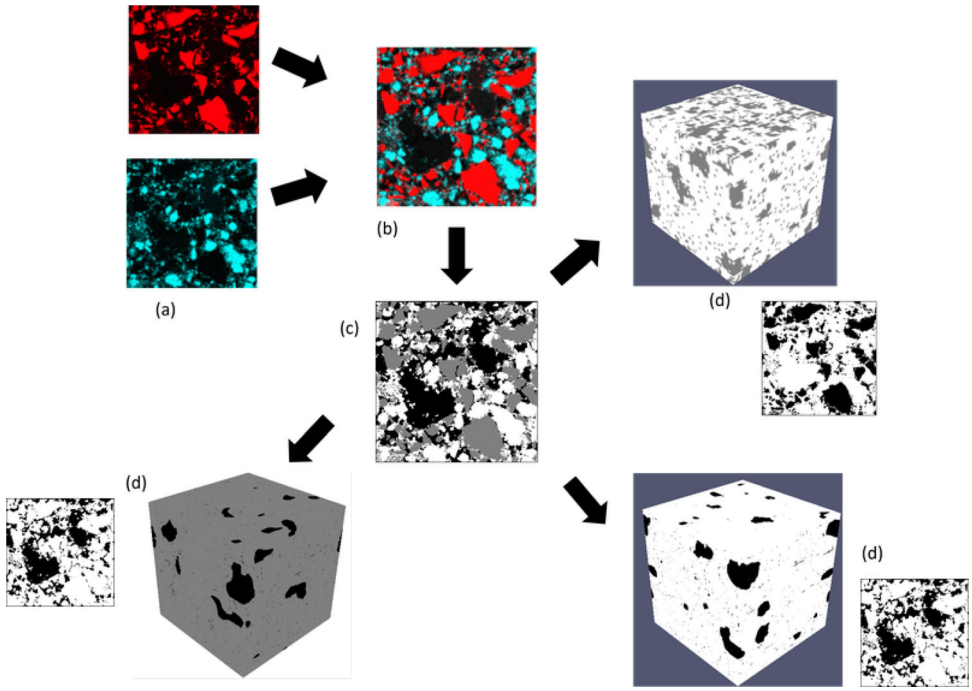

Fig. 4. Multiphase materials can be reconstructed by isolating non-ordered pairs of values in a grayscale image and reconstructing each phase pair separately. Each reconstruction is inserted into a master array, where each "pixel" contains the value from each reconstruction at that same point. This can be extended to arbitrary numbers of phases, although many phases in one image raises the difficulties of characterizing very low volume fraction materials. Here, we combine two RGB images obtained by EDS (a)[23] into a single image (b), then convert to grayscale and segment (c). The segmented image is then split according to each combination of $\binom{p}{2}$, where p is the number of phases, from which a binary reconstruction can proceed as normal (d). Here, three binary phase reconstructions are illustrated.

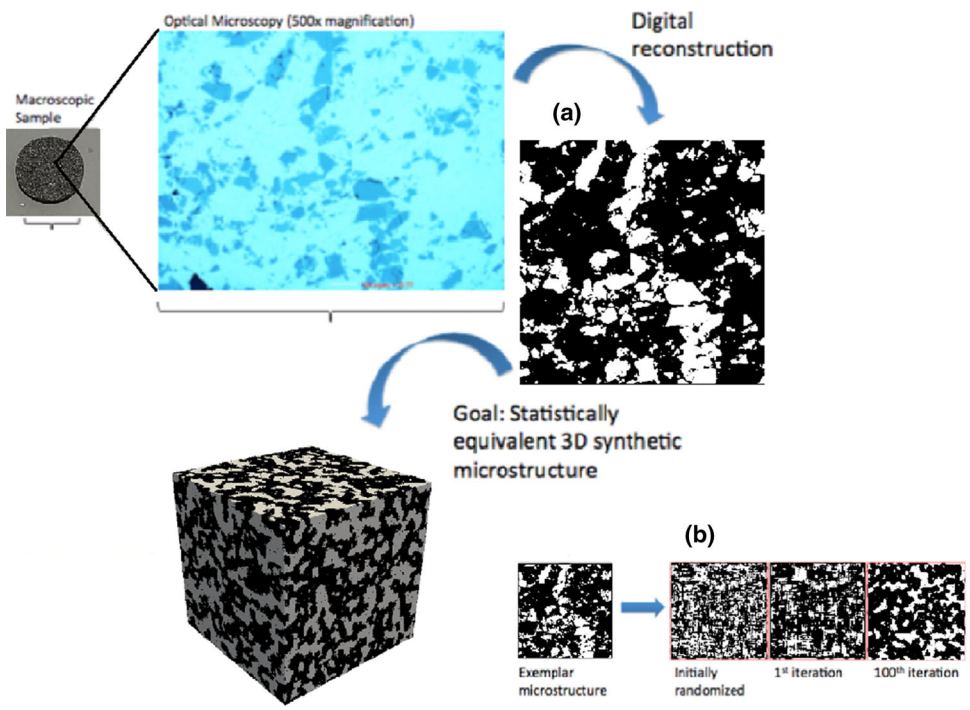

Fig. 5. (a) HARE generates 3D microstructures from segmented 2D exemplars while preserving important characteristics, such as volume fraction of a given phase. An optical microscopy image of a metal matrix composite material was successfully reconstructed into 3D. (b) A proof-of-concept two-dimensional reconstruction of an exemplar, showing the algorithm's progress across 100 iterations.

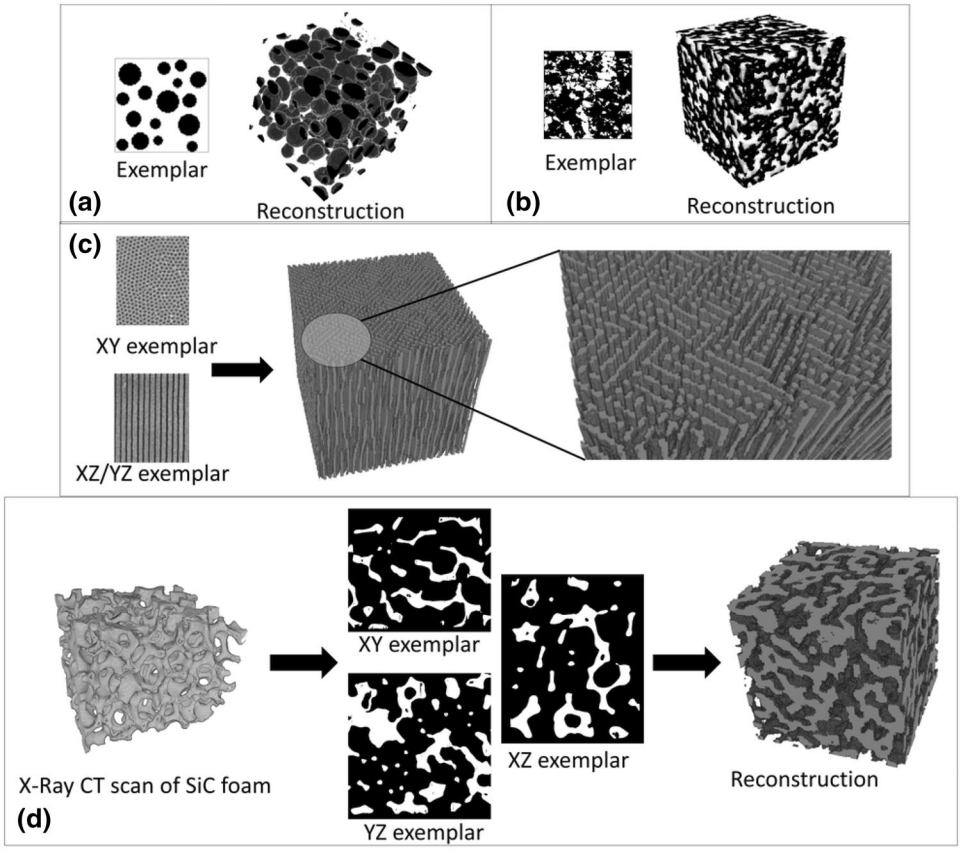

Fig. 6. HARE reproduces isotropic and anisotropic materials from computer-generated images as well as digital scans of real materials. 3D microstructures generated from (a) a single exemplar of a material with spherical inclusions, (b) a single exemplar of a two-phase material, (c) two orthogonal exemplars of a fibrous material, and (d) three orthogonal exemplars of a foam.

shown in Fig. 6d). The reconstruction of the fibrous material from two orthogonal exemplars shown in Fig. 6c results in a 45° arrangement of fibers. For a texture such as this, including a third exemplar would provide the additional information necessary to ensure that the reconstructed microstructure is uniquely defined. The reconstruction of the SiC foam shown in Fig. 6d was based upon three orthogonal slices obtained by x-ray computed tomography. HARE reconstructs a synthetic structure that closely resembles the morphology of the original microstructure.

Repeated tests have demonstrated that use of the histogram reweighting scheme given by Eq. 4 is necessary to generate microstructures that resemble their exemplars; without it, the voting scheme is unable to distinguish between over- and undersampled pixels and adjust the phase ID updates accordingly, and the reconstruction overwhelmingly mis-samples exemplar neighborhoods and produces highly unrepresentative structures. We affirm the similar conclusions reached by both Kopf et al.[15] and Turner et al.[12] regarding the necessity of histogram reweighting to correctly match global statistics in conjunction with neighborhood matching to reproduce local statistics. Similarly, the multithreading must be accompanied by a patching scheme to accurately orchestrate the reconstruction at the boundaries between sections; neglecting it renders cubic reconstructions unusable for any material or physical application. However, plate structures can if necessary be extracted from a non-patched reconstruction while retaining the desired statistical equivalence.

The reconstruction of multiphasic materials has been developed into a simple, yet prototypic, approach. By isolating pairs of phases, we can avoid a complete overhaul of the voting scheme while preserving the exemplars' structures; an earlier attempt at reconstructing all phases at once showed that the solution quickly converged to either a single phase or eliminated the middle phases after only a few iterations. An added benefit of the new prototype sidesteps the inherently biphasic nature of HARE, instead of the all-or-nothing choice of phases in the cubic and plate reconstructions, wherein the reconstructions for each phase pair are allowed to coexist via superposition. This prototype may be expanded by incorporating more sophisticated methods to preserve low-volume fraction phases.

SUMMARY AND FUTURE WORK

The HARE algorithm is a solid foundation for future efforts in multi-scale modeling to support advanced manufacturing and materials discovery. A large part of materials science lies within the development of new materials for specific purposes, and our algorithm provides a convenient and robust base from which to generate statistically representative synthetic microstructures. This work builds upon the hybrid MATLAB/C++-based algorithm of Turner and Kalidindi[12] and offers additional features, such as histogram reweighting, multithreading, and an option for generating plate microstructures. The code was written entirely in Python to provide extensibility and harness the power of open-source scientific computing libraries, toolkits and machine-learning algorithms. Python can easily interface with DREAM.3D workflows and run automated pipelines. As mentioned earlier, the "defects" in the reconstruction produced by the algorithm cannot be ignored when analyzing the material across various scales, since the roles of semi-random defects and small-scale heterogeneity have yet to be fully defined in regard to their effect on bulk material properties and consequent performance. These defects are often precisely the features we wish to capture. This algorithm, when coupled with machine-learning techniques, can be a powerful tool for materials design.[32] For example, synthetically-generated microstructures exhibiting defects could be used to train machine-learning algorithms, such as convolutional neural networks.[33] Additionally, this algorithm could be incorporated into advanced finite element modeling tools.[34] With this approach, the incorporation of embedded sensors into additively-manufactured parts can be evaluated.

By constructing microstructures of specific texture, the fabrication of materials with chosen macroscale properties emergent from microscale features with performance tailored for a given application can be facilitated. We believe this will be a powerful tool when combined with future multiphase reconstruction capabilities. Currently, the algorithm can only reconstruct two-phase materials. By expanding the multi-phase capabilities, the algorithm will become exponentially more useful to researchers who possess chemical and phase data. Further refinements can be incorporated through the inclusion of additional convergence heuristics to narrow the solution space. Two-point spatial correlations would allow FLANN to preferentially select nearest neighbors in the reconstruction that are spatially closest to the associated exemplar neighborhood, the mathematics of which are well understood.[18] Similarly, incorporating a simulated annealing sub-algorithm would push solutions out of local minimums toward the global optimum, a technique often used in combinatorial optimizations.[35] Implementation of these additional heuristics would accommodate low-volume percent materials, which currently tend to converge to single-phase reconstructions and are not representative of the materials from which they were taken. As mentioned earlier, the adaptation to accept multivalued exemplars such as those generated by x-ray fluorescence mapping or EDS would greatly expand HARE's ability to reconstruct the chemical or phase composition of a material.

ACKNOWLEDGEMENTS

This work was supported in part by Albert Kruger of the U.S. Department of Energy's (DOE) Waste Treatment and Immobilization Plant Project of the Office of River Protection and by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI). Javier Morales contributed to the development of the Python code. Michael Jackson of Blue Quartz Software provided guidance on DREAM.3D. X-ray CT images of the SiC foam were acquired by Joshua Kane, Austin Matthews and Robert Siefert of the INL. We wish to thank Tom Kielbaso and Michael Velez from UES, Inc. (Dayton, OH) for providing the optical microscopy images of the $HfAl_3$-Al sample. This work was performed by Battelle Energy Alliance, LLC under DOE Idaho Operations Contract DE-AC07-05ID14517. This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

REFERENCES

1. N. R. Council *Integrated Computational Materials Engineering: A Transformational Discipline for Improved Competitiveness and National Security*. The National Academies Press (2008).
2. R. Feynman, *Eng. Sci.* 23, 22 (1960).
3. D. Guillen and W. Harris, *Metall. Mater. Trans. E* 3, 123 (2016).
4. R. Bostanabad, Y. Zhang, X. Li, T. Kearney, L. Brinson, D. Apley, W. Liu, and W. Chen, *Prog. Mater. Sci.* 95, 1 (2018).
5. H. Garmenstani, M. Baniassadi, D. Li, M. Fathi, and S. Ahzi, *Int. J. Theor. Appl. Multiscale Mech.* 1, 134 (2009).
6. M. Baniassadi, B. Mortazavi, H. Hamedani, H. Garmestani, S. Garmestani, M. Fathi-Torbaghan, D. Ruch, and M. Khaleel, *Comput. Mater. Sci.* 51, 372 (2012).
7. H. Garmestani, M. Baniassadi, D. Li, M. Fathi, and S. Ahzi, *Int. J. Theor. Appl. Multiscale Mech.* 1, 134 (2009).
8. D. Fullwood, S. Niezgoda, B. Adams, and S. Kalidindi, *Prog. Mater. Sci.* 55, 477 (2010).
9. A. Gokhale, A. Tewari, and H. Garmestani, *Scr. Mater.* 53, 989 (2005).
10. D. Fullwood, S. Niezgoda, and S. Kalidindi, *Acta Mater.* 56, 942 (2008).
11. B. Adams, H. Garmestani, and G. Saheli, *J. Comput. Aided Mater. Des.* 11, 103 (2004).
12. D. Turner and S. Kalidindi, *Acta Mater.* 102, 136 (2016).

13. H. Xu, D. Dikin, C. Burkhart, and W. Chen, *Comput. Mater. Sci.* 85, 206 (2014).
14. C. Yeong and S. Torquato, *Phys. Rev. E* 58, 224 (1998).
15. J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.-T. Wong, Solid Texture Synthesis from 2D Exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, vol. 26, pp. 2:1–2:9 (2007).
16. J. Chen and B. Wang, *Vis. Comput.* 26, 253 (2010).
17. V. Sundararaghavan, *Integr. Mater. Manuf. Innov.* 3, 19 (2014).
18. S. Niezgoda, D. Fullwood, and S. Kalidindi, *Acta Mater.* 56, 5285 (2008).
19. R. Urs, J.D. Costa, and C. Germain, *IEEE Trans. Image Process.* 23, 1820 (2014).
20. N. Pietroni, P. Cignoni, M. Otaduy, and R. Scopigno, *J. Latex Class Files* 6, 1 (2007).
21. L. Liang, C. Liu, Y. Xu, B. Guo, and H. Shum, *ACM Trans. Graph.* 20, 127 (2001).
22. M. Muja, *FLANN—Fast Library for Approximate Nearest Neighbors: User Manual*. https://github.com/mariusmuja/flann.
23. D. Dixon, M. Schweiger, B. Riley, R. Pokorny, and P. Hrma, *Environ. Sci. Technol.* 49, 8856 (2015).
24. M. Groeber and M. Jackson, *Integr. Mater. Manuf. Innov.* 3, 5 (2014).
25. A. Rollett, D. Saylor, J. Firdy, B. El-Dasher, A. Brahme, S. Lee, C. Cornwell, and R. Noack, *AIP Conf. Proc.* 712, 71 (2004).
26. A. Rollett, S.-B. Lee, R. Campman, and G. Rohrer, *Annu. Rev. Mater. Res.* 37, 627 (2007).
27. M. Graef, *Representation and Reconstruction of Three-dimensional Microstructires in Ni-Based Superalloys* (2010) Carnegie Mellon University. p. 32.
28. J. MacSleyne, J. Simmons, and M. Graef, *Model. Simul. Mater. Sci. Eng.* 16, 045008 (2008).
29. A. Mamistvalov, *IEEE Trans. Pattern Anal. Mach. Intell.* 20, 819 (1998).
30. M. Groeber, S. Ghosh, M. Uchic, and D. Dimiduk, *Acta Mater.* 56, 1257 (2008).
31. J. Simmons, P. Chuang, M. Comer, M. Uchic, J. Spowart, and M.D. Graef, *Model. Simul. Mater. Sci. Eng.* 17, 025002 (2009).
32. D.S. Bulgarevich, S. Tsukamoto, T. Kasuya, M. Demura, and M. Watanabe, *Sci. Rep.* 8, 2078 (2018).
33. A. Kitahara and E. Holm, *Integr. Mater. Manuf. Innov.* 7, 148 (2018).
34. S. Novascone, P. Medvedev, J.W. Peterson, Y. Zhang, and J. Hales, *J. Nucl. Mater.* 508, 226 (2018).
35. P. Czyżżak and A. Jaszkiewicz, *J. Multi Criteria Decis. Anal.* 7, 34 (1998).