Towards Random Generation of Microstructures of Spatially Varying Materials from Orthogonal Sections

Robert C. Foster[1],Scott Vander Wiel[1], Veronica Livescu[1], Curt Bronkhorst[2]

Los Alamos National Laboratory[1], University of Wisconsin[2]

August 26, 2019

LA-UR-19-28535

Abstract

New additive manufacturing techniques create materials with microstructures that are not easily represented by traditional simulation techniques, such as ellipsoid packing algorithms. Building upon previous techniques, a new framework for geometric simulation is introduced which allows for more flexible simulation and estimation of microstructure in materials with spatial variation, such as additively manufactured materials. The framework recreates growth of grains, allowing for a per-grain specification of growth properties, which is unavailable to techniques that rely on optimal packing procedures. An estimation technique is also introduced that allows for estimation of the hyperparameters of the growth model based only on a partial sample of orthogonal slices.

Keywords— Microstructure Simulation, Additive Manufacturing, Elliptical Growth Model

1 Introduction

The advent of additive manufacturing demands new techniques to understand both the manufacturing process and resulting properties of the new materials. In particular, models are needed in order to produce digitally simulated microstructures, which can then be used for further computational analysis in order to determine properties of the additively manufactured materials. Currently, geometric methods for microstructure simulation - which rely on growth of polygons in space (such as in the Dream3D program) rather than simulations of physics - often use an optimal packing of ellipsoids in order to create a simulated microstructure ([1–9]). Validation of the packing algorithm may be found in [10], and a general state-of-the-art review of computational microstructure reconstruction may be found in [11]. A discussion of microstructure variance as a stochastic process may be found in [12].

The optimal packing method is not easily adaptable to spatial variation in the grain data set. Additively manufactured materials, as shown below in Figure 1, may exhibit extreme spatial variation of a type not seen in traditional microstructures. Even those without such extreme variation may show varying grain morphologies in different locations due to the manufacturing process. A microstructure generation method based on an annealing procedure that optimally packs grains may not be flexible enough to recreate such microstructures, as the process of reordering grains in space destroys any spatial trends that may be present.

This paper focuses on creating a framework for generation of statistically representative grain volumes that is more easily adaptable for spatial variation. A variation of the elliptical growth algorithm of [13] is used for simulation of microstructures, with new statistical distributions introduced (particularly in the area of orientated grain growth) in order to simulate microstructures from a small number of parameters for each grain and hyperparameters controlling the statistical distributions of grain parameters across the sample of material. The hyperparameters of these statistical distributions may be estimated to produce traditional spatially homogeneous microstructures, and said distributions can be more flexibly adapted to represent spatial trends in the material. Furthermore, a new type of area-weighted cumulative distribution function is introduced to compare grain characteristics between microstructures. This area-weighted cumulative distribution function is used to adapt estimation procedures to determine hyperparameter estimates in the presence of differing simulation container volumes and grain densities as growth occurs. Finally, this growth algorithm is computationally feasible, requiring only a few minutes of growth time on a standard laptop for a simulated microstructure of one million voxels.

Section 2 describes the grain growth model and new simulation methods

used in this paper to generate microstructures, while Section 3 describes how the hyperparameters of the growth model may be estimated from a set of orthogonal EBSD images. Section 4 applies this to a sample of homogeneous material using the IN100 data set included with the Dream3D software and described in [14], while Section 5 shows the flexibility of the method in adaptation to an additively manufactured data set that exhibits spatial variation and missing data. Section 6 presents a few avenues for further improvement of the model and research.

2 Growth Model

Modeling grain growth by ellipsoids has a long history within materials science ([7]). This paper continues the tradition, using a modified form of the elliptical growth process as described in [13]. The technical details of the growth process in [13] are used; however, this papers expands the research by describing a new, more general procedure for simulation and estimation. The main differences are as follows: (1) [13] relies on having the full serial-sectioned 3D microstructure available for estimation. In practice, this is a time-consuming process which is seldom carried out. Here, it is assumed only that a set of orthogonal EBSD slices is available, and as shown in Section 5, even all orthogonal directions may not be available. Given this, the goal is not exact grain-by-grain recreation of a given sample, but rather simulation of new microstructures which share distributional characteristics with the original sample, as determined by the orthogonal slices. By focusing on properties in two-dimensional slices rather than a full grain-by-grain reconstruction, estimation may proceed with less information, even as spatial variation in the microstructure is estimated as well, and multiple random realizations of representative grain ensembles can be generated. (2) For the purpose of simulation, [13] uses a Nataf transformation in order to generate a marked Poisson process, such that the distributions of marks (including relevant correlations) match those of a sample data set. The goal of this paper is to develop a much more flexible framework, controlled by known hyperparameters for a growth process, so that hyperparameter estimates may be found in order to generate simulated microstructures which match, in terms of measurements taken on two-dimensional slices, those of a sample data set. Lastly, (3) rather than conducting an exhaustive query of all grains to determine which grain should be assigned to a single element of the simulated microstructure, the method described here mirrors the growth process itself by growing ellipsoids in iterations and allowing for consideration of intersecting grains. In this way, grain assignments are computed more efficiently.

The end result of a run of the elliptical growth model is a simulated microstructure consisting of a fully voxelated volume in which each voxel is assigned

Figure 1: An EBSD image of a sample of additively manufactured 304L stainless steel created at Los Alamos National Laboratory. The image shows severe spatial variation of grain characteristics, caused by the radial cooling from a weld bead formed in the LENS (laser engineered net shaping) process..

to a single grain. As such, a process is required for generating properties of each grain within the microstructure, which will then be used to assign voxels to grains. In general, the crystallographic orientation must also be defined to create a fully representative microstructure, but this aspect is not pursued in this paper. The per-grain parameters considered in this paper are nucleation site, growth velocities along the principal axes of the ellipse, and a rotation matrix controlling the principal axes.

The procedure for generating a simulated data set is then as follows:

1. Pick the number of grains N to be generated, the minimum distance r between nucleation sites, and the total size of the simulation volume, including size of voxels.

2. Generate a set of N nucleation sites consisting of points $X_g = (x_g, y_g, z_g)^T$ (where $g = 1 \dots N$ indexes grain) in three-dimensional space.

3. Choose model hyperparameters, consisting of central orientation S from the 3D rotation group SO(3), orientation concentration κ_S, target aspect ratios k_1 and k_2, and velocity concentration κ_v.

4. Simulate parameters for each grain, consisting of an orientation $O_g \in SO(3)$ and growth velocites $[v_{1,g}, v_{2,g}, v_{3,g}]$ so that the ratio of the expectations of growth velocities is equal to the target mean aspect ratio.

5. Use the elliptical growth algorithm to grow out each grain, according to distance metric

$$D(x, X_g) = \left[(x - X_g)^T O_g^T (\text{diag } v_g)^{-2} O_g (x - X_g)\right]^{1/2} \tag{1}$$

   where x is a 1x3 vector of the voxel center, X_g is a 1x3 vector of the nucleation site of grain g, O_g is the 3x3 orientation matrix for grain g, and diag v_g is a 3x3 matrix with diagonal elements given by the three growth velocities.

In this procedure, the simulation volume and minimum distance between nucleation sites is chosen a priori. The hyperparameters which control the statistical distributions of the grain parameters are the target aspect ratios k_1 and k_2, the velocity concentration κ_v, central orientation S, and orientation concentration κ_S. These statistical distributions are used to simulate parameters for each grain - the nucleation sites (x_g, y_g, z_g), the growth velocities $[v_{1,g}, v_{2,g}, v_{3,g}]$, and the rotation matrices O_g. The manner in which each parameter is simulated and each hyperparameter affects the growth process is as follows.

For the generation of nucleation sites, a hard-core Matérn process is used. The hard-core Matérn process is described in [15]. This process places nucleation sites uniformly within a given space, subject to the condition that no two nucleation sites are allowed to be within distance r of each other. Though it may require adjustment for any sample of material, a default value of $r = 0.2$ microns works well for the samples of material considered in Sections 4 and 5. At $r = 0$, nucleation sites may (unrealistically) be placed immediately next to each other, while as r increases the points will increasingly trend toward a regular lattice in order to satisfy the distance constraint.

To generate growth velocities v_1, v_2, and v_3, three hyperparameters are used: the mean aspect ratio k_1 of the first over second principal axis of the ellipsoid, the mean aspect ratio k_2 of the second over third principal axis of the ellipsoid, and a concentration hyperparameter κ_v which controls the variance of the growth velocities around the means. Fixing the expected velocity for the first principal axis at $E[v_1] = 1$ (which is used in all simulations in this paper, though some other convenient value may also be used), then $E[v_2]$ and $E[v_3]$ may be easily calculated so that the ratios of expected values for velocities match the aspect ratios k_1 and k_2. This is appealing in that given a set of aspect ratios, the particular values of the growth are not necessarily important - rescaling the growth velocities simply means that the ellipsoids will grow at a slower or faster rate, but the resulting microstructure will be identical. Hence, the particular growth velocity values may be adapted to the elliptical growth algorithm, so long as aspect ratios are maintained. Then given the concentration hyperparameter κ_v, the velocities for a given grain are simulated from a gamma distribution.

$$v_i \sim Gamma\left(\kappa_v, \frac{\kappa_v}{E[v_i]}\right) \tag{2}$$

where the mean is calculated as the ratio of the two hyperparameters, and so $E[v_i] = \kappa_v/(\kappa_v/E[v_i])$. This gives a formula for the standard deviation of the velocities around the expected value as $sd(v_i) = (E[v_i])^2/s$. Note also that if two independent random variables X and Y are gamma distributed, then the quantity X/Y follows a generalized beta prime distribution, which is closely related to the beta distributions used to simulate aspect ratios in [6] and the Dream3D program, though the ratio of expecations of random variables is not, in general, equal to the expctation of ratios.

For large values of the concentration hyperparameter κ_v, the sample aspect ratios will necessarily be close to the target aspect ratio, while smaller values allow for a greater spread around the target. Functionally, large κ_v values will cause all grains to grow nearly uniformly along their axes, giving non-convex grain boundaries, while smaller values will allow for much more variation in growth along

the axes, producing more free-form grain shapes. Furthermore, the distributions will tend to be skewed towards larger aspect ratios for smaller κ_v values and closer to normally distributed for larger κ_v values. This is shown below in Figure 2.

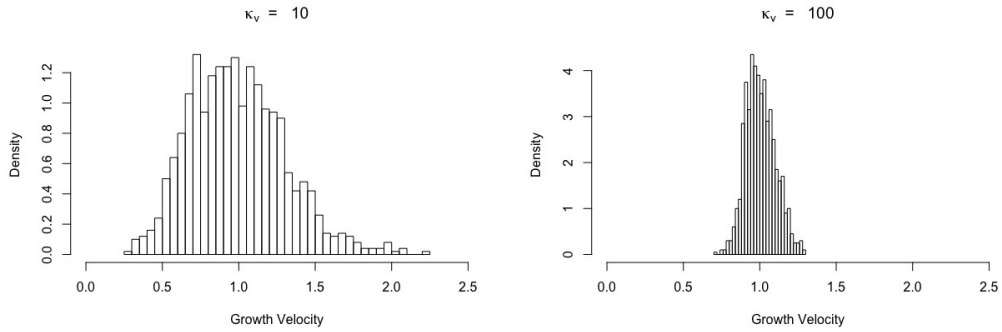

Figure 2: Two histograms of simulated growth velocities are shown, both from gamma distributions with expected growth velocities $E[v] = 1$. On the left, $\kappa_v = 10$ is used. On the right,$\kappa_v = 100$ is used. Smaller values of κ_v increase the variance of potential growth velocities for grains, and allow for a skewed distribution that occasionally produces grains with extreme growth in one direction. At the theoretical $\kappa_v = \infty$, the growth velocity would be $E[v]$ for all grains with no variation, and so the growth velocity for a given direction would be identical for all grains, producing grains similar in morphology to those of a Voronoi tessellation for equal aspect ratios, and non-convex grain boundaries in the case of unequal aspect ratios and preferential growth direction.

For generating random rotations, the uniform axis random spin (UARS) method of [16] and [17] is used. UARS distributions have been used to model crystallographic orientation; however, the statistical properties of the method make it ideal for modeling rotations as well. The UARS method requires a central rotation matrix S, which may be parameterized in a usual way (Bunge-Euler angles, for example) and a concentration hyperparameter κ_S, which controls how closely the random orientations cluster around the central orientation S. The UARS system generates an axis uniformly on the unit sphere and then spins the central rotation S round the axis by a random spin angle. The κ_S hyperparameter in UARS controls a circular symmetric distribution (about 0), which generates the random spin angles.

Though any circular distribution may be used, for this paper a Von Mises

distribution was applied to generate the random perturbations. The Von Mises distribution ranges from $-\pi$ to π and has probability density function

$$f(r|\kappa_S) = \frac{e^{\kappa_S \cos(r)}}{2\pi I_0(\kappa_S)}$$

where $I_0(\kappa_S)$ is the Bessel function of order 0. A value of $\kappa_S = 0$ gives a probability distribution that is uniform on the range $-\pi$ to π, while positive values of κ_S give a bell-shaped probability distribution centered at 0 whose variance decreases as κ_S increases. The effect of this on the distribution of simulated orientations is that $\kappa_S = 0$ produces orientations which are nearly uniform in the set of all rotations, while larger κ_S values create rotations that are increasingly closer to S. This is shown below in Figure 3. Note that the standard deviation of the spin angle decreases rapidly as a function of κ_S.

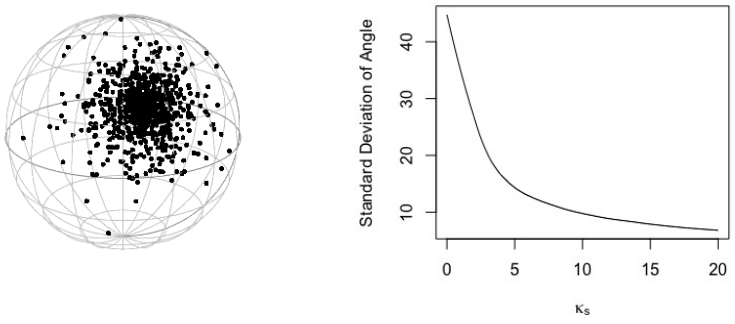

Figure 3: On the left, 1000 rotations from a UARS simulation with central orientation S given by Bunge-Euler angles $(\pi/4, \pi/4, \pi/4)$ controlling the orientation and concentration parameter $\kappa_S = 10$. A projection of the primary axis onto the unit sphere is shown. On the right, the standard deviation of spin angle (in degrees) of the rotated axes away from the central orientation S is plotted as a function of the concentration parameter κ_S. It is clear that increasing the κ_S value increases concentration around the central orientation S.

For the final step, the ellipsoids are grown in the voxelated space and distances are calculated only in the case of grain collisions, as shown in the pseudocode of Appendix A. Though not the main focus this paper, significant effort was devoted to making this growth process computationally feasible for use. To

that end, an algorithm was developed which models the growth process by growing ellipsoids within the voxelated space and halting growth in a direction upon collision with another grain. A space consisting of [100 x 100 x 100] voxels and 1000 grains takes roughly two to five minutes to compute on a Macbook pro, depending on the growth parameters (primarily the velocities), though the algorithm will require fine tuning for other sets of parameters, number of grains, or voxel number and dimension, and poor tuning may increase the required computation time. Similarly, a space consisting of [120 x 200 x 200] voxels and 2500 grains takes approximately 10 minutes, as compared to 8 minutes for a similarly sized model in [13]. This code was developed and run in the R programming language, and would likely increase in speed by orders of magnitude if ported to a lower-level language such as Fortran. Furthermore, the code was run sequentially, but parallelization of aspects of the code is possible. Pseudocode of the algorithm and dependent functions can be found in Appendix A.

Figure 4 shows a slice taken from a simulated data set generated using the procedure. The parameters for the growth were specifically chosen to produce elongated grains along a preferential growth direction in order to demonstrate the ability of the procedure to generate non-equiaxed and convex grains. A normal quantile plot of log equivalent sphere diameters for the simulated data set is also shown in Figure 4. The quantile plot indicates that the distribution of equivalent sphere diameters - proportional to the cube-root of the grain volumes - is approximately log-normal, but with departures from log-normality at the tails. In particular, the simulated data set shows slightly too many small grains and slightly too few large grains to accurately fit a log-normal distribution. [18] and [19] analyze an serially sectioned microstructure - the IN100 set, discussed further in Section 4 - and show that it also exhibits approximate log-normalityof grain sizes with departures at the tails, though with equiaxed rather than elongated grains and with slightly too many rather than too few large grains than log-normality predicts. Preprocessing and postprocessing are suggested in order to more flexibly fit the simulated data set to the observed data set, and such tools could be implemented for this procedure as well. Taken as a whole, it is clear that this procedure is generating physically plausible microstructures which are similar to those from more traditional simulation techniques.

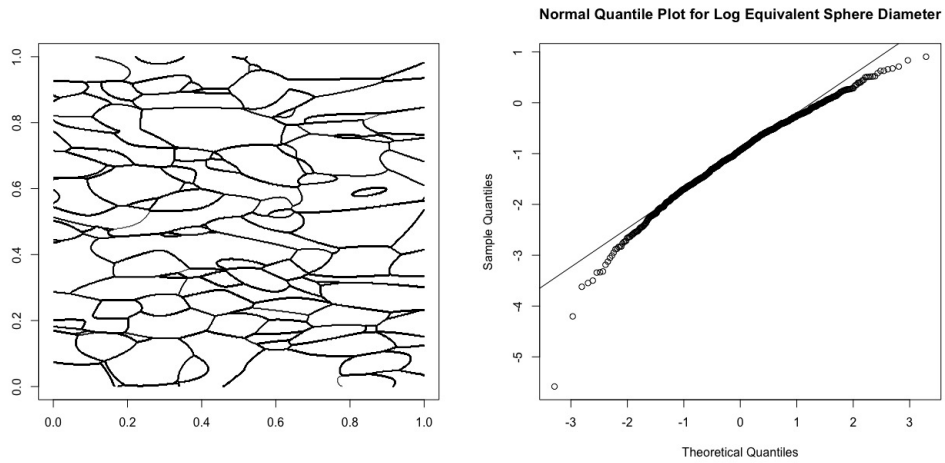

Figure 4: On the left, a sample slice taken from a simulated data set with growth parameters $\kappa_S = 10$ and Bunge-Euler angles $(0, \pi/2, 0)$ controlling the growth direction, and aspect ratios $k_1 = 1$ and $k_2 = 0.3$ with concentration parameter $\kappa_v = 10$. A total of 1000 grains was simulated within a 10x10x10 box with voxel side length 0.025, for a total of 64 million voxels assigned to grains. On the right, a normal quantile plot for the log equivalent sphere diameters of grain volumes taken from the same data set.

3 Estimation Procedure

As serial sectioning can be prohibitively time consuming and expensive, it is necessary to devise a method that may operate in the presence of limited data - potentially with only a few orthogonal slices taken in each direction. As [7] notes, the problem of reconstructing materials from limited information (such as orthogonal slices) has a long history. The problem of reconstruction from serial-sectioned EBSD images is described in [20] in particular. State of the art techniques for the general problem of computational microstructure characterization and deconstruction are described in [11], and the particular problem of elliptical grain reconstruction from orthogonal slices has been considered in [21]. Here, an alternate approach is taken.

If one were to attempt a full grain-by-grain reconstruction of a microstructure, then estimation would focus primarily on determining the parameters, such as the growth velocities $[v_{1,g}, v_{2,g}, v_{3,g}]$ and the orientation matrix O_g, for each grain. However, the main targets of estimation for this paper are not the param-

eters of each grain, but rather the hyperparameters controlling the (potentially spatially-varying) distributions of grain parameters. Only orthogonal slices are assumed to be available, and so full grain-by-grain reconstruction by estimating parameters for each three-dimensional grain is not possible - only characteristics of the two-dimensional slices themselves can be determined. The goal, then, is to use these characteristics to estimate hyperparameters of the growth model described in Section 2 such that orthogonal slices taken from the simulated microstructures show the same grain characteristics as orthogonal slices taken from the real sample of material, according to some measure of similarity.

For the measure of similarity between simulated and observed microstructures, the probability density function $f(x)$ of grain characteristics has often been considered, and close matches between the proability density functions of observed and simulated data sets are presented as evidence of good fit of the model. Typically this has been performed on grain parameters, or three-dimensional quantities such as the grain volume (or transformations such as the equivalent sphere diameter) or nearest neighbor distance. These are not available for the two-dimensional orthogonal slices, but empirical distributions of two-dimensional grain characteristics such as grain areas, aspect ratios, and axis angles are available.

Rather than estimating the simple probability density functions, however, the area-weighted cumulative distribution function is used for direct comparison between data sets. For a given slice with grains indexed by g and corresponding grain area A_g, the area weighted cumulative distribution function for a quantity a_g of a grain is

$$G(a) = \frac{\sum_g A_g * I(a_g \leq a)}{\sum_g A_g} \tag{3}$$

For example, if x_g is the aspect ratio of an ellipse approximation to the 2D, grain, then $G(x)$ is the fraction of area occupied by grains with aspect ratio no greater than x. In this way, the effect of small grains upon the CDF is minimized and increasingly more weight is given to increasingly large grains, which can be expected to have larger influence on the behavior of the material. In practice, EBSD images computed from samples of materials often have unassigned pixels (or as software output identifies them, pixels assigned uniquely and solely to their own grain) which are typically ignored for the purpose of calculating statistics such as the average equivalent sphere diameter. But this process is sensitive to the cutoff area below which grains are ignored. Area weighting presents a modified version of the cutoff idea, allowing small grains to be nearly discounted for the purpose of determining distributions of grain characteristics, while giving increasingly more weight to increasingly larger grains.

In this paper, for each grain within a slice the moment of inertia tensor

is calculated and the grain shape is approximated by the ellipse having the same tensor. The aspect ratio and rotation angle of the grain are defined as those of the approximating ellipse. Since the slices exist as collections of labeled pixels, the area of a particular grain is calculated by simply summing the total number of pixels assigned to said grain and multiplying by the area of a single pixel. Again, these characteristics are taken on two-dimensional ellipses rather than three-dimensional ellipsoids due to the nature of the data as orthogonal slices. Hence, there is only one distribution of aspect ratios, one of rotation angles specifying the ellipse orientations, and areas are used rather than volumes. Assuming that slices taken in the same direction exhibit homogeneous characteristics, these characteristics can then be pooled over all slices in the direction to create a CDF, which is area weighted according to Equation 3. An example of an area-weighted distribution of aspect ratios taken from orthogonal slices in the X direction is shown below in Figure 5.

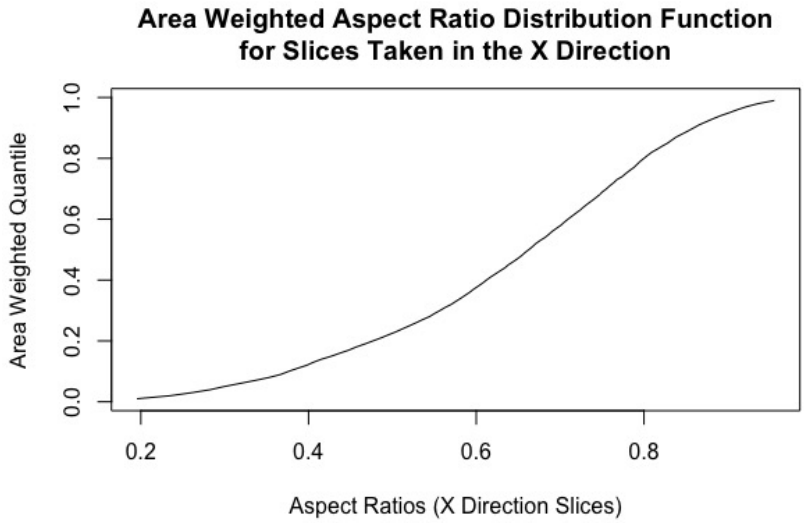

Figure 5: The area-weighted distribution function of aspect ratios for slices taken in the X direction from the IN100 data set included as a sample set with Dream3D and examined in Section 4.

A distance statistic is created by summing the minimum distance between area-weighted CDFs of the simulated and observed data sets over several quantities of interest.

$$D(G^{sim}, G^{Obs}) = \sum_{d=1}^{3} \sum_{q=1}^{3} \sup_{a} |G_d^{sim}(a_q) - G_d^{obs}(a_q)| \tag{4}$$

In Equation (4) above, the sum is taken over the three directions - identified as $x, y,$ and z to follow convention - indicated by $d = 1, 2, 3$ and the three quantities of interest $q = 1, 2, 3$ respectively indicating the area of grains a_1, aspect ratios of grains a_2, and angles of grains a_3. The quantity $G_d^{sim}(a_q)$ is the area-weighted CDF for slices taken from the geometrically simulated data set in direction d and analyzed for quantity a_q, while the quantity $G_d^{obs}(a_q)$ is the same for slices taken from the observed data set. For example, $G_3^{obs}(a_2)$ would indicated the distribution of aspect ratios of grains over all slices taken in the z direction for the observed data set (that is, taken from actual data, not simulated).

Point estimates of hyperparameters for the growth process described in Section 2 are those that produce a simulated data set which minimizes the distance statistic in equation 4. In order to determine the best fit, it may be necessary to take into account the density of grains and total microstructure/simulation cube volume - both of these will influence, at a minimum, the distribution of grain areas.

This issue of differening volumes and grain densities presents a problem when attempting to compare the simulated data sets with observed data sets. Weighting by area deals well with the density issue; however, the range of grain areas is still affected by the differing volumes - larger volumes will produce larger grains, all else being equal. Given area-weighted CDFs $G_d^{sim}(a_1)$ and $G_d^{obs}(a_1)$ for the simulated and observed distributions of area, respectively, a scaling constant K is introduced which minimizes the quantity

$$\begin{aligned}(\sup_{a_1} |G_1^{sim}(K \times a_1) - G_1^{obs}(a_1)| + \sup_{x_1} |G_2^{sim}(K \times a_1) - G_2^{obs}(a_1)| \\ + \sup_{a_1} |G_3^{sim}(K \times a_1) - G_3^{obs}(a_1)|)\end{aligned} \tag{5}$$

That is, K is the scaling constant which, when applied to the x, y, and z direction slices each individually, minimizes the area portion of the distance metric given in equation 4. This was calculated by a simple computational minimization scheme.

This scaling constant is necessary in order to directly compare simulated microstructures grown in different sized simulation cubes and with different nucleation site densities. Simple ratios of grain areas and densities proved inadequate in equalizing distributions grown from the same hyperparameters but in different sized cubes, especially in the presence of elongated grains.

A library of simulated data sets was created in order to facilitate estimation. To show the flexibility of the approach - that simulated data sets may be

compared to observed data sets with different microstructure total volumes and grain densities - each simulated data set used 1000 grains grown within a [10 x 10 x 10] μm sized volume, divided into voxels of side length 0.1 μm, for a total of one million voxels per data set - this nucleation site density of 1 grain per cubic micron is not radically different than densities of observed material samples considered, but different enough that grain density effects will become apparent in the distributions of grain characteristics and must be accounted for using Equation 5. These simulated data sets spanned 15 equidistant aspect ratios between 0.1 and 1.0 for each of k_1 and k_2 individually, four values of κ_S (0, 1, 5, and 10) indicating no, low, medium, and high clustering of orientations were used, along with three values of κ_v (1, 10, 100) indicating low, medium, and high clustering of aspect ratios around their target values.

The set of orientations was determined from a space-filling sequence in SO(3), the space of rotation matrices. The deterministic method of [22] was used to generate a space-filling sequence which provides "incremental generation, optimal dispersion-reduction with each additional sample, explicit neighborhood structure, the lowest metric distortion for grid neighbored edges, [and] equivolumetric partition of SO(3) into grid regions." This method does generate rotations that are mirrors of each other along either the x, y, or z axes, and these duplicate rotations were identified and removed from the set of orientations considered - as the ellipsoids grow symmetrically, a simple rotation of one hundred and eighty degrees along any ellipsoid axis should produce an identical voxelated space. A total of 370 unique default orientations were considered, for 999000 total simulated data sets.

It is clear that identifiability of hyperparameters, especially in the case of equiaxed grains, is likely to be an issue – there may be multiple sets of hyperparameters which produce nearly the same fit for any data set, and when there is no preferential direction for grain orientation any estimation procedure should return an orientation concentration parameter near $\kappa_S = 0$, with the central orientation S playing little role in generation of orientations. As such many possible values of S will be nearly equally likely. In other cases with preferentially elongated or flattened grains, S will be well identified by the data and the method should be able to select a set of hyperparameters that simulate microstructures with slice characteristics matching those of the EBSD images themselves.

For this paper, the distributions of grain area, grain aspect ratio, and orientation angle were considered for each of three orthogonal directions - slices taken along the X, Y, and Z axes, respectively - for a total of nine area-weighted CDFs. The best-fit match is given by the sum of distances, including scaling for areas as in equation 5 above.

4 IN100 Example

To test this methodology on a sample of material with a standard morphology, the IN100 data set included with the Dream3D software was reconstructed with a standard pipeline and the resulting voxelated space was converted to orthogonal slices, and information on grain areas, aspect ratios, and angles was then extracted.

Nickel-based superalloys like IN100 are commonly used in turbine engines because of good performance at elevated temperatures ([23]), and the fine grain size allows for grains to be counted on the mesoscale ([18]), which, in combination with serial sectioning, gives a full and clear picture of the microstructure. A full reconstruction of the IN500 data set using Dream3D is shown below in Figure 6).

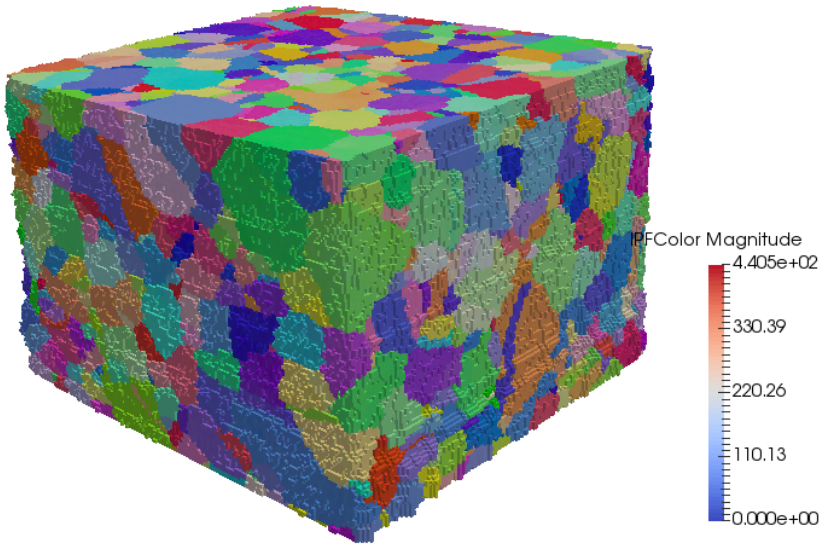

Figure 6: The fully-recreated microstructure of the IN100 data set as included with the Dream3D Package

Point estimates of growth hyperparameters were found using the estimation method of Section 3, and the top five are shown below in Table 1. Of the 999000 data sets, the best-fit match is an elliptical growth model with central orientation S given by Bunge-Euler angles $(\theta_1, \phi, \theta_2) = (1.122, 2.500, 0.762)$, orientation concentration parameter $\kappa_S = 0$, mean aspect rations $k_1 = 1.0$ and $k_2 = 1.0$, and aspect ratio concentration parameter given by $\kappa_v = 10$.

Previous research ([5]) has shown it reasonable to treat the IN100 data set as approximately equiaxed, and the estimation procedure reveals the same, as

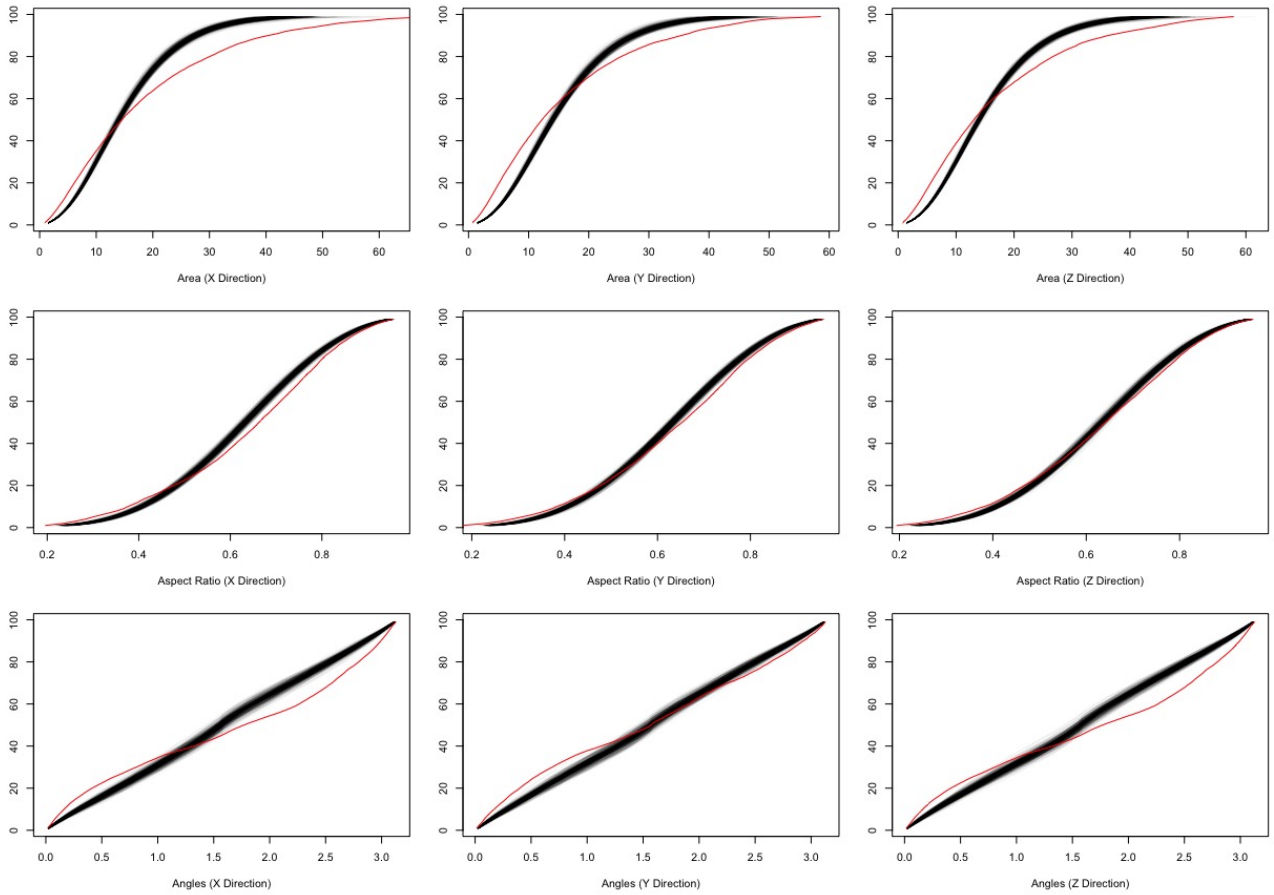

Figure 7: This image shows the area-weighted CDFs for the observed IN100 data set (Red) and the simulated data set (Black). A total of 500 simulated data sets was created and plotted in order to show the approximate range of variation in area-weighted CDF that would be expected from a single set of hyperparameters. Differences between the black bands and red curve indicate lack of fit in the ellipsoid growth statistical model. The number of grains used to generate each simulated curve was matched to that of the observed data set by resamplng.

the best-fit point estimates of the hyperparameters also produce an approximately equiaxed microstructure. The UARS method with concentration parameter $\kappa_S = 0$ indicates no preferential direction for morphological orientation, while the mean aspect ratios k_1 and k_2 both at 1.0 indicate no preferential axis for growth. The concentration parameter $\kappa_v = 10$ yielding a moderate concentration allows for grain shapes which are convex and not simply Voronoi tessellations in the cube - which is what a high concentration parameter κ_v would produce - but does not allow for occasional elongated grains, which is what a low κ_v would produce.

Figure 7 shows the mismatch between the area-weighted CDFs for the observed and simulated data sets. For the best-fit point estimates, a total of 500 simulated data sets was calculated, and the area weighted CDF plotted for each to show the approximate uncertainty in a single simulation, and to show where there exists model discrepancy - that is, aspects of real grains that elliptical growth is simply unable to reproduce - rather than simple randomness in the growth process.

	k_1	k_2	κ_v	κ_s	θ_1	ϕ	θ_2	Avg. Distance
1	1.000	1.000	10	0	1.122	2.500	0.762	0.075
2	0.871	0.936	10	5	6.078	1.193	0.157	0.076
3	0.807	0.807	10	1	3.012	1.710	0.144	0.077
4	0.871	1.000	10	5	2.937	1.193	0.157	0.077
5	0.936	1.000	10	0	0.205	1.949	0.157	0.079

Table 1: The top five sets of estimate parameter values, ranked by average distance metric (over the nine quantity/direction combinations) of the resulting simulated data set, for the IN100 microstructure.

The best fit shows an average area-weighted CDF difference of only 0.075, averaged over the nine possible combinations of direction and quantity of interest. This represents an average difference of 0.075 in between the red and black lines shown in Figure 6 above. Of course, some quantities (aspect ratios and angles) are reproduced more accurately than others (areas), and this is obscured by a simple average. Care should still be taken to observe the fit of each component in order to detect model deficiencies.

The area-weighted distributions of aspect ratios and angles show a good fit with only minor model discrepancy, while the area-weighted distributions of areas show a much more severe difference in shape. This can be partly attributed to the difference in total volume and grain density in the observed and simulated data sets, partly attributed to characteristics of observed data (such as edge effects) which have not been accounted for, and partly due to general model discrepancy. Unlike [13], no post-processing was performed on these simulated data

sets, and so there may still exist artifacts of the growth process, such as discontinuous grains. Fixing these may improve the match between observed and simulated area-weighted area CDFs; however, given that the estimation procedure is able to identify reasonable parameter estimates using the unprocessed data sets, it should not drastically affect the results of the estimation procedure, with the exception of the average distance metric itself should decrease. Furthermore, the area-weighted CDFs will also virtually ignore grains that might typically be removed in post-processing, such as those consisting of only one or a few voxels.

5 Spatially Varying Material Example

The flexibility of this method may also be used for grain generation in the presence of spatial variation, which is commonly seen in additively manufactured materials, as in Figure 1. A sample of Tantalum was procured by Los Alamos National Laboratory. This sample shows variation in the morphology, with the texture driven by $< 001 >$ crystallographic directions aligned with the TT of the plate, while allowing $< 001 >$ and $< 101 >$ directions to rotate during the processing based on local conditions. A large slice in the A direction from the material - not used for estimation - is shown in Figure 8. The A direction is orthogonal to TT.

The material also exhibits morphology trends, with grain size varying smoothly as a function of spatial location. The effect of this is that EBSD images taken at different locations may exhibit different grain sizes, which must be accounted for in the analysis. In fact, EBSD images used for estimation in this study were taken physically close to each other for a given sample direction, but physically far apart for different directions. This must be considered in any sort of estimation technique.

In addition, a small number of slices was available in only two orthogonal directions - 8 EBSD slices taken in the A-direction, and 6 EBSD slices taken in the TT direction. Determining growth hyperparameters for this sample requires adaptation of the method described in Section 3, both for the lack of information in direction B (orthogonal to both A and TT) and the spatially varying morphology. Fortunately, this is easy to accomplish within the estimation framework.

In order to accomplish estimation with data from only two orthogonal directions, assumptions must be made about the third orthogonal direction (or a large degree of uncertainty must be accepted) - [21], for example, estimates slices from two orthogonal directions by assuming that the distribution of ellipsoids is independent of position in the sample, and that there is no variation in orientation of the ellipsoids. While these assumptions are certainly valid for the materials studied within the paper, materials with spatial variation in the microstructure

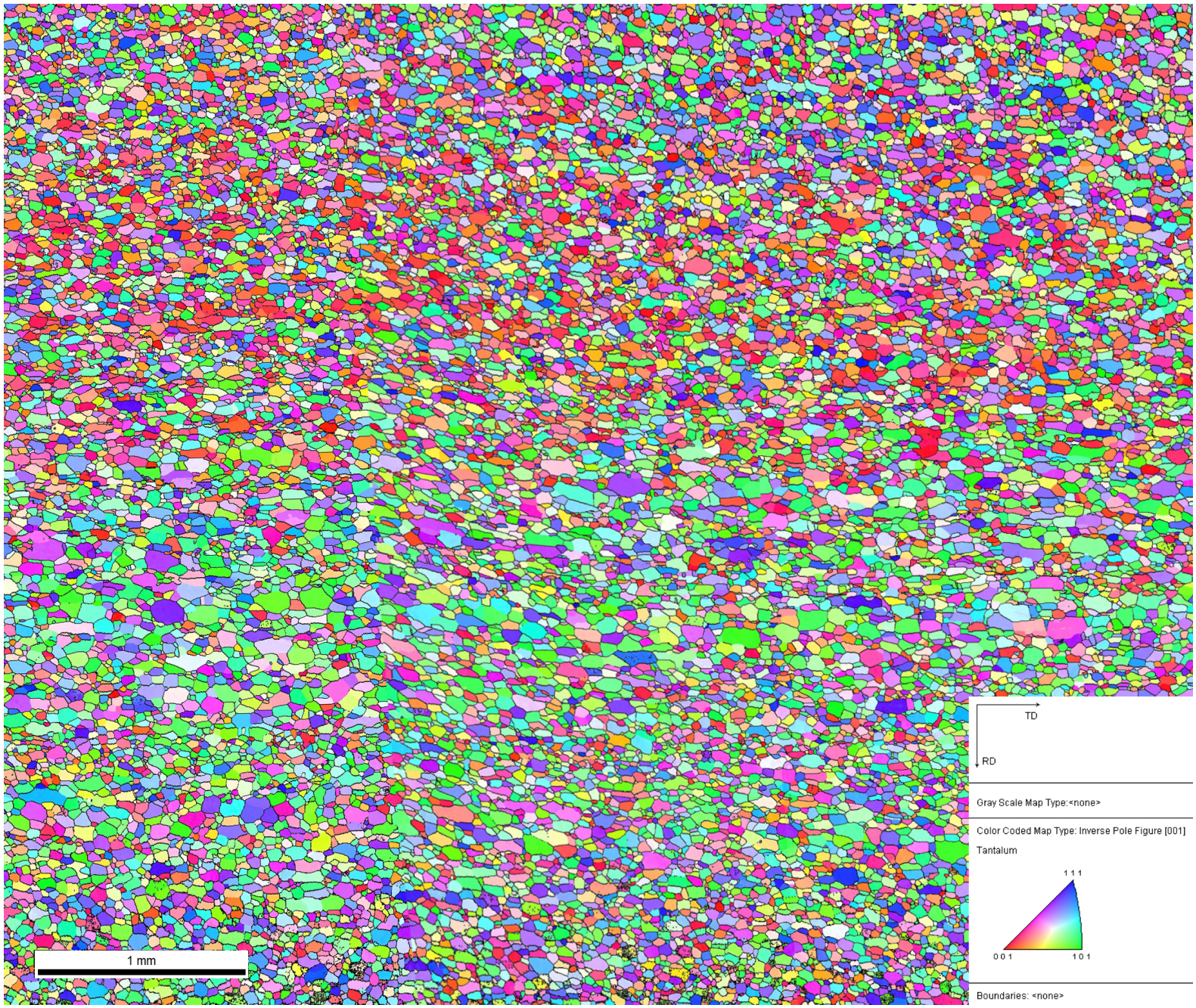

Figure 8: A large slice taken from the sample of Tantalum, showing spatial variation in the morphology of the grain size distribution caused by the particular manufacturing process.

violate one or both of the assumptions, and so alternate assumptions must be used. Based on knowledge of the material, the following analysis assumes that the distributions of grain characteristics in the missing (B) direction were the same as the distributions of grain characteristics in the A direction. To deal with the varying grain sizes due to different locations, it was decided to use two separate area scaling constants of the type given in equation 5 - one for both the A direction and B direction (treated by using the A direction distributions as if they were the B direction distributions), and one for the TT direction. The scaling constant for the A and B directions was the one that minimized, by way of a numerical minimization algorithm, the difference between the observed area weighted area CDFs and the simulated area weighted area CDFs for the A and B directions (identified as X and Y in the simulated data set), while the scaling constant for the TT direction minimized the difference between the area weighted area CDFs for the TT direction in the observed data set and the Z direction in the simulated data set.

	k_1	k_2	κ_v	κ_s	θ_1	ϕ	θ_2	Avg. Distance
1	1.000	0.614	10	10	5.353	0.200	0.797	0.079
2	0.936	0.614	100	10	1.910	2.88	0.000	0.088
3	1.000	0.614	100	10	5.353	0.200	0.797	0.089
4	1.000	0.614	10	5	5.353	0.200	0.797	0.090
5	1.000	0.550	100	5	5.353	0.200	0.797	0.091

Table 2: The top five sets of estimated parameter values, ranked by average distance metric (over the nine quantity/direction combinations) of the resulting simulated data set, for the Tantalum Starck microstructure.

The best-fit point estimates of the model were mean aspect ratios $k_1 = 1$ and $k_2 = 0.614$ (showing a slightly elongated preferential direction) with concentration parameter $\kappa_v = 10$ and Bunge-Euler angles $S = (5.353, 0.200, 0.797)$ with concentration $\kappa_S = 10$, as shown above in Table 2. Note that, as compared to the best-fit parameters chosen for the equiaxed example shown in Table 1, the estimated parameters (apart from the Bunge-Euler angles for the second best set of values) are all fairly close to each other, suggesting strong identifiability of growth parameters for models with elongated grains and a preferential growth direction.

Because of the sparsity of the original simulations, this point was used to search for a higher-resolution search of hyperparameters around those already identified. After considering other close rotations, aspect ratios, and concentrations, the best-fit estimates of the expanded (higher resolution) search were given by mean aspect ratios $k_1 = 0.9875$ and $k_2 = 0.618$ with concentration parame-

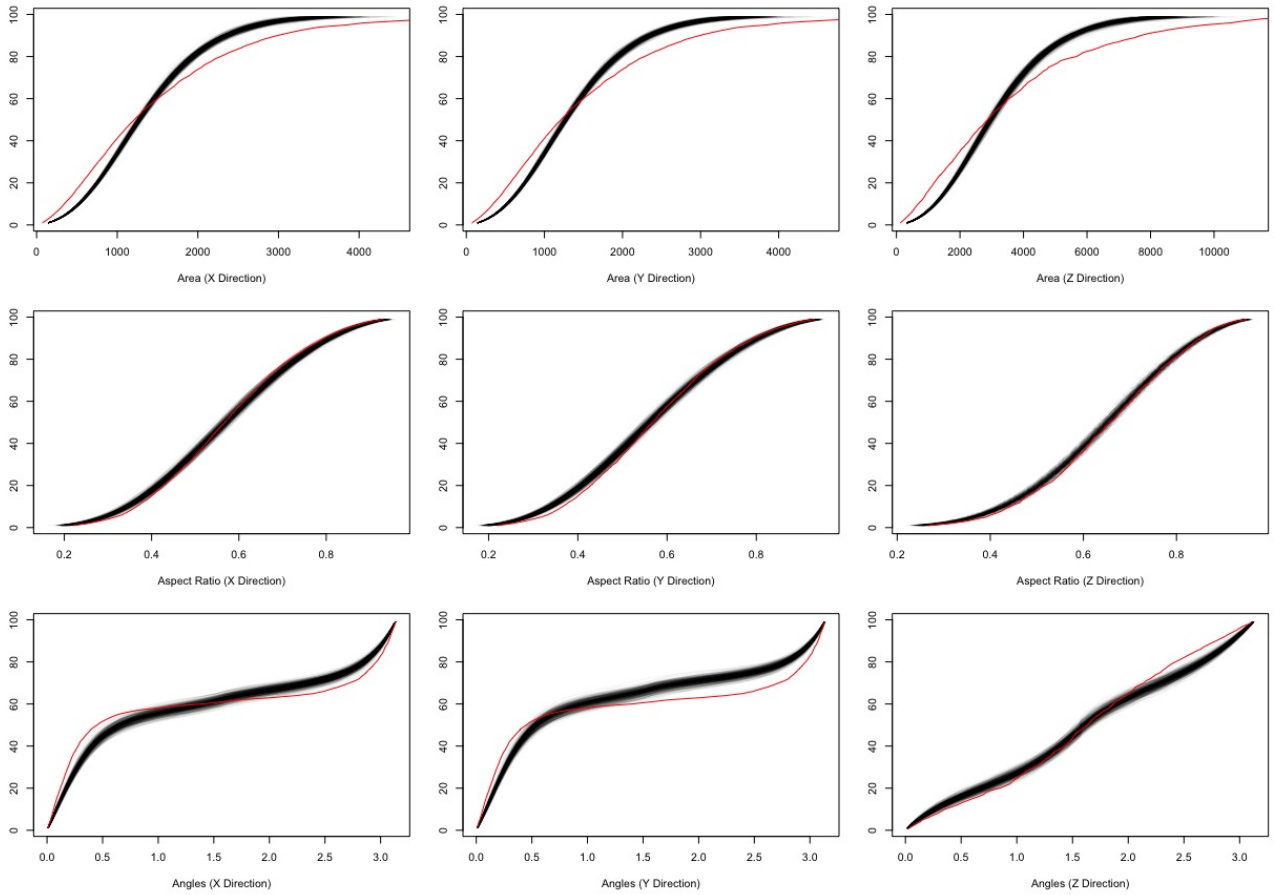

Figure 9: This image shows the area-weighted CDFs for the observed TaStarck data set (Red) and the best-fit simulated data set (Black). A total of 500 simulated data sets was created and plotted in order to show the approximate range of variation in area-weighted CDF that would be expected from a single set of hyperparameters. Differences between the black bands and red curve indicate lack of fit in the ellipsoid growth statistical model. The number of grains used to generate each simulated curve was matched to that of the observed data set by resamplng.

ter $\kappa_v = 20$ and Bunge-Euler angles $S = (5.353, 0.200, 0.797)$ with concentration $\kappa_S = 15$. This set of growth parameters gives near equal growth among the primary and secondary axes of the ellipsoids, but elongated growth along the tertiary axis, with a moderately high concentration of growth velocities near the ratio of the expectations so as to keep the elongation consistent. The rotation identified directs this growth along a mean axis that is approximately 11.54 degrees askew from the traditional z-axis in Cartesian coordinates a with a moderately high concentration of variation, matching with elongation along the TT of the plate as seen in Figure 8.

The best-fit area weighted CDFs for the observed and simulated data sets are shown in Figure 9. Once again, 500 data sets were simulated at the best-fit response in order to show the range of variation. Though aspect ratios are captured well, the observed area-weighted CDFs of angles and areas show discrepancy of the model. The area discrepancy can once again, however, be attributed to differing total volumes and grain densities and lack of post-processing, and it must also be kept in mind that the total number of slices available as data was relatively small, partially explaining the discrepancy in the aspect ratios. Of course, the elliptical growth model is only a rough geometric approximation, so it is not expected to entirely capture all aspects of the observed data.

Random microstructure realization could proceed by varying the hardcore Matérn process that governs the nucleation point generation. By creating a random field that smoothly transitions the density of nucleation sites according to an empirically fit intensity function, the varying grain sizes seen in the sample of Tantalum in Figure 8 can be generated. Estimation of such a field, however, would require EBSD images from a rich set of physical locations in the sample in order to understand the range of variation and rate of transition. For the data set considered, all the A slices were taken physically close to each other, and all the TT direction slices were taken physically close to each other, but the A location and TT location were physically separate in the material. This gives information only about two possible locations, which is not sufficient to estimate the properties of such a random field.

If further information were available, then it could be used to incorporate spatial properties of the material into the model. For example, the sample of material in Figure 8 shows a varying grain size distribution, with grains near the edges along one axis noticeably smaller and more numerous than grains in the center of the sample. A simple method that can be implemented to recreate this feature is to vary the hardcore Matérn process by allowing the nucleation site density to vary as a function of the nucleation site of the grain by manipulation of the minimum distance between nucleation sites. For example, letting x_g be the location of the nucleation site along the x axis in the [10 x 10 x 10] μm sized

simulation volume and defining the minimum distance between nucleation sites by the function

$$r(x_g) = \frac{1}{25} x_g (10 - x_g) \tag{6}$$

allows for nucleation site density to be larger near the borders of the simulation volume and smaller in the center, as in Figure 8. A slice taken from a simulated microstructure implementing this method is shown in Figure 10.

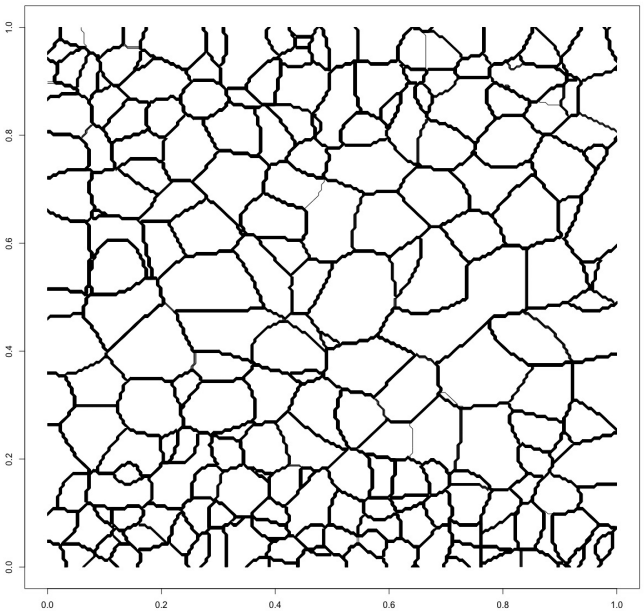

Figure 10: A slice from a simulated microstructure showing variation in the grain areas, similarly to Figure 8. For this microstructure, the nucleation site generation procedure was altered from the hardcore Matérn process to be more dense along the top and bottom of the sample of material and less dense in the center of the sample of the material.

A natural extension of this work is to estimate the underlying spatial variation in the hyperparameters controlling the growth process. One method is by estimation of a spatial intensity field for the hyperparameters, as in Equation (6). If this is spatial field is parametric, additional difficulties are introduced for estimation of the parameters. In general, the objective function would be evaluated

only approximately on simulated microstructures, which may be time-consuming to produce and show difficulties in convergence. It may also be possible to extend the area-weighted CDF of Equation (3) to detect or estimate the magnitude of spatial variation by local weighting over a spatial grid of locations.

6 Future Work

There are a few aspects which present immediate avenues for improvement: generation of a large number of simulated grains is not, in general, a practical plan for estimation, especially for those who lack the computational resources to do so. New methods for estimation can and should be developed for microstructures in the presence of spatial variation. This will in turn imply the development of new techniques for sampling materials in order to capture as much spatial variation as possible, as multiple slices taken at a single location - as in the data of Section 5 - will fail to give the information necessary for full estimation of any sort of spatial trends in the material as a whole. Furthermore, and as previously discussed, no post-processing was applied to the simulated microstructures in this data set. Standard post-processing is likely to increase the accuracy of the algorithm, with respect to producing microstructures that mirror observed ones. Lastly, expansion of the algorithm itself is possible. Ellipsoids were used for their well-known properties and historic use in geometric modeling of microstructures, but other shapes or adapatations are entirely plausible. Given the per-grain assignment of growth characteristics, it is possible to assign differing growth shapes or random characteristics drawn from different distributions assigned to individual grains or regions of a simulation cube. The hard-core Matérn process can also be generalized for spatially-varying grains by, for example, using an inhomogeneous Poisson process - either changing the density of nucleation sites within various locations, or by varying the minimum distance between nucleation sites. Lastly, the mechanical details of the additive manufacturing process can be used to guide the growth process - mimicking the radial cooling of the LENS process used to produce the sample in Figure 1, for example.

7 Conclusion

Current software and methods are insufficient to address micro-scale simulation of materials with spatial variation such as arises in new additive manufacturing methods. Even the simple spatial variation seen with new manufacturing methods of traditional materials such as in Figure 8 presents difficulties that standard software and techniques are unable to resolve. Allowing for per-grain control of

growth velocities and orientations drastically aids in this. Per-grain control will also certainly aid in generating complex digital microstructures such as the one shown in Figure 1 from additively manufactured materials.

The methods in this paper represent only a step towards a more flexible geometric simulation mechanism that can handle the radical spatial variation seen in such samples. In many respects, this paper addresses problems encountered for which no clear or common solution was available, such as estimation in the presence of spatial variation simultaneously with estimation using only orthogonal slices in only two directions. It is hoped that future work may expand upon these methods in order to flexibly simulate examples of additively generated microstructures.

Several new innovations are introduced that may find use outside of the geometric simulation application - the UARS method of [16] and [17], previously applied only to crystallographic rotations, also works well for simulation of rotations for grain orientations. This is an improvement over the more empirical distributions used in [6]. Furthermore, the area-weighted CDF of Section 3 is introduced for comparison between two data sets. The area-weighted CDF gives emphasis to grains in a way that may be more natural to think about to material scientists, and may be more fruitful for general analysis.

The future of material science will involve new manufacturing methods which create non-traditional microstructures, and so too will new tools be required for analysis. This paper represents one step towards creating such tools.

References

[1] A. Brahme, David M. Saylor, Joseph M. Fridy, and Anthony D. Rollett. Statistically representative three-dimensional microstructures for modeling microstructural evolution in aluminum. 2003.

[2] A. D. Rollett, D. Saylor, J. Fridy, B. S. El-Dasher, A. Brahme, S.-B. Lee, C. Cornwell, and R. Noack. Modeling Polycrystalline Microstructures in 3D. In S. Ghosh, J. C. Castro, and J. K. Lee, editors, *American Institute of Physics Conference Series*, volume 712 of *American Institute of Physics Conference Series*, pages 71–77, June 2004.

[3] A. Brahme, M.H. Alvi, D. Saylor, J. Fridy, and A.D. Rollett. 3d reconstruction of microstructure in a commercial purity aluminum. *Scripta Materialia*, 55(1):75 – 80, 2006. Viewpoint set no. 41 "3D Characterization and Analysis of Materials" Organized by G. Spanos.

[4] Michael Groeber, Somnath Ghosh, Michael D. Uchic, and Dennis M. Dimiduk. Developing a robust 3-d characterization-representation framework for modeling polycrystalline materials. *JOM*, 59(9):32–36, Sep 2007.

[5] Michael Groeber, Somnath Ghosh, Michael D. Uchich, and Dennis M. Dimiduk. A framework for automated analysis and simulation of 3d polycrystalline microstructures, part 1: Statistical characterization. *Acta Materia*, 56:1257–1273, 2008.

[6] Michael Groeber, Somnath Ghosh, Michael D. Uchich, and Dennis M. Dimiduk. A framework for automated analysis and simulation of 3d polycrystalline microstructures, part 2: Synthetic structure generation. *Acta Materia*, 56:1274–1287, 2008.

[7] Stephen D. Sintay, Michael A. Groeber, and Anthony D. Rollett. *3D Reconstruction of Digital Microstructures*, pages 139–153. Springer US, Boston, MA, 2009.

[8] S D Sintay and A D Rollett. Testing the accuracy of microstructure reconstruction in three dimensions using phantoms. *Modelling and Simulation in Materials Science and Engineering*, 20(7):075005, 2012.

[9] Michael A. Groeber and Michael A. Jackson. Dream.3d: A digital representation environment for the analysis of microstructure in 3d. *Integrating Materials and Manufacturing Innovation*, 3(5), 2014.

[10] Sudipto Mandal, Jacky Lao, Sean Donegan, and Anthony D. Rollett. Generation of statistically representative synthetic three-dimensional microstructures. *Scripta Materialia*, 146:128 – 132, 2018.

[11] Ramin Bostanabad, Yichi Zhang, Xiaolin Li, Tucker Kearney, L. Catherin Brinson, Daniel W. Apley, Wing Kan Liu, and Wei Chen. Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques. *Progress in Materials Science*, 95:1 – 41, 2018.

[12] Stephen R. Niezgoda, Yuksel C. Yabansu, and Surya R. Kalidindi. Understanding and visualizing microstructure and microstructure variance as a stochastic process. *Acta Materialia*, 59(16):6387 – 6400, 2011.

[13] Kirubel Teferra and Lori Graham-Brady. Tessellation growth models for polycrystalline microstructures. *Computational Materials Science*, 102:57–67, 2015.

[14] G. S. Rohrer, J. Li, S. Lee, A. D. Rollett, M. Groeber, and M. D. Uchic. Deriving grain boundary character distributions and relative grain boundary energies from three-dimensional ebsd data. *Materials Science and Technology*, 26(6):661–669, 2010.

[15] J. Teichmann, F. Ballani, and K.G. van den Boogaart. Generalizations of matérn's hard-core point processes. *Spatial Statistics*, 3:33 – 53, 2013.

[16] Melissa A. Bingham, Daniel J. Nordman, and Stephen B. Vardeman. Modeling and inference for measured crystal orientations and a tractable class of symmetric distributions for rotations in three dimensions. *Journal of the American Statistical Association*, 104(488):1385–1397, Dec 2009.

[17] Melissa A. Bingham, Daniel J. Nordman, and Stephen B. Vardeman. Bayes inference for a tractable new class of non-symmetric distributions for 3-dimensional rotations. *Journal of Agricultural, Biological, and Environmental Statistics*, 17(4):527–543, 2012.

[18] Joseph C. Tucker, Lisa H. Chan, Gregory S. Rohrer, Michael A. Groeber, and Anthony D. Rollett. Tail departure of log-normal grain size distributions in synthetic three-dimensional microstructures. *Metallurgical and Materials Transactions A*, 43A:2810–2822, 2012.

[19] S.P. Donegan, J.C. Tucker, A.D. Rollett, K. Barmak, and M. Groeber. Extreme value analysis of tail departure from log-normality in experimental and simulated grain size distributions. *Acta Materialia*, 61(15):5595 – 5604, 2013.

[20] M.A. Groeber, B.K. Haley, M.D. Uchic, D.M. Dimiduk, and S. Ghosh. 3d reconstruction and characterization of polycrystalline microstructures using a fib–sem system. *Materials Characterization*, 57(4):259 – 273, 2006.

[21] David M. Saylor, Joseph Fridy, Bassem S. El-Dasher, Kee-Young Jung, and Anthony D. Rollett. Statistically representative three-dimensional microstructures based on orthogonal observation sections. *Metallurgical and Materials Transactions A*, 35A:1969–1979, 2004.

[22] Anna Yershova, Jain Swati, Steven LaValle, and Jlie Mitchell. Generating uniform incremental grids on so(3) using the hopf fibration. *The International Journal of Robotics Research*, 2009.

[23] Mahesh Shenoy, Yustianto Tjiptowidjojo, and David McDowell. Microstructure-sensitive modeling of polycrystalline in100. *International Journal of Plasticity*, 24(10):1694 – 1730, 2008. Special Issue in Honor of Jean-Louis Chaboche.

A Pseudocode for Elliptical Growth Algorithm

EGM = function(x, y, z, v_1, v_2, v_3, 0, ℓ, dt, $maxes$) {

Input Quantities: Assume N total grains. The input quantities should be of the following form:

x, y, and z are $1 \times N$ vectors consisting of the (x, y, z) coordinates of the nucleation sites. There is one coordinate for each grain.

Likewise, v_1, v_2, and v_3 are $1 \times N$ vectors consisting of the growth velocities along the primary, secondary, and tertiary ellipse axes for a grain.

O is a $3 \times 3 \times N$ array or matrix, such that each 3×3 element corresponds to the rotation matrix for each grain.

ℓ is the sidelength of the voxels in in the simulation cube. The total volume of each voxel is ℓ^3.

dt is the time step for growth. Setting larger values makes grains grow quicker at each time iteration, resulting in more collisions between growing ellipsoids (meaning more computation necessary). If dt is too small, grains may stop growing early because they don't encounter any new voxels between iterations. Setting dt = 0.1 works well for growth velocities with means in the approximately 1 - 10 range.

$maxes$ is a vector of three numbers consisting of the maximum length, width, and height of the simulation growth cube. These were assumed to be cleanly divisible by ℓ, but if they aren't rounding will be necessary.

Calculated Quantities: These are the quantities that should be created for use in the elliptical growth algorithm:

$pts.x$, $pts.y$, and $pts.z$ should be the number of voxels along a given edge in each direction. For a given direction i from 1 to 3, this is $maxes[i]/\ell$. For example, a side of length 10 with $\ell = 0.1$ would have 100 voxels along it, and a side of length 12 with $\ell = 0.25$ would have 48 voxels along it.

$grains$ is an N element list, array, or some other type of object which is created and contains information about each grain. Some of the information is dupli-

cated, but collecting it in one location is useful. The following were used in code referenced in the paper:

- x, y, and z, the nucleation site of the grain
- v_1, v_2, and v_3, the growth velocities along the three major axes of the grain.
- O, the rotation matrix of the grain axes.
- r_{min} and r_{max}, the current growth radii of the ellipsoids. These will grow by dt at each iteration, with $r_{min} = 0$ and $r_{max} = dt$ set for initial values.
- $x.index, y.index$, and $z.index$ are the indices of the voxel where the nucleation site of the grain is located. These are calculated by taking the ceiling of each of $x/\ell, y/\ell$, and z/ℓ, respectively.
- $vox.count$ is the number of voxels assigned to a grain at the current step. This can be used to easily calculate the volume.
- $iterations.since.new.voxels$ is a counter that counts the number of iterations since new voxels have been added. This can be used to control how quickly the algorithm discards grains if they appear to have stopped growing.

If such a list, array, or some other type of object is unavailable, each of these may be used in their original form or simply created apart from the rest. Given an individual grain identifier, a call to *grains* should provide all relevant information about the grain with said identifier.

$voxdata.grains$ should be a $pts.x \times pts.y \times pts.z \times k$ array or matrix, with k zeros at each point. This array is used to store the grains that are competing for assignment at each voxel. For the algorithm as used in the paper, $k = 50$ was chosen somewhat arbitrarily as a large number, but if there are more than 50 grains competing for a voxel the computation will either crash abruptly or be forced to constantly lengthen the vector, which causes dramatic slowdown. A solution is to increase k, or decrease dt to slow the growth down. This array will be necessarily large and if there are issues of excessive memory usage, this variable is likely to be the culprit.

$voxdata.grains.length$ should be a $pts.x \times pts.y \times pts.z$ array or matrix that keeps track of the number of grains competing for a certain voxel - this is the number of nonzero elements of the point in voxdata.grains.

vox is a $pts.x \times pts.y \times pts.z$ array or matrix that will contain the final assignment of each voxel to a grain.

grains.considered is a vector of integers from 1 to N which serves as grain identifiers. This vector will be shortened and have grains removed from consideration when they stop growing.

Algorithm: The actual growth of grains, using the preceding inputs and calculated quantities, should proceed as follows. All new variables should be instantiated anew each iteration of the algorithm:

while(Stopping criterion not met) {

new.voxels is a $3 \times k$ matrix or array, where k is some large number. This will contain the x, y, and z indices of the new voxels to be assigned to grains in a current iteration.

new.voxels.count is a counter variable of the total number of new voxels to be assigned in this iteration. It should be set to 0 each time through the loop.

for(grains currently in consideration, given by *grains.considered*) {

ellipsoid.points = getIndicesWithinRotatedEllipsoid(current grain)

The function getIndicesWithinRotatedEllipsoid() is described at the end of this appendix. The function takes data from the current grain still in consideration - found in the corresponding element of *grains* - and the output *ellipsoids.points* is a 3-row matrix or array of indices of voxels - ranging from 1 to the corresponding *pts.x*, *pts.y*, or *pts.z* - for the current grain such that the "radius" of the rotated ellipsoid is between r_{min}^2 and r_{max}^2.

for(voxels with indices given by *ellipsoid.points*) {

if(The number of voxels with indices in *ellipsoids.points* > 0 AND the current voxel has not already been assigned to a grain) {

voxdata.grains.length[current voxel] is incremented by 1 to note that at least one new grain is claiming the current voxel this iteration.

voxdata.grains[current voxel] is appended with the current grain in consideration in the first nonzero element. The in-

teger given by *voxdata.grains.length*[current voxel] should be the first nonzero element.

new.voxel.count is incremented by one and used to assign the current voxel indices to a row of *new.voxels*.

}

}

For the *grains* object corresponding to the current grain in consideration, increase the r_{min} and r_{max} by dt. Also increase *iterations.since.new.voxels* by 1 for the *grains* object. This will be reset to 0 in the next iteration if the grain is assigned to a voxel.

}

to.assign is a 3-row matrix or array created by taking the unique rows of *new.voxels*, since it will almost certainly contain repeats, to obtain the multiple grains competing for the same voxel.

if(*to.assign* is not empty) {

for(voxels with indices given by rows of *to.assign*) {

if(*voxdata.grains.length*[voxel given by row of *to.assign*] == 1) {

If this is the case, there is only one grain competing for a given voxel. Assign *vox*[voxel given by row of *to.assign*] to the one grain in consideration, which can be found in the first element of *voxdata.grains*[voxel given by row of *to.assign*].

For the *grains* object corresponding to the one grain being assigned to a voxel, increase the *vox.count* element by one and set *iterations.since.new.voxels* equal to zero.

}

if(*voxdata.grains.length*[voxel given by row of *to.assign*] > 1) {

If this is the case, there are multiple grains competing for a given voxel. The integer *voxdata.grains.length*[voxel given by row of *to.assign*] will give the number of grains in competition, and *voxdata.grains*[voxel given by row of *to.assign*] gives the index numbers of the grains in competition, which can be used to call corresponding *grains* objects.

Using the center of the voxel as the target, the distance metric given in equation (1) is calculated for each grain. The grain that has the smallest distance is assigned to the voxel in *vox*.

For the *grains* object corresponding to the grain being assigned to the voxel, increase the *vox.count* element by one and set *iterations.since.new.voxels* equal to zero.

}

}

}

to.remove is created as a vector of length N that may be boolean or integer, but will contain two values - one to indicate that a grain should be removed after the current iteration, and one to indicate that a grain should not be removed.

for(*grains* objects corresponding to grains in *grains.considered*) {

If *iterations.since.new.voxels* is 0 for the current *grains* object, set the *to.remove* element corresponding to grains equal to the "remove" value.

In practice, it works well to also check that some minimum growth criterion has been met - that a certain number of iterations has passed, or that *vox.count* for the *grains* object is larger than 0, indicating that the grain has been assigned to at least one voxel.

}

Create a new *grains.considered* vector consisting of only the elements of the

current *grains.considered* object which do not have the "remove" value in the *to.remove* vector.

}

The stopping criterion is not extremely important, as for later iterations there should be very few grains still in consideration, and so the iterations should be very quick to execute. Several options that were used at various times were a simple large number, checking if all voxels had been assigned to grains, or that the minimum of all growth velocities had, starting from the centroid of the simulation cube, grown large enough to have reached the furthest edge of the simulation cube. Some combination would likely be ideal.

The key to the EGM() function is the balance between growing grains and removing them from consideration upon cessation of growth. In early iterations, grain assignment should be quick due to the small ellipsoid size and the lack of collisions between grains competing for the same voxel. In later iterations, grain assignment should be quick due to many grains having been removed from consideration due to halted growth. Running time for the algorithm should roughly follow a pyramid pattern with respect to iteration - fast early iterations, longer intermediate iterations, and fast late iterations.

In practice, there were very rarely a small number of voxels that, even after the stopping criterion, remained unassigned to grains. For these, the distance metric of equation (1) was calculated over all N grains. Since the number of voxels was typically very small (typically less than 0.1% of all voxels), this only took a few seconds to compute.

The *vox* element should be returned. If desired, the volumes of grains should be quick to compute at this point as well.

}

getIndicesWithinRotatedEllipsoid = function(x, y, z, O, v_1, v_2, v_3, r_{min}, r_{max}, ℓ, $pts.x$, $pts.y$, $pts.z$) {

Input quantities: Each of these should correspond to a single grain, except for the voxel characteristics given by $\ell, pts.x, pts.y$, and $pts.z$.

The real numbers x, y, and z are the nucleation site of the grain.

The real numbers v_1, v_2, and v_3 are the growth velocities along the axis of the grain.

The 3×3 matrix O is the rotation matrix for the axes of the grain.

The real numbers r_{min} and r_{max} are the current growth radii of the ellipsoid.

ℓ is the side length of each voxel, while $pts.x$, $pts.y$, and $pts.z$ are the number of voxels along a given edge in each direction.

Calculated quantities: These are calculated from the input quantities to help find indices of voxels, ranging from 1 to $pts.x$, $pts.y$, or $pts.z$, depending on the direction that the grain has grown into at a given iteration.

E is created as a 3×3 diagonal matrix with entry $E[d, d]$ for $d = 1, 2, 3$ given by $1/v_d^2$.

$\lambda = (O^T E O)$ gives the velocities in the rotated directions, and $\lambda^{-1} = (O^T E O)^{-1}$ gives constants that will be used to create a box that will contain the rotated ellipsoid.

$x.index, y.index$, and $z.index$ are the indices of the voxel where the nucleation site of the grain is located. These are calculated by taking the ceiling of each of $x/\ell, y/\ell$, and z/ℓ, respectively.

$cube.indices.x$ is created as a vector of integers ranging from $\max(1, \text{floor}(x.index - \sqrt{\lambda^{-1}[1,1]} * r_{max}/\ell))$ to $\min(\text{ceiling}(x.index + \sqrt{\lambda^{-1}[1,1]} * r_{max}/\ell), pts.x)$. Similarly $cube.indices.y$ and $cube.indices.z$ are created with the same formula but using $y.index$ with $\lambda^{-1}[2,2]$ and $pts.y$, and $z.index$ with $\lambda^{-1}[3,3]$ and $pts.z$, respectively.

For reference, define $length.x$ as the length of $cube.indices.x$, and similarly $length.y$ and $length.z$ as the length of $cube.indices.y$ and $cube.indices.z$.

$cube.indices$ is created as a $3 \times l.x * l.y * l.z$ array or matrix such that each column corresponds to a point (x, y, z) in three-dimensional space such that plotting all of them should show a (discrete) cube with a filled interior such that all possible $cube.indices.x$, $cube.indices.y$, and $cube.indices.z$ combination is represented.

The rows can be individually created by

1. Repeating each individual *cube.indices.x* element a total of $l.y * l.z$ times.
2. Repeating each individual *cube.indices.y* element a total of $l.z$ times, and then repeating that entire vector a total of $l.x$ times.
3. Repeating the entire *cube.indices.z* vector a total of $l.x * l.y$ times.

cube.centroid is created as a the (x,y,z) coordinates of the centers of the voxels given by *cube.indices*, and is calculated by taking each element of *cube.indices*, subtracting 1, multiplying the result by ℓ, and then adding $\ell/2$.

Algorithm: This section calculates indices of voxels the rotated ellipsoid has reached given r_{min} and r_{max}.

Given the previous sections, gathering the points of the rotated ellipsoid is rather simple - it is simply applying a vectorized version of equation (1). In particular, by considering the formula in terms of $\lambda = (O^T EO)$, operations can be algebraically applied to entire rows of *cube.centroid*, drastically speeding up computing time.

After the calculation, a vector *distances* is created for each column of *cube.centroid*, for the distance (using the growth metric of Equation 1 from the nucleation site to the centroid of the voxel. Then, using some computationally efficient method, find only the elements of *distances* that are between r_{min} and r_{max}. These will give a set of voxels that form a hollow ellipsoid consisting of the voxels with "radii" between r_{min} and r_{max}.

The algorithm should return the rows of *cube.indices* that have *distances* between r_{min} and r_{max}.

}