Integrating Materials and Manufacturing Innovation (2022) 11:109–120
https://doi.org/10.1007/s40192-022-00251-w

TECHNICAL ARTICLE

XtalMesh Toolkit: High-Fidelity Mesh Generation of Polycrystals

Jonathan M. Hestroffer[1] · Irene J. Beyerlein[1,2]

Received: 17 November 2021 / Accepted: 5 January 2022 / Published online: 1 February 2022

Abstract
A method for generating high-fidelity, boundary conforming tetrahedral mesh of three-dimensional (3D) polycrystalline microstructures is presented. With growing interest into subgrain scale micromechanics of materials, crystal plasticity finite element (CPFE) models must adapt not only their respective constitutive laws, but also their model geometry to the finer scale, namely the representation of grains and grain boundary junctions. Additionally, with the increasing availability of microstructure datasets obtained via 3D tomography experiments, it is possible to characterize the 3D topology of grains. From these advancements in experiment emerge both an opportunity and challenge for researchers to develop model microstructures, specifically finite element meshes, which best preserve grain topology for the accurate representation of boundary conditions in polycrystalline materials. To accomplish this, an open-source code called XtalMesh was created and is presented here. XtalMesh works by smoothing input voxel microstructure data using a feature-aware Laplacian smoothing algorithm that preserves complex grain topology and leverages state-of-the-art tetrahedralization code fTetWild to generate volume mesh. In this work, the workflow and associated algorithms of XtalMesh are described in detail using a synthetically generated example microstructure. For demonstration, we present a case study involving mesh generation of an experimentally obtained microstructure of nickel-based superalloy Inconel 718 (IN718).

Keywords Microstructure · Meshing · Polycrystals · Crystal plasticity finite element

Introduction

Apart from the crystallographic orientation of individual grains, the morphology of grains and grain boundary interfaces are also known to contribute to the development of localized stress and strain in polycrystals. In polycrystalline nickel-based superalloys, for example, the morphological characteristics of twin boundary planes, including their twin plane area and spatial orientation with respect to the loading direction, are critical factors in determining the location of strain localization and crack initiation during monotonic and cyclic loading [1–3]. Additionally, incipient plasticity in these materials, such as the initiation of slip bands, has been found to correlate strongly with grain morphology. Recent experimental work on the nickel-based super alloy, Inconel 718 (IN718), shows that a large fraction of slip bands visible at the specimen surface are found to emanate from subsurface grain boundary triple junctions, the lines of intersection formed by three grains [4]. Therefore, when modeling the micromechanics of such materials, high-fidelity representations of microstructures, those which preserve grain topology, are desirable for accurate calculation of spatially resolved fields near or at microstructural heterogeneities. One mesoscale modeling technique known as crystal plasticity finite element method (CPFEM), routinely used to calculate the subgranular full-field spatial mappings of stress and strain in polycrystalline materials, requires discretization of the desired microstructure in the form of finite element mesh to approximate the complex mechanical boundary conditions present within polycrystals. The process of generating a mesh for a given microstructure is computationally intensive and non-trivial, and for the purposes of calculating spatially resolved fields at the intragranular scale, it is critical to develop automated mesh generation

✉ Jonathan M. Hestroffer
jonathanhestroffer@ucsb.edu

Irene J. Beyerlein
beyerlein@ucsb.edu

1 Materials Department, University of California, Santa Barbara, CA, USA

2 Department of Mechanical Engineering, University of California, Santa Barbara, Santa Barbara, CA, USA

workflows that can transform synthetically or experimentally obtained microstructure data into high-fidelity mesh.

Presently, there exist a few commercial software that can accomplish this task, but no open-source codes exist for high-fidelity microstructure meshing. Commercial codes can pose some disadvantages. They typically require purchasing or procuring a product license of the product and in some cases, the intended research needs be aligned with or fall into an applicable category. Also, some of the algorithms cannot be shared because they contain proprietary information. In the case of high-fidelity microstructure meshing, where features such as grain boundaries and triple junction lines are preserved, there are a limited number of commercial codes available. Notable examples include CUBIT [5] and Simmetrix [6]. Both are full-featured software tool kits for computer-aided design (CAD) model preparation and finite element mesh generation, and each has a module dedicated for mesh generation given three-dimensional (3D) voxel microstructure data. Other commercial codes which do not have dedicated modules for processing microstructures data have been leveraged by researchers to mesh microstructures. Work by Lee and co-authors [7] demonstrates how such a workflow operates. Researchers first prepared a feature-preserving surface mesh of a grain boundary network using their own in-house smoothing algorithms [8] and then generated volume mesh using the commercial software HyperMesh [9] with their surface mesh as input.

Options for open-source codes are much more limited, and most that exist are not well suited for high-fidelity microstructure meshing. Neper [10], one of the most widely used and mature open-source software packages for polycrystal generation and meshing, is built around the use of Laguerre or Voronoi tessellation techniques to create simplified convex-cell representations of grains. While Neper supports 3D microstructure image data as input, and certain parameters of the tessellation process can be modified to promote feature preservation, Neper is by-design best suited for applications that permit substantial simplification of the microstructure. While tessellation methods maybe be appropriate for some alloys, others, like nickel-based superalloys, have complex grain morphologies that can be severely altered if represented as simplified convex cells. Other lesser known open-source codes, like MicroStructPy [11], apply similar tessellation approaches to microstructure meshing and do not prioritize subgranular feature preservation. To the authors' knowledge, the only open-source code currently available that preserves microstructural features and generates volume mesh is Voxel2Tet [12]; however, its use is limited to one article where only its voxel smoothing algorithm was utilized, and a separate tetrahedralization algorithm was adopted to produce a valid mesh for an experimentally obtained microstructure of IN718 [4].

Here, we present XtalMesh (read: crystal mesh), an easy-to-use, containerized program for high-fidelity meshing of microstructures, and is publicly available on GitHub [13]. Given 3D microstructure data, XtalMesh generates unstructured, boundary-conforming tetrahedral mesh that preserves the underlying grain topology. XtalMesh is built upon a suite of geometry processing libraries and tools leveraged to create its meshing workflow and its design allows for customization and fast prototyping of other workflows if desired. In this article, we detail the workflow and algorithms of XtalMesh and provide information on processing time and memory usage for different input and parameter configurations. We also present a case study where a mesh is generated for an experimentally obtained 3D microstructure of nickel-based superalloy IN718. There we assess microstructure fidelity, mesh quality metrics, and perform a CPFE simulation to verify analysis-readiness of the mesh and investigate subgrain scale micromechanics.

Setup and Installation

XtalMesh was developed to run in a ”containerized” environment where all of its application code, libraries, and dependencies are encapsulated in a light-weight single executable package of software. One of the major advantages of containerization is that it allows applications to become stand-alone and portable, able to run on any host operating system (OS) or even cloud platforms. For typical users, all that is required to setup XtalMesh is to install any OS-level virtualization software platform, such as Docker [14], then download the publicly available container image. Once downloaded, XtalMesh can be run as a container through any command line interface. Installation instructions and all source code can be found on GitHub [13].

Workflow

The general workflow of XtalMesh can be seen in Fig. 1.

Pre-Processing

Before XtalMesh can be run, the necessary input files must be generated. There are a total of four input files needed, together they describe the complete surface geometry of all grains in the input voxel microstructure in the form of triangle mesh. These files contain information about the geometry of the model including the spatial coordinates of nodes and triangle-node connectivity. They also contain microstructural information, such as node types, describing whether a node is located in the interior or exterior of the representative volume element (RVE) and what type

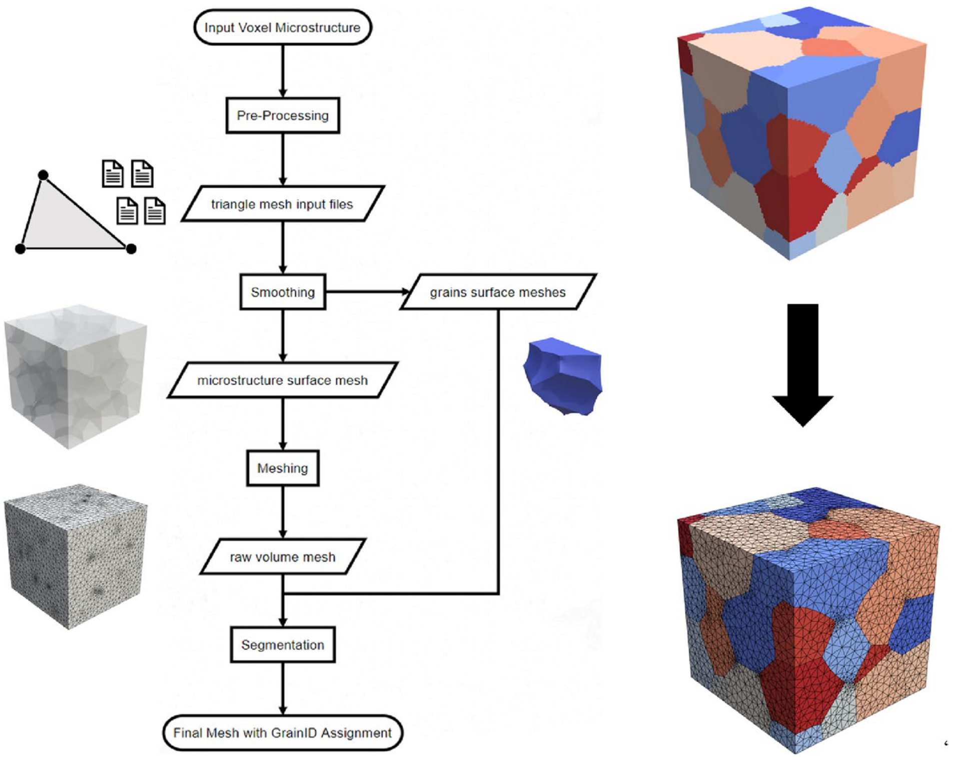
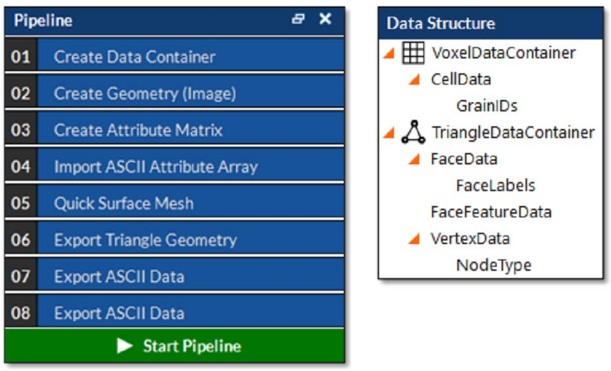

Input Voxel Microstructure
Pre-Processing
triangle mesh input files
Smoothing
grains surface meshes
microstructure surface mesh
Meshing
raw volume mesh
Segmentation
Final Mesh with GrainID Assignment

Fig. 1 Flowchart of XtalMesh



Pipeline
01 Create Data Container
02 Create Geometry (Image)
03 Create Attribute Matrix
04 Import ASCII Attribute Array
05 Quick Surface Mesh
06 Export Triangle Geometry
07 Export ASCII Data
08 Export ASCII Data
Start Pipeline

Data Structure
VoxelDataContainer
CellData
GrainIDs
TriangleDataContainer
FaceData
FaceLabels
FaceFeatureData
VertexData
NodeType

Fig. 2 Sample DREAM.3D pipeline including relevant filters to create surface mesh for a voxel microstructure and export information of nodes, node types, triangles, and triangle face labels [15]

of feature it belongs to such as a grain boundary, a triple junction line or a quadruple point. Lastly, for every triangle facet of the mesh, representing a 2D element of a grain boundary, there is a record of what grains share the boundary. These files can be created from scratch by the user; however, they can be generated more efficiently using DREAM.3D [15], a sample pipeline can be seen in Fig. 2. There, the user defines the input voxel geometry, imports the microstructure as a text file of grain IDs, and applies the 'Quick Surface Mesh' filter to generate the needed triangle data.

Smoothing

With the surface geometry of the input microstructure fully defined, the smoothing process of XtalMesh can proceed. A mesh can be represented as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with vertices $\mathbf{V}$ and edges $\mathbf{E}$, where $\mathbf{V} = \left[\mathbf{v}_1^T, \mathbf{v}_2^T, \ldots, \mathbf{v}_n^T\right]^T$, $\mathbf{v}_i \in \mathbb{R}^3$. The classic Laplacian smoothing algorithm describes the displacement of a vertex $\mathbf{v}_i$ to a new position $\mathbf{v}_i'$ using the following formula

$$\mathbf{v}_i' = \mathbf{v}_i + \lambda \boldsymbol{\delta}_i \tag{1}$$

where $0 \leq \lambda \leq 1$ is a scale factor, and $\boldsymbol{\delta}_i$ represents the Laplacian of $\mathbf{v}_i$, calculated as the difference between $\mathbf{v}_i$ and the weighted centroid of its first-order neighborhood vertices $\{i,j\} \in \mathbf{E}$

$$\boldsymbol{\delta}_i = \sum_{\{i,j\}\in\mathbf{E}} w_{ij}\left(\mathbf{v}_j - \mathbf{v}_i\right) \tag{2}$$

where $\sum_{\{i,j\}\in\mathbf{E}} w_{ij} = 1$, and the choice of weights is given by

$$w_{ij} = \frac{\omega_{ij}}{\sum_{\{i,k\}\in\mathbf{E}} \omega_{ik}} \tag{3}$$

Computing the displacement update for all vertices in the mesh requires Laplacians $\boldsymbol{\Delta} = \left[\boldsymbol{\delta}_1^T, \boldsymbol{\delta}_2^T, \ldots, \boldsymbol{\delta}_n^T\right]^T$ given by

$$\boldsymbol{\Delta} = \mathbf{L}\mathbf{V} \tag{4}$$

where $\mathbf{L}$ is the $n \times n$ Laplacian matrix defined as,

$$\mathbf{L}_{ij} = \begin{cases} -1 & i = j \\ w_{ij} & (i,j) \in \mathbf{E} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

, and the general update equation for the entire mesh becomes

$$\mathbf{V}' = \mathbf{V} + \lambda\boldsymbol{\Delta} \tag{6}$$

For the voxel surface mesh of the input microstructure, a feature-aware constrained Laplacian smoothing algorithm is developed where certain vertices can be free to move, $\mathbf{V}_{free}$, OR fixed, $\mathbf{V}_{fixed}$, weights can be non-uniform, and an additional geometric constraint matrix $\mathbf{C} = \left[\mathbf{c}_1^T, \mathbf{c}_2^T, \dots \mathbf{c}_n^T\right]^T$ is applied. $\mathbf{C}$ restricts the displacement of vertices along the three orthogonal directions based on their location with respect to the RVE and serves to retain the original cubic shape of the RVE. Values of $\mathbf{c}_i$ are determined by the following,

$$\mathbf{c}_i = \begin{cases} [1,1,1] & \text{interior (no constraint)} \\ [0,1,1] & \text{x-face} \\ [1,0,1] & \text{y-face} \\ [1,1,0] & \text{z-face} \\ [0,0,1] & \text{xy-edge} \\ [1,0,0] & \text{yz-edge} \\ [0,1,0] & \text{xz-edge} \\ [0,0,0] & \text{corner (fully constrained)} \end{cases} \tag{7}$$

$\mathbf{C}$ is multiplied with $\boldsymbol{\Delta}$ through an element-wise product, and the new update equation becomes,

$$\mathbf{V}' = \begin{cases} \mathbf{V} + \lambda(\mathbf{C} \circ \boldsymbol{\Delta}) & \mathbf{V} \in \mathbf{V}_{free} \\ \mathbf{V} & \mathbf{V} \in \mathbf{V}_{fixed} \end{cases} \tag{8}$$

In this workflow, RVEs are smoothed in three steps, wherein each step defines a particular set of $\mathbf{V}_{free}$ and $\mathbf{V}_{fixed}$ as well as weights, ω_{ij}. Figure 3 shows the order of smoothing operations and their effects for an example microstructure, which is synthetically generated. The first step involves smoothing the exterior triple lines, which represent the intersection of grain boundaries and the RVE exterior surface. Vertices belonging to exterior triple lines and their intersections, identified as exterior quadruple points, are free to move, while all other vertices of the mesh are fixed. Non-uniform weights are applied according to edge types as follows,

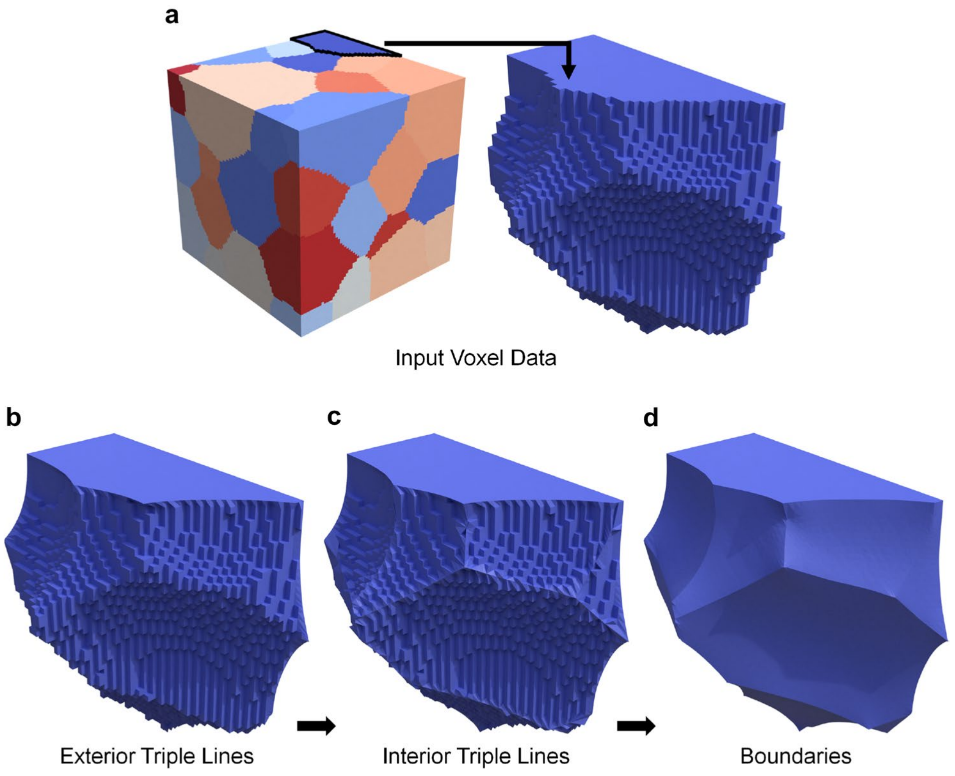


Fig. 3 Diagram of the smoothing process. **a** Input voxel microstructure for a synthetically generated example microstructure, showing the voxelated form of a select grain. **b** Select grain after smoothing exterior grain boundary triple junction lines, **c** interior triple lines, **d** and subsequent smoothing of boundaries

$$\omega_{ij} = \begin{cases} 1 & (i,j) \in \mathbf{E}_{triple-triple} \\ 2 & (i,j) \in \mathbf{E}_{triple-quad} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

A larger weight is applied to edges connecting triple-line and quadruple-point vertices so as to enhance the smoothing of triple-line vertices local to quadruple points. This is done because quadruple-point vertices do not undergo significant displacement during smoothing. Therefore, with an otherwise uniform weighting scheme, triple-line vertices close to quadruple points would smooth more slowly than those located in the middle of the triple line. After smoothing the exterior triple lines, the interior triple lines and their intersections are smoothed in a similar manner with the same edge-weight scheme defined above. The only difference is that vertices belonging to interior triple lines and quadruple points are free while all others are fixed. Lastly, vertices belonging to grain boundaries at both the exterior and interior of the RVE are smoothed with uniform weights, $\omega_{ij} = 1$, while all other vertices are fixed.

The surface mesh smoothing algorithm implemented works solely to provide smooth representations of grains in a computationally efficient manner and is not intended to produce optimized surface mesh. Therefore self-intersections and non-manifold geometry may exist in the final output. However, these occurrences are minor and inconsequential to later stages of meshing. Once smoothing has been completed, surface mesh for every individual grain as well as the entire microstructure is written to the working directory, ready for subsequent volume meshing.

Meshing

The final step of the XtalMesh workflow is volume meshing. Having prepared a smoothed representation of the microstructure, it is now crucial that a valid, analysis-ready mesh can be generated, while preserving the underlying grain topology. In addition, the meshing algorithm selected must be capable of processing any input as XtalMesh makes no assumptions about the user's material and/or microstructure. Here, we have implemented the highly robust tetrahedral meshing method known as fTetWild [16] and have adapted it to mesh polycrystals. Traditional tetrahedral meshing algorithms, like Delaunay-based methods, widely used in commercial software for their efficiency, make strong assumptions on the input surface mesh, requiring the input to be watertight, manifold, and free of self-intersections. In reality, many 3D models have these defects and require manual cleanup of these artifacts before these methods can create valid tetrahedral mesh. The publicly available Thingi10k dataset provides a good representation of the models found *in the wild*, comprised of 10,000 non-sanitized 3D printing models created by over 1,000 users [17]. Researchers tested the robustness of several state-of-the-art meshing codes using the Thingi10k dataset and have found that their success rates in producing a valid mesh can vary greatly [18]. As an example, TetGen, the most commonly used Delaunay-based meshing code, was only able to successfully mesh 50% of the models found in the dataset. In contrast, fTetWild, our algorithm of choice, exhibits a 100% success rate thanks to its envelope-based meshing algorithm [16]. More detail on the algorithm can be found in the original paper for fTetWild, as well as its predecessor TetWild [16, 18].

fTetWild was specifically designed for robust automatic 3D meshing pipelines, making no assumptions on the input surface mesh, and produces a valid tetrahedral mesh for any perfect or imperfect input of arbitrary complexity. It is for this robustness that fTetWild algorithm works first by simplifying the input triangle mesh, where vertices might be merged or edges might be collapsed, while ensuring triangles stays within an envelope distance, ϵ of the original input. After this operation, a background tetrahedral mesh is generated around the new input vertices, and triangles of the input are inserted into the background mesh, forming a new tetrahedron based on their intersections with the background mesh. As triangle insertion proceeds, local mesh operations are taking place to improve mesh quality. These processes continue until all input surface triangles have been inserted, and all tetrahedral elements are above a minimum quality threshold. fTetWild determines the quality of tetrahedral elements according to their 3D conformal energy [19], a quantity with possible values ranging from three to infinity, with a regular tetrahedron having a conformal energy of three. The meshing algorithm stops mesh optimization when all elements are below a threshold energy (default: 10), or the number of iterations reaches a maximum (default: 80).

The meshing process of XtalMesh is shown in Fig. 4. XtalMesh proceeds by passing to fTetWild the recently smoothed surface mesh of the whole microstructure. The fTetWild algorithm then runs and outputs a volume mesh that is processed further by XtalMesh. Because fTetWild was not designed with the intent of multi-material meshing, it is the job of XtalMesh to segment the mesh according to the respective grains of the input microstructure. This additional processing step consists of assigning grain IDs to the elements of the mesh based on which grain surface mesh that they are located within. This grain segmentation step utilizes a generalized winding number algorithm for efficient computation of inside/outside condition [20].

XtalMesh allows users to control certain meshing parameters, treated as direct inputs into the fTetWild algorithm. These user-defined parameters include the target edge length, l, which determines the average element size, and envelope size, ϵ, which represents the maximum deviation

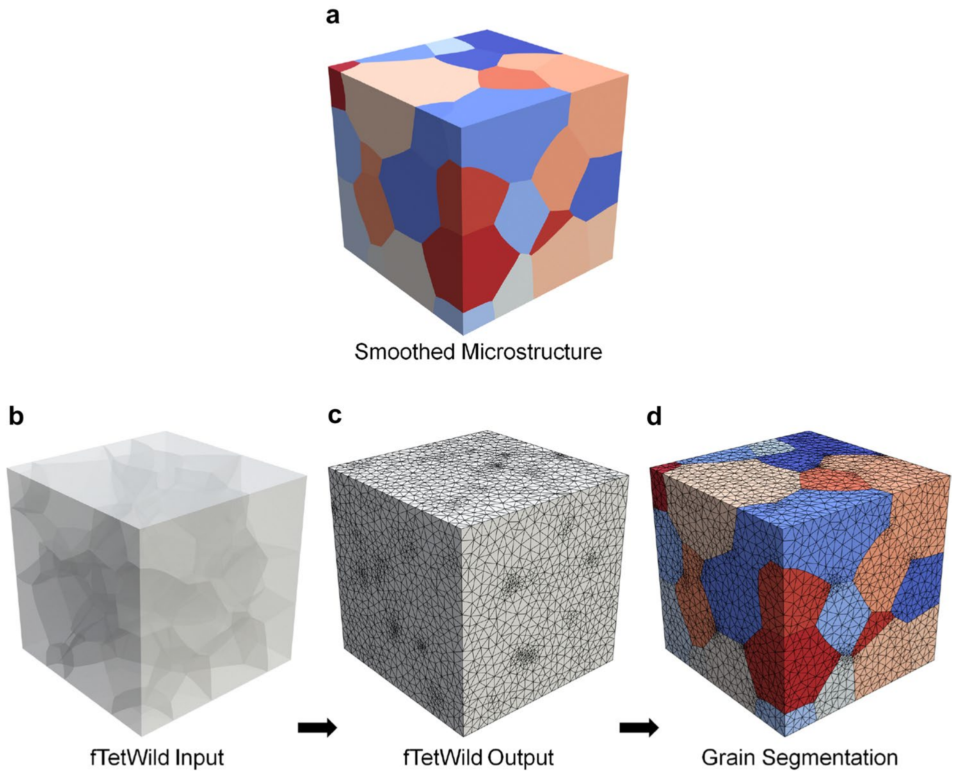


Fig. 4 Diagram of the meshing process. **a** Smoothed input surface microstructure, **b** triangle geometry of all grain boundary surfaces used as input into the fTetWild algorithm. **c** Volume mesh output of fTetWild, which are the elements produced within grain surface meshes. Yet, the algorithm is still unaware of grain ID assignment. **d** Final mesh after segmenting elements according to grain IDs

allowed from the input surface mesh. Each parameter is expressed as a fractional length of the body diagonal of the input model b. With b, a value for l of 0.05, results in an average element edge length of $\frac{b}{20}$, assuming the input geometry permits such a coarse mesh without sacrificing mesh quality. Likewise, for the other parameter, a value for ϵ of 0.001 will restrict fTetWild's approximation of the surface mesh (think re-positioning vertices) to within a distance of $\frac{b}{1000}$ from the original input surface. This holds true regardless of the input. To demonstrate, Fig. 5 presents the individual effects of l and ϵ on the mesh for the previously smoothed microstructure in Fig. 3. For relatively large values of l and ϵ, a coarse mesh is created as shown in Fig. 5a. In the case that a more accurate representation of the input grain shape is desired but for a similar mesh resolution, smaller values of ϵ can be used. Doing so will have the effect of refining the mesh in regions of the grain unable to be represented with the defined element size, see Fig. 5b. To produce a mesh that is both finer in element size and

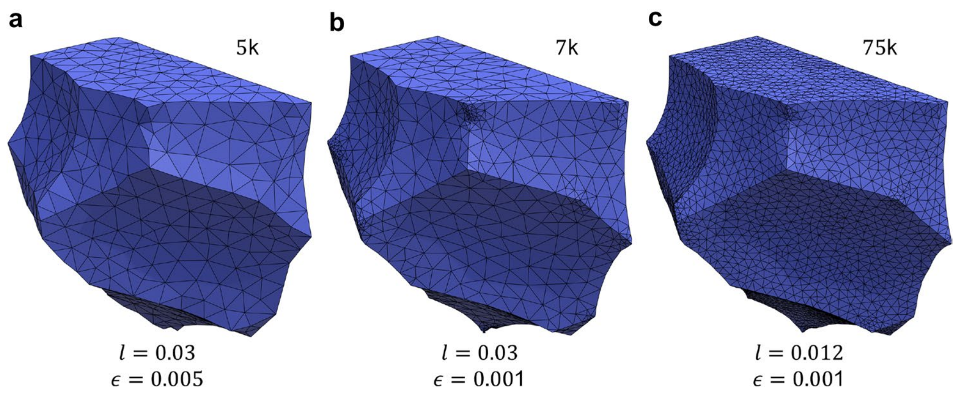


Fig. 5 Visualization of meshing parameter effects for one grain within a synthetic microstructure meshed using three different choices of l and ϵ. The resulting meshes are characterized according to their average element size and approximation of input surface microstructure as (**a**) coarse with low precision, **b** coarse with high precision, and (**c**) fine with high precision. Total number of elements for the grain is shown in the upper right corner of each sub-figure

Table 1 Processing time and peak RAM usage for XtalMesh smoothing and meshing operations for synthetic microstructures of increasing size, expressed by their cube length in voxels.

Input Size		Smoothing (50 iterations)		Meshing ($l = 0.05, \epsilon = 0.001$)	
voxels^3	# grains	time (s)	RAM (GB)	time (s)	RAM (GB)
100^3	91	50	0.8	107	0.9
150^3	267	147	1.8	342	2.3
200^3	587	350	3.8	787	4.4
250^3	1166	678	7.0	1733	9.4
300^3	1929	1231	12.0	2964	18.0

The generated microstructures also have an average grain diameter of 27 voxels. All cases were run on an Intel® Xeon® W-2145 processor with 14 cores

provides a better approximation of the input surface mesh, then relatively small values for both l and ϵ can be chosen, as shown in Fig. 5c.

Computational Resources and Performance

The processing time of XtalMesh and the amount of random-access memory (RAM) it uses depend largely on the input data and choice of meshing parameters. To help users develop a effective mesh generation process within the capabilities of their system, we analyze the effects that each factor has on total processing time and peak RAM usage for both the smoothing and meshing steps of XtalMesh. The results, presented in Table 1, indicate that for most applications, both computation time and peak memory usage strongly depend on the size of the input data. Understanding RAM usage is of particular importance, since available system memory will ultimately limit the achievable mesh resolutions and even input model sizes.

While the smoothing operation's use of RAM depends solely on input data size, the meshing operation usage depends also on the target edge length l. A separate analysis was conducted in which three synthetically generated microstructures of different size and number of grains were meshed for a range of l values. The results first show that for a given model, an intrinsic peak memory usage can be associated with the total triangle facet geometry that must be processed, and that it increases with larger input size, this is evident in the plateaus of memory usage for large target edge lengths, see Fig. 6a. Not until a target edge length value of 1×10^{-2} or smaller does memory usage noticeably change for the tested models. As shown in Fig. 6b as well, the peak RAM usage clearly does not increase beyond that of the intrinsic input data memory until higher mesh resolutions, greater than 1 million elements for the cases tested. Based on the results of this analysis, it is recommended to use the minimum allowable input size for the desired application if either computation time or memory usage are of concern is recommended. Lastly, both the smoothing and meshing operations of XtalMesh benefit from parallel processing; therefore, it is also recommended to use all available CPU cores to minimize execution time.

Results and Discussion

Case Study - Tribeam Data

XtalMesh is not limited to processing synthetically generated microstructures and is equally capable of generating

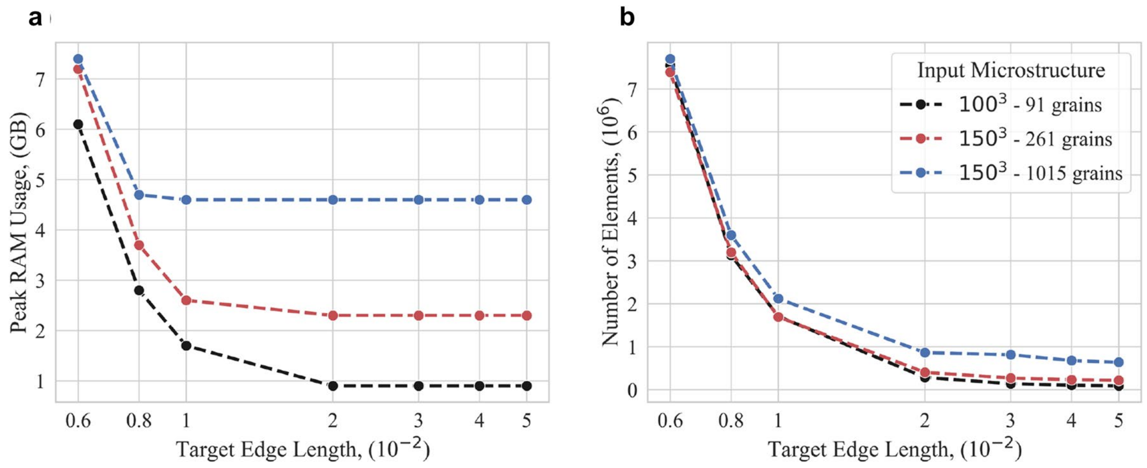


Fig. 6 **a** Peak RAM usage during volume meshing, and (**b**) the number of elements generated for the synthetic model microstructures meshed with varying target edge length, l. The three models differ in size and number of grains

analysis-ready mesh for experimentally obtained data. To demonstrate this, a case study is carried out involving a 3D microstructure of nickel-based superalloy IN718 collected by Tribeam serial sectioning [21], seen in Fig. 7a. This microstructural dataset was obtained at a voxel resolution of 1μm, and the subset chosen for this analysis contains 313 grains. Due to inherent error introduced during the reconstruction of the 3D dataset and alignment of electron backscatter diffraction (EBSD) slices, all faces of the dataset are flat except for those corresponding to the free surface of the specimen. To process non-flat surface features with XtalMesh, the free surface is filled with voxels until a flat surface is achieved as shown in Fig. 12 in Appendix. This free surface "cap" of voxels is treated as a separate grain during the smoothing process and is removed before meshing. The final mesh produced is shown in Fig. 7b and contains 2,058,526 elements.

One of the two main functions of XtalMesh is to produce a high-fidelity mesh representation of the input microstructure. Visual comparison of the input data and generated mesh confirms this result for the IN718 microstructure where practically all grain topology is preserved, including high aspect ratio twins found throughout the microstructure. The high level of fidelity achieved by XtalMesh is made more apparent with the grain-level comparison found in Fig. 7c, where the raw voxel representation of a selected grain is overlaid onto its generated mesh. From various viewing angles, it is apparent that the divergence of the generated grain mesh from its original input is largely confined to within a few voxel-widths. Fidelity of the mesh was also

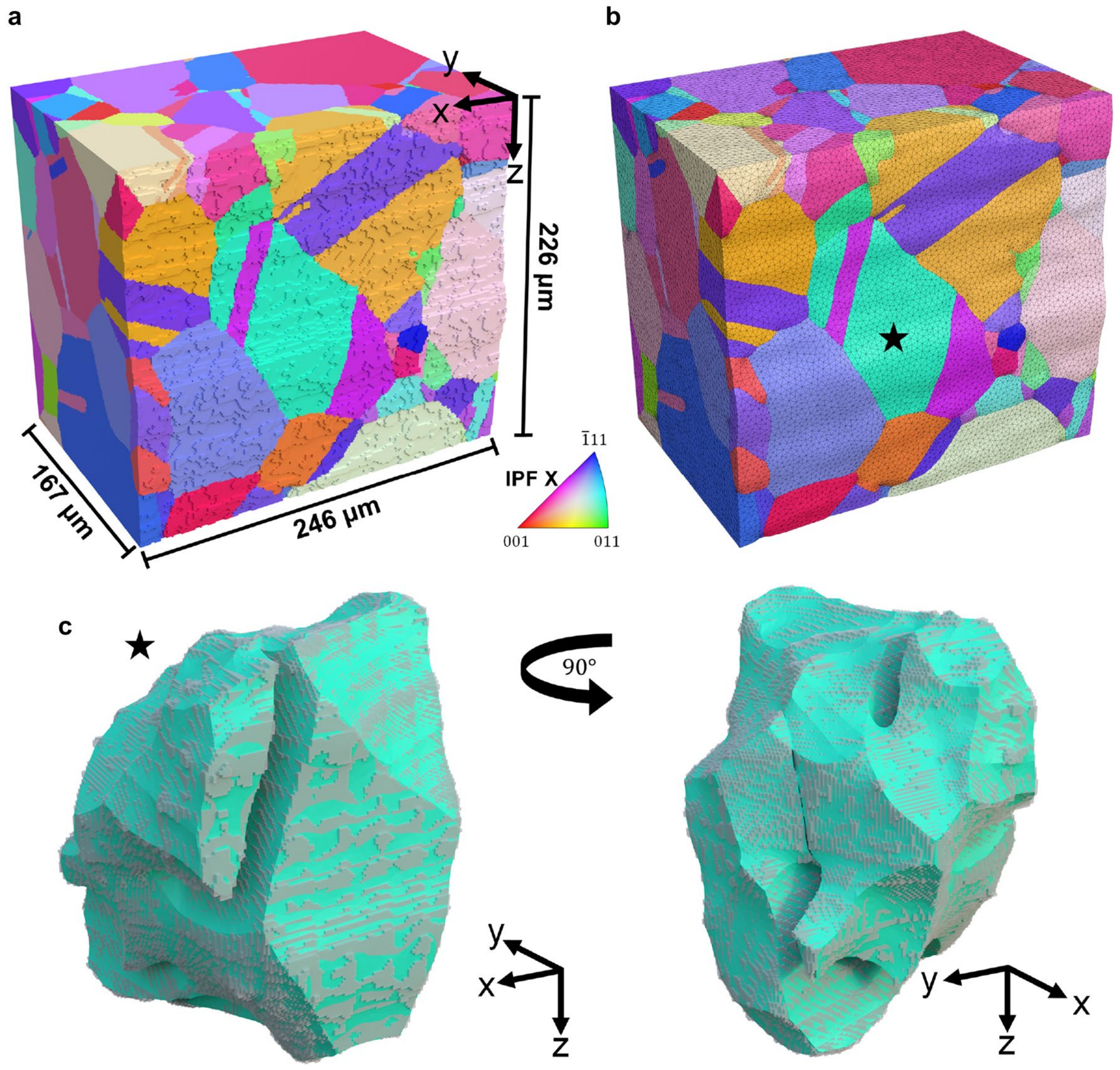


Fig. 7 **a** 3D reconstruction of Tribeam experiment dataset for IN718, and (**b**) mesh generated with XtalMesh with a star indicating the grain selected for further visual comparison. **c** Selected grain from the IN718 mesh (blue) with original voxel grain geometry overlaid (transparent gray). Two viewing angles are provided. Dataset obtained from Stinville et al. [21]

analyzed quantitatively, computed as the percentage of mesh elements found inside of their original grain voxel meshes. An element was determined to be inside or outside based on the location of its centroid with respect to its grain voxel mesh. From this calculation, it is determined that 98.3% of mesh elements generated are located inside their original grain volumes, this considerable overlap supports the method's high-fidelity classification.

Apart from outputting an accurate representation of the input microstructure, another main function of XtalMesh is to produce valid, analysis-ready meshes. A mesh is considered *valid* if all elements have positive volume and *analysis-ready* if the elements are of sufficiently high quality for numerical finite element simulations. To illustrate the validity and quality of the generated mesh, histograms of three different tetrahedral element quality metrics are provided in Fig. 8. The metrics chosen were the scaled Jacobian, shape, and minimum dihedral angle, as defined in The Verdict Library Reference Manual [22].

To further validate the mesh, we perform a full-field crystal plasticity finite element (CPFE) simulation of tensile straining in the x-direction to a strain of 0.8%. Details of the constitutive law, and chosen material parameters, can be found in [4]. Figure 9a compares the calculated stress-strain curve with the experimental one. The calculated stress at 0.8% strain corresponds roughly to the measured 0.2% offset yield stress. With the simulation results from this mesh, different CPFE variables can be analyzed, such as the von Mises stress field in Fig. 9b. The visualizations here are accomplished using the open-source software ParaView (https://www.paraview.org/) [23]. While such macroscopic analyses and verification of grain-average response can be executed using much simpler microstructure representations like those generated by Neper, high-fidelity microstructure representations enable more precise verification and analysis methods and allow for subgrain scale correlations to be drawn from model results. For example, Fig. 10 shows a slip activity map measured from high-resolution digital image correlation (HR-DIC) for a region within the deformed IN718 microstructure in Fig. 7c, when viewing in the y-direction. The primary slip activity involves slip systems C3:$\left(\overline{1}\overline{1}1\right)[101]$ and C5:$\left(\overline{1}\overline{1}1\right)[\overline{1}10]$. In agreement, the model not only indicates that these systems are active in the cross-section seen experimentally, but also throughout the grain. Further, it suggests that their activity is heterogeneously distributed with systems C5 and C3 activated at the front- and back-facing regions of the grain, respectively, as can be seen in Fig. 11. We attribute the agreement on the local slip activity distribution to the high-fidelity microstructure representation.

Conclusions

In this work, we present XtalMesh, an open-source code used to generate analysis-ready meshes of polycrystals emphasizing high-fidelity microstructure representation. Given voxel microstructure data, XtalMesh smooths grain boundaries and triple junction lines and produces valid, boundary conforming tetrahedral mesh that preserves the underlying grain topology. This code is suitable for both synthetically generated and experimentally measured microstructures, even those with high aspect ratio lamellae, such as twins. Its containerized format allows for portability and simple setup on any host operating system

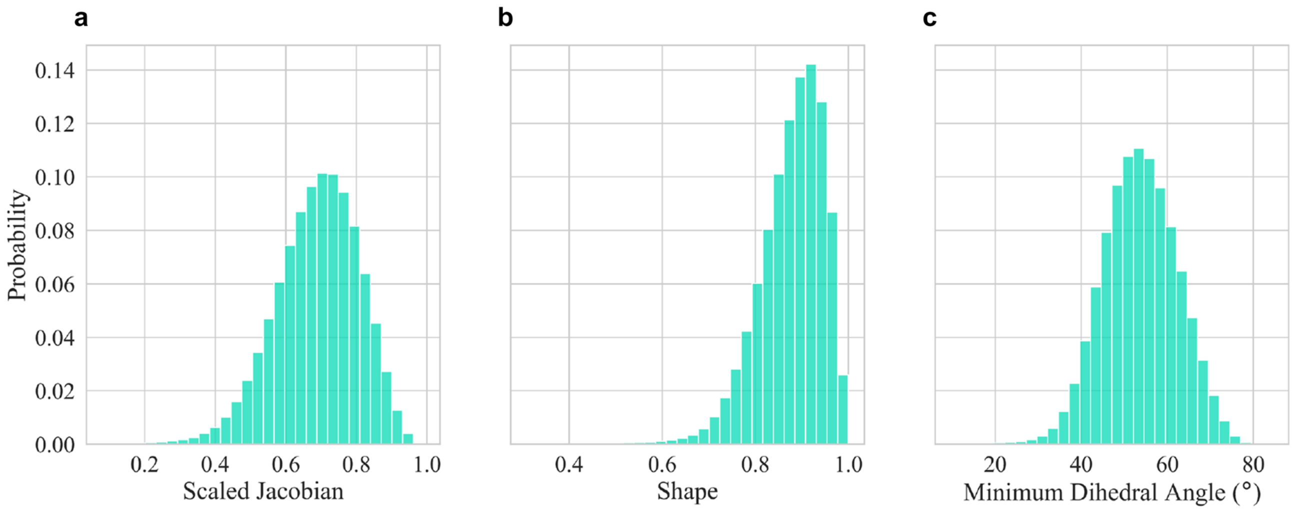


Fig. 8 Tetrahedron mesh quality statistics for the generated IN718 mesh. Metrics include (**a**) scaled Jacobian, **b** shape, and (**c**) minimum dihedral angle as defined by the verdict library reference manual [22]

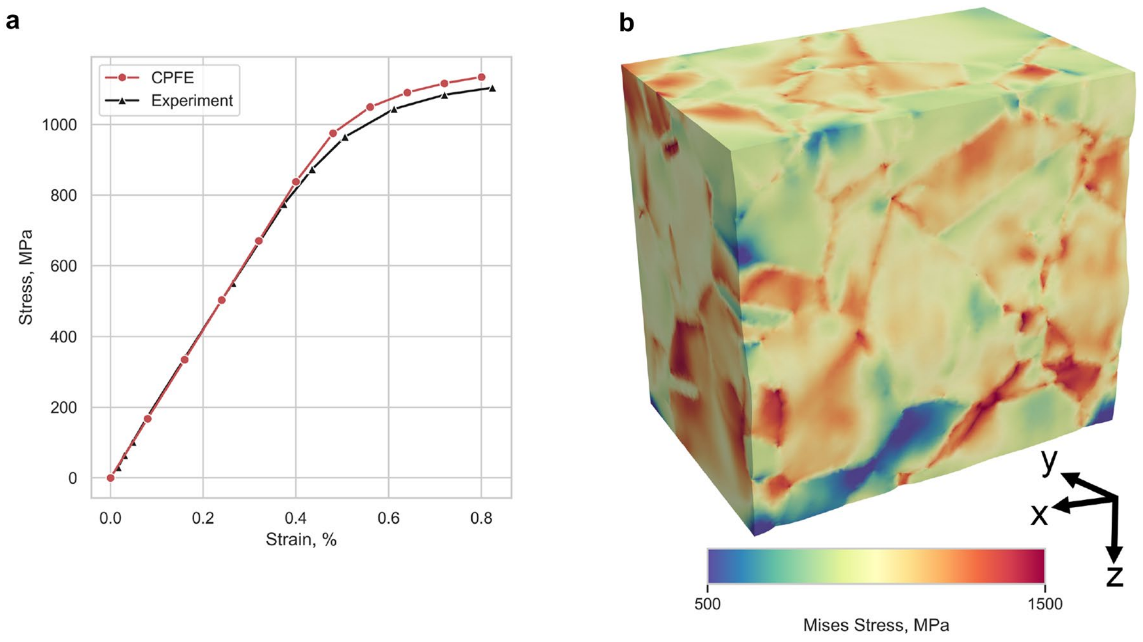


Fig. 9 **a** Engineering stress-strain curves for the IN718 material measured by experiment and simulated by CPFE with the XtalMesh model and (**b**) CPFE calculated von Mises stress field. Experimental data obtained from Stinville et al. [24]

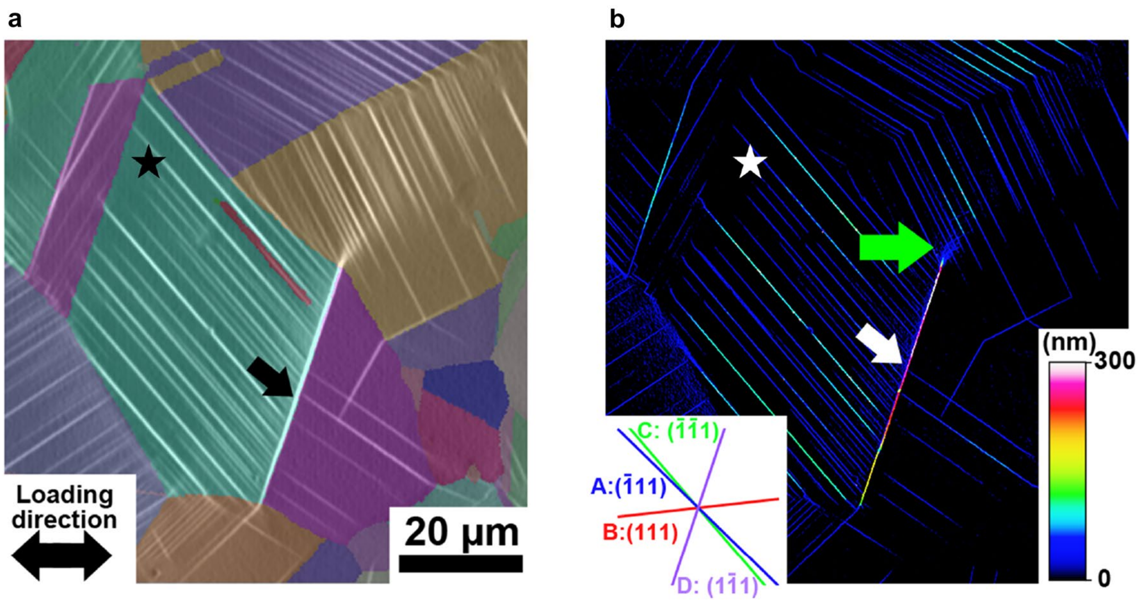


Fig. 10 HR-DIC strain field overlaid on an EBSD map (**a**), and in-plane slip displacement map (**b**) for a region of IN718 microstructure observed after uniaxial tension to 1.83% strain. The star in each subfigure denotes the grain of interest found in Fig. 7c viewed in the y-direction. Data obtained from Stinville et al. [24]

or cloud platform. The base workflow of XtalMesh and its algorithms are described in detail, and a case study is presented involving mesh generation for a microstructure of the nickel-based superalloy IN718 collected by Tribeam serial sectioning. There, the high-fidelity classification of the mesh is confirmed qualitatively through visual comparisons with the input microstructure and quantitatively on a per-element basis. Finally, a crystal plasticity finite element simulation is performed to confirm analysis-readiness of the mesh where macroscopic and subgrain scale results of the model are validated by experiment.

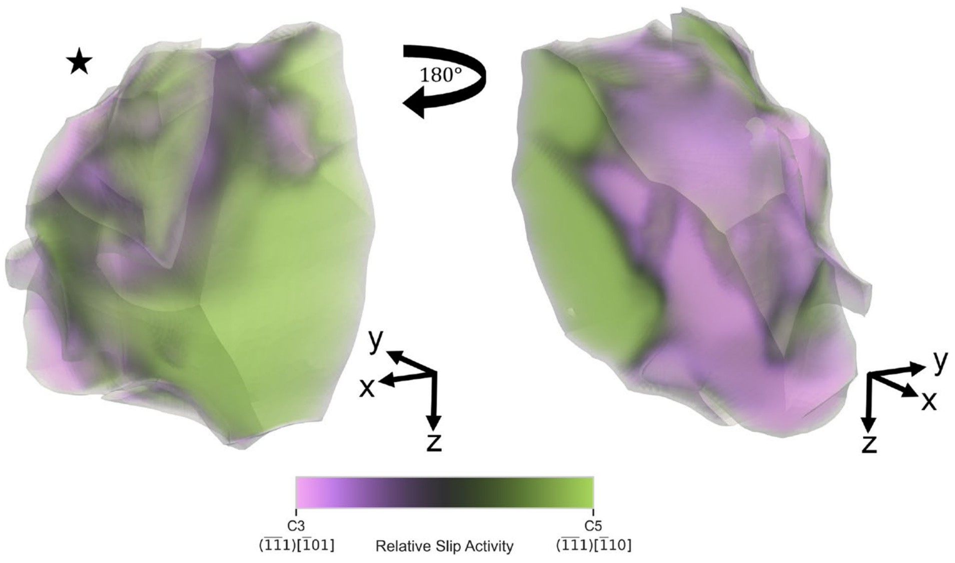

★
180°
y
x
z
C3
$(\bar{1}\bar{1}1)[\bar{1}01]$
Relative Slip Activity
C5
$(\bar{1}\bar{1}1)[\bar{1}10]$

Fig. 11 Calculated relative slip activity of slip systems C3 and C5 throughout the volume of the grain of interest found in Fig. 7c, seen from two viewing angles. Relative activity ranges from 100% C3 to 100% C5. Model data obtained from Stinville et al. [21]

Appendix

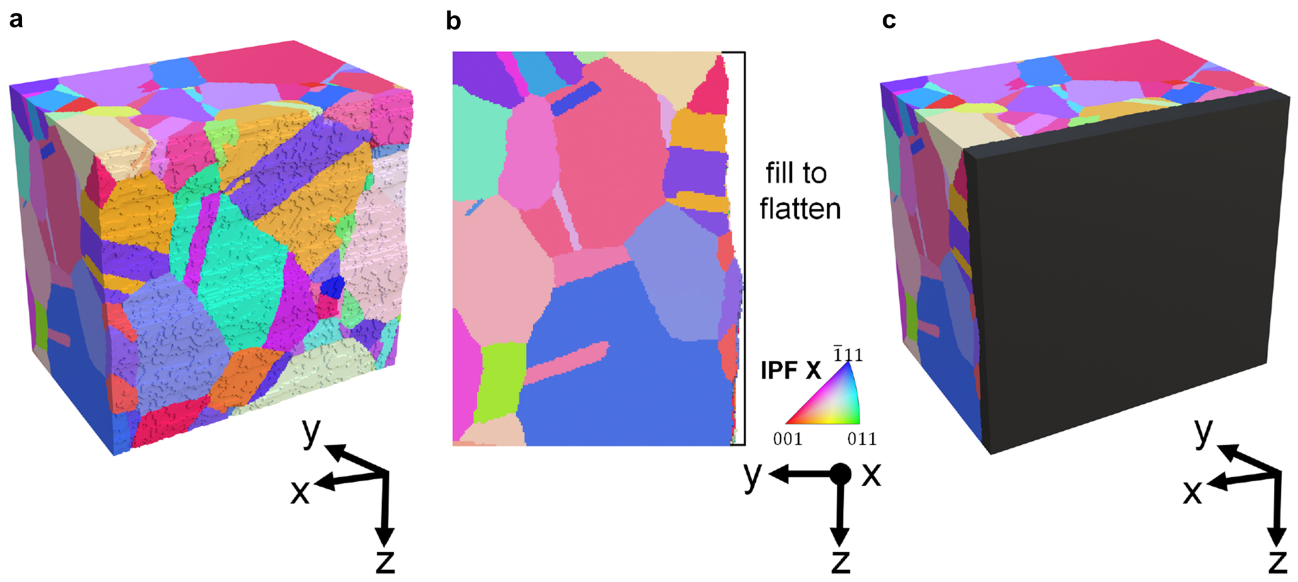


Fig. 12 **a** 3D reconstruction of Tribeam experiment dataset for IN718. **b** Same Tribeam dataset viewed along the x-direction to visualize the rough surface, this area is filled with voxels until flat. **c** Tribeam dataset with the new surface “cap” of voxels in dark gray. Dataset obtained from Stinville et al. [21]

Acknowledgements This work is funded by the U.S. Department of Energy, Office of Basic Energy Sciences Program DE-SC0018901.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Miao J, Pollock TM, Wayne Jones J (2009) Crystallographic fatigue crack initiation in nickel-based superalloy rené 88dt at elevated temperature. Acta Materialia 57(20):5964–5974
2. Stein C, Lee S, Rollett A (2012) An analysis of fatigue crack initiation using 2D orientation mapping and full-field simulation of elastic stress response. Wiley, London
3. Stinville JC, Martin E, Karadge M, Ismonov S, Soare M, Hanlon T, Sundaram S, Echlin MP, Callahan PG, Lenthe WC, Miller VM, Miao J, Wessman AE, Finlay R, Loghin A, Marte J, Pollock TM (2018) Fatigue deformation in a polycrystalline nickel base superalloy at intermediate and high temperature: competing failure modes. Acta Mater 152:16–33. https://doi.org/10.1016/j.actamat.2018.03.035
4. Charpagne M, Hestroffer J, Polonsky A, Echlin M, Texier D, Valle V, Beyerlein I, Pollock T, Stinville J (2021) Slip localization in inconel 718: a three-dimensional and statistical perspective. Acta Materialia 215:117037. https://doi.org/10.1016/j.actamat.2021.117037
5. Cubit geometry and meshing toolkit, sandia national laboratories (2019). http://cubit.sandia.gov
6. Simmetrix inc. simulation modeling suite. http://www.simmetrix.com/
7. Lee M-J, Jeon Y-J, Son G-E, Sung S, Kim J-Y, Han H, Cho S, Jung S-H, Lee S (2018) Grain boundary conformed volumetric mesh generation from a three-dimensional voxellated polycrystalline microstructure. Met Mater Int. https://doi.org/10.1007/s12540-018-0083-x
8. Lee S, Rohrer G, Rollett A (2014) Three-dimensional digital approximations of grain boundary networks in polycrystals. Modell Simul Mater Sci Eng 22(2):025017
9. HyperMesh (2017). https://altairhyperworks.com/product/HyperMesh
10. Quey R, Dawson P, Barbe F (2011) Large-scale 3d random polycrystals for the finite element method: generation, meshing and remeshing. Comput Methods Appl Mech Eng 200(17):1729–1745. https://doi.org/10.1016/j.cma.2011.01.002
11. Hart KA, Rimoli JJ (2020) Microstructpy: a statistical microstructure mesh generator in python. SoftwareX. https://doi.org/10.1016/j.softx.2020.100595
12. Sandström C (2016) Voxel2Tet. https://github.com/CarlSandstrom/Voxel2Tet
13. Hestroffer JM (2021) XtalMesh https://github.com/jonathanhestroffer/XtalMesh
14. Merkel D (2014) Docker: lightweight linux containers for consistent development and deployment. Linux J 2014(239):2
15. Groeber MA, Jackson MA (2014) DREAM3d: a digital representation environment for the analysis of microstructure in 3d. Integr Mater Manuf Innov 3(1):56–72. https://doi.org/10.1186/2193-9772-3-5
16. Hu Y, Schneider T, Wang B, Zorin D, Panozzo D (2020) Fast tetrahedral meshing in the wild. arXiv:1908.03581
17. Zhou Q, Jacobson A, Thingi10k: a dataset of 10, 000 3d-printing models, arXiv:abs/1605.04797
18. Hu Y, Zhou Q, Gao X, Jacobson A, Zorin D, Panozzo D (2018) Tetrahedral meshing in the wild. ACM Trans Graph. https://doi.org/10.1145/3197517.3201353
19. Rabinovich M, Poranne R, Panozzo D, Sorkine-Hornung O (2017) Scalable locally injective mappings. ACM Trans Graph. https://doi.org/10.1145/2983621
20. Jacobson A, Kavan L, Sorkine O, Robust inside-outside segmentation using generalized winding numbers. ACM Trans Graph 32 (4)
21. Stinville J, Hestroffer J, Charpagne M, Polonsky A, Echlin M, Torbet C, Valle V, Shadle D, Nygren K, Miller A, Loghin A, Beyerlein I, Pollock T (2021) Multi-modal dataset of a polycrystalline metallic material: 3d microstructure and deformation fields, in review
22. Stimpson C, Ernst C, Knupp P, Pébay P, Thompson D (2007) The verdict library reference manual
23. Ahrens J, Geveci B, Law C (2005) Paraview: an end-user tool for large data visualization. In: The visualization handbook 2005
24. Stinville J, Charpagne M, Bourdin F, Callahan P, Chen Z, Echlin M, Texier D, Cormier J, Villechaise P, Pollock T, Valle V (2020) Measurement of elastic and rotation fields during irreversible deformation using heaviside-digital image correlation. Mater Character 169:110600. https://doi.org/10.1016/j.matchar.2020.110600