Real-time porosity mapping and visualization for synchrotron tomography

Aniket Tekawade, Viktor Nikitin, Yashas Satapathy, Zhengchun Liu, Xuan Zhang, Peter Kenesei, Francesco De Carlo, Rajkumar Kettimuthu, Ian Foster

***Abstract*—In applications of X-ray computed tomography (CT) for engineering materials and manufactured parts, CT volumes (obtained after reconstruction) represent a 3D spatial density map of the sample material which is often binarized to represent void regions which could be cracks, pores, or manufactured features. The CT volume is an implicit representation of the void region which is often not adequate to realize either quantitative measurements (e.g., porosity analysis) or visualization (e.g., finding cracks). Furthermore, much of the volume represents the same region (either material or void) thereby adding redundant computation when all voxels in the object space are reconstructed. We describe an algorithm and its implementation that directly reduces raw data from a CT detector into explicit void shape representations without reconstructing all voxels. We first implement a filtered back-projection step on GPU that operates on subsets of the voxel space where voids may be present and a compact 3D convolutional neural network that binarizes (segments) that voxel subset. Next, we implement a complete scheme to detect such voxel subsets, reconstruct them and extract a triangular face mesh for visualization. For experiments, we use a real dataset of a mosaic scan of a 6-millimeter-wide and 1-centimeter-tall 3D printed steel part acquired by scanning the sample for 2 hours at 18 different fields-of-view with 1.17 micrometer pixels. We report computation time measurements using a single GPU for selectively reducing 100 gigabytes of 16-bit raw detector data into 3D mesh, we show that real-time (i.e., data-reduction time $<<$ data-acquisition time) mapping of micrometer-sized porosity in centimeter-sized parts is possible on a computer directly linked to the X-ray CT system. With our code, the extraction of face mesh for visualization for raw CT data from a 2 megapixel camera would take between 1-5 minutes in most scenarios.**

***Index Terms*—X-ray Tomography, Computed Tomography, 3D convolutional neural networks, real-time analysis.**

I. INTRODUCTION

RAW data acquired in a typical computed tomography system is a set of projection images (or radiographs) as the sample is rotated about a known axis. The set of projections and the corresponding angles θ are then used to reconstruct a voxel image $f(x, y, z)$ or tomographic object where the intensity f at each voxel coordinate x, y, z scales with the location attenuation coefficient of the sample material (product of material density and mass attenuation coefficient). In many applications of CT for porosity measurement, this voxel image largely represents a single material (e.g., steel, copper, battery material) and void regions representing cracks, individual pores, interconnected porous regions, or manufactured features such as orifices [1]–[5]. The voxel image needs to be further binarized (or segmented) such that the binarized intensity at each voxel coordinate represents either material or void space. This results in an implicit definition of the surface between the void and metal regions. An additional connected components labeling step may assign different labels $l(x, y, z)$ for voids disconnected from each other to allow size and shape measurements. The disconnected voids and their locations in the object form map of porosity in the material. For visualizing this map, a subsequent marching cubes [6] step may be used to compute the explicit surface in the form of a polygonal mesh which stores a list of faces (list of vertices and a vertex connectivity map representing the faces). Ultimately, neither the raw projection data nor the reconstructed voxel images are desired. Typically, such porosity workflows that reduce raw projections to the explicit 3D model result in a reduction of data by several orders of magnitude while preserving information of the measured sample's morphology. If this data-reduction workflow was fast (real-time) as well as repeatable, one may consider it as a form of lossy compression, much like the vectorization of RGB images. Subsequently, one may reduce the raw projection data to 3D models in real-time and there would be no need to retain the projection data. The need to save projection data for porosity measurement applications of micro-CT originates from the lack of automation and fidelity in data reduction from raw data into 3D visualization and measurements of pore morphology. Such real-time data reduction is one of the key bottlenecks to realizing next generation CT applications such as mosaic scanning of larger samples [7], [8] (i.e., scanning the sample at different positions followed by stitching procedures), dimensional metrology [9], [10], or real-time interactive visualization for *in-situ* studies [11].

In this work, we describe the algorithms and implementation behind a real-time porosity mapping and visualization code which is now available as open-source "Tomo2Mesh" (see supporting information in section VI). The code features a *voxel subset reconstruction scheme* that reconstructs any arbitrary subset of voxels from the overall tomography object space representing regions of interest with filtered back-projection (FBP) followed by denoising and binarization with a compact 3D convolutional neural network. For the application of porosity mapping, the regions of interest are first identified by a *coarse mapping scheme* to identify disconnected voids and the detail of voids in these regions is improved to full voxel resolution by the subset reconstruction scheme. Voids are stored as a collection of 3D images with their corresponding locations in the object space. We implemented

All authors are affiliated with Argonne National Laboratory. A. Tekawade, Y. Satapathy, Z. Liu, R. Kettimuthu and I. Foster are with the Data Science and Learning Division. V. Nikitin, P. Kenesei and F. De Carlo are with the X-ray Science Division. X. Zhang is with the Nuclear Science Division.

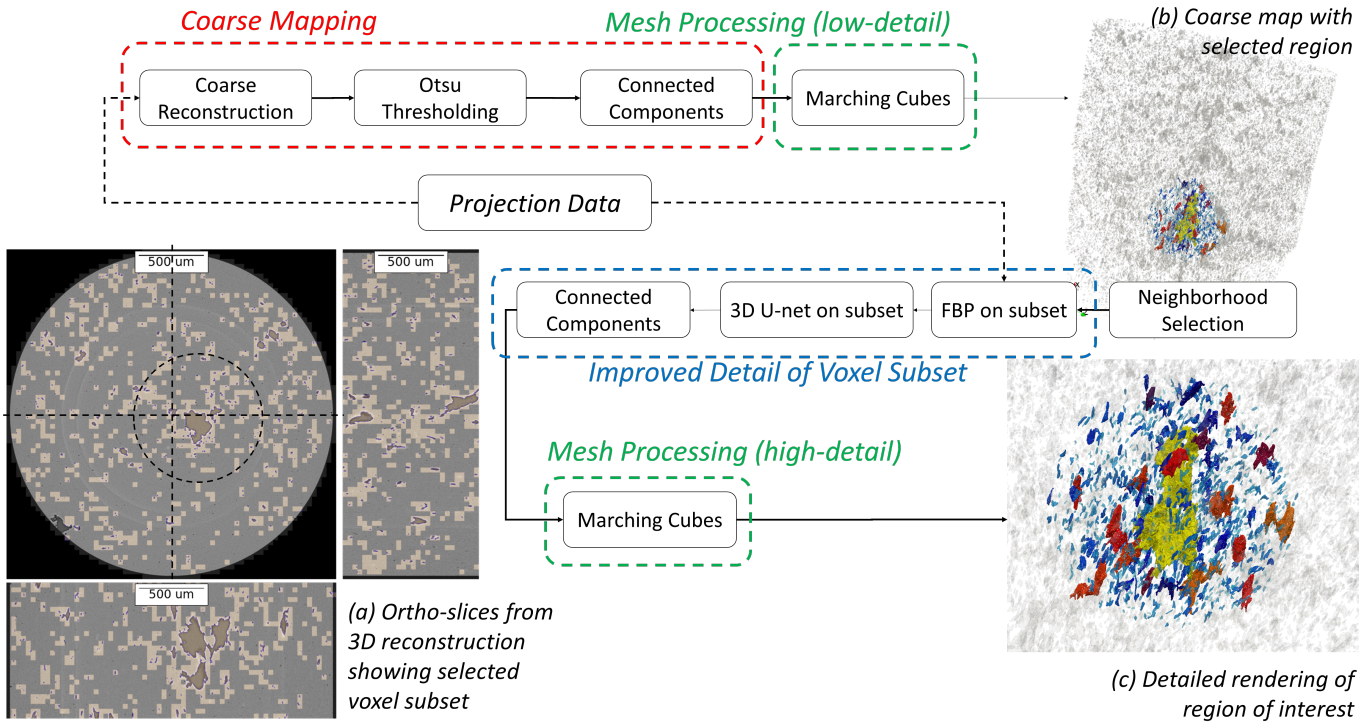

Fig. 1. An illustration of the real-time porosity mapping code. *Coarse mapping:* The projection data is first accessed to perform a coarse reconstruction and disconnected voids are identified with a connected components step. *Voxel subset reconstruction:* Then, a voxel subset is selected based on some criteria and recovered with a subset reconstruction scheme. *Mesh processing:* Marching cubes step generates a face mesh from binarized voxel data. First, a low-detail rendering (b) is generated from the coarse map and a high-detail rendering (c) is generated for the reconstructed voxel subset. Here, we show a selection criteria that finds all voids in a spherical neighborhood of the largest void (see dotted line on the three orthogonal slices of the fully reconstructed 3D volume (a)).

data structures to store and associated methods to process this collection in Python. By measuring the morphological attributes of voids in a collection (e.g., size, Feret diameter, local number density, etc), one may sort or down-select this collection based on shape (e.g., select only large pores or long cracks) or location (e.g., neighborhood of the largest void) for subsequent visualization. A parallel marching cubes implementation extracts a face mesh for each void, assigns texture based on some morphological attribute (e.g., total size), then forms a single face mesh in .ply format for visualization in Paraview or other interactive software. The end-to-end code was optimized for real-time processing on a workstation with a single NVIDIA Quadro RTX 8000 GPU (48 gigabytes of working memory) and 256 gigabytes of RAM. Our choice represents a compute resource that has direct connectivity to a CMOS camera acquiring raw projection images from a X-ray-to-visible-light scintillator [12] in a modern synchrotron tomography system. However, the code can be easily deployed on multiple GPU devices by exploiting the single-program-multiple-data (SPMD) parallelism along the z-axis that is typical for reconstruction algorithms involving synchrotron fan-beam [13].

By ”real-time”, we imply that the processing times are significantly shorter than the data acquisition times. For our time measurements, we used raw data obtained by mosaic scanning a 3d-printed steel cylindrical specimen (nominal diameter 6-millimeter) under monochromatic hard X-ray at the Advanced Photon source (APS) beamline 1-ID. The mosaic comprised of 3 horizontal and 6 vertical positions (total 18 scans). The field-of-view for each scan was approximately 2.2 mm wide and 1.4 mm tall with 1.17 micrometer voxel size. With an exposure time of 0.13 milliseconds for 3000 projections, scanning this mosaic took $\approx$ 2 hours not counting instrument-specific overheads in moving motors to reposition the sample between scans. This resulted in 3000 projections over 180 degrees sample rotation with raw image size 4096$\times$4096 (100 gigabytes of raw data at 16-bit precision) that would reconstruct to a 3D volume with 4096^3 voxels (referred to as *4k* volume). For performance measurements of our code, we generated a *2k* dataset by binning the raw *4k* images ($4096 \times 4096 \rightarrow 2048 \times 2048$) and reducing the angular step size by half resulting in 1500 raw images over 180 degrees rotation. Effectively, this *2k* data would take 1 hour to acquire with resolution of 2.34 micrometer per pixel with the same exposure time per projection. The specimen was printed using a GE Concept Laser Powder Bed Fusion printer, stress-relieved at 650 degC for 1 hour post printing and then creep-tested until rupture at 550 degC, under 275 MPa. The mosaic scans were performed to see through the fractured surface of one of the ruptured pieces. More about the specimen and the test can be found in previous work [14].

The rest of the manuscript is organized as follows. In section III, we describe the subset reconstruction scheme to recover binarized sub-volumes within the tomography object

space and characterize the speed-up achieved with respect to the baseline case where the entire object is recovered. In section IV, we apply this scheme for porosity mapping and visualization in the 3D printed part. By porosity map, we mean computation of a collection of 3D images representing disconnected voids with a corresponding list of their locations in the sample's coordinate system. By visualization, we mean the computation of a triangular mesh representing porosity within the material. The faces in the triangular mesh are colored based on morphological measurements of the void or their position in the sample's coordinate system. In section V, we conclude by discussing the broader impact of this development towards bringing interactive, quantitative visualization closer to the instrument (visualization-at-the-edge) with real-time data reduction.

II. RELATED WORK

A. Porosity analysis

Porosity mapping and measurement is a well-established application of micro-CT spanning various areas of engineering materials research. More recently, there has been a focus on fast and accurate porosity analysis by the additive manufacturing community as a means to realize metrology and inspection of finished parts [15]. Porosity analysis is typically done offline (or in post-processing) using commercial codes [16]. They provide limited ability for customization and the data generated from the instrument cannot be streamed directly to the computation kernels for any real-time processing. Without real-time data reduction, it is not feasible to map larger fields-of-view of a sample at fundamental resolution limit of the CT instrument due to the volume of data generated [7].

B. Real-time CT reconstruction

In the synchrotron community, there is much interest in real-time reconstruction, segmentation (or labeling), and visualization owing to the low exposure time and high data-rates allowed by synchrotron X-ray flux. Synchrotron micro-CT allows acquiring *in-situ* 3D images of pore deformation phenomena, recognized as 4D studies (4th dimension is time). One example is that reported by Carlton et al [17] for studying damage evolution mechanisms in additively manufactured steel. In fact, it was demonstrated that one could perform 3D imaging of crack propagation with 1 second time resolution on a synchrotron micro-CT instrument [18], thereby making data-processing times a key bottleneck to achieve real-time visualization. As mentioned earlier, visualizing pores, cracks, or other types of voids has required a full 3D reconstruction of the object, followed by a segmentation step for labeling voids and subsequent processing into a format such as mesh object for visualization. Real-time 3D reconstruction was previously demonstrate by Bicer et al [19] but it required transferring data to a supercomputer. An "ortho-slice" reconstruction approach was reported by Buurlage et al [11] wherein just 3 orthogonal slices through a predefined voxel coordinate in the object space would be reconstructed. Nikitin et al adopted this approach on a computer directly connected to the camera with cuda-accelerated computing and achieved millisecond latency for reconstructing orthogonal slices [12]. However, this is not a substitute for full 3D reconstruction when the region-of-interest is not previously known.

C. Real-time segmentation and visualization

Furthermore, these 3D reconstruction schemes output grayscale data which needs further processing into a visualization of porosity. We do not know of related work that achieves this in real-time due to the need for automation of the image processing steps and extensive code optimization as the data passes through multiple data structures (raw data, tomographic volume, collection of void data and polygonal mesh for visualization). Convolutional neural networks (CNN) have been employed to solve the automation problem [4], [20]–[23] and it is now conclusive that CNNs are superior to most conventional denoising or segmentation methods even with limited training data. However, we will show here, CNNs are significantly slower due to the several hundred convolution operations involved. In this work, we show that optimizing CNN architectures for short inference time and performing inference only on guessed regions-of-interest saves significant computation time.

III. CT RECONSTRUCTION OF VOXEL SUBSETS

Fig. 1 shows an illustration of the void mapping concept and its implementation. The inset in the left-hand-side shows three orthogonal slices from a fully reconstructed tomographic volume of a 3D printed steel part. The bright voxels represent the steel material while the dark voxels represent void or air. After binarization, the void space is labeled as 1 and the metal is labeled as 0. A rendering of the voids found within a spherical neighborhood of 800-micrometer-radius of the biggest void is shown on the bottom right. This represents the chosen subset that is reconstructed with full voxel resolution. Another rendering of all detected voids colored by their size (i.e., the total volume enclosed by the void) is shown on the top-right. This represents the complete map of voids detected in a coarse reconstruction step. The novelty in our algorithm is in acknowledging that not all voxels in the volume represent useful information. Any voxel in the metal far away from the void will have the same intensity (plus noise) scaling with the attenuation coefficient of metal. If one could rapidly recover a coarse map which approximates the location and extent of the voids, then one could reconstruct only a subset of all the voxels that belong inside and around the voids, resulting in significant saving of computation time when compared to a complete reconstruction of the volume. In the orthogonal slices in Fig. 1, such a subset of voxels comprising of the voids is shown as a set of copper-colored boxes. In this section, we first describe the data structures we implemented to store and process such an arbitrary subset $v \subseteq V$ of the reconstructed object space V (III-A). Then, we describe the implementation of the following key steps: (1) filtered back-projection to recover the subset (III-B), and (2) binary segmentation of the subset using a convolutional neural network (III-C). Denoting the ratio of total number of voxels in v to those in the subset V as $r = N(v)/N(V)$, we show that the net saving in

computation time scales roughly linearly with the "sparsity", which is the value of $1/r$, implying that the algorithm is faster for sparser volumes. The sparsity factor is primarily influenced by the method used for selecting the subset v (referred to as the coarse map). The coarse mapping scheme is described in section IV-A.

A. Data structures for a voxel subset: Grid and Voids

Fig. 2 shows an illustration of two different data structures (*Grid* P and *Voids* S) to represent a neighborhood containing voids. Such a neighborhood is simply a collection of voxels j where each voxel is associated with a 3D coordinate x_j, y_j, z_j and the two data structures store these coordinates in different ways. Our code involves three main steps (1) filtered back-projection (fbp), (2) binarization with 3D U-net and (3) connected components labeling of individual voids. For the fbp stepIII-B, it is sufficient to represent this collection as a C-contiguous array without the need for a special data structure.

1) Grid: However, we implemented the binarization step (section III-C) with a 3D convolutional neural network similar to 3D U-net. 3D U-net exploits spatial locality which means that the grayscale intensity of a given voxel is classified as 1 or 0 with context from the neighboring voxels. The *Grid* data structure splits the voxels from the full object space into a grid of non-overlapping patches with equal size ($32 \times 32 \times 32$) as shown by the yellow and green squares in Fig. 2. The input to U-net is the list of yellow-colored patches (those belonging to the neighborhood where voids exist). We denote the subset of voxels belonging to *Grid* as P.

2) Voids: As shown in Fig. 2, the coordinates of a bounding box that completely encloses a void may not exactly contain whole patches from the *Grid* data structure. Furthermore, the bounding box around a void may contain parts of other voids. To isolate individual voids for measuring their morphological attributes and extracting a triangular mesh separately for each void, we implemented another data structure (referred to as *Voids*) for storing a collection of bounding box coordinates and corresponding sub-volume images of isolated voids (by removing parts of other voids with a connected components labeling step). Next, we implemented methods for transferring voxel coordinates of selected voids from *Voids* to *Grid* for performing subset reconstruction and back from *Grid* to *Voids* for mapping voids between coarse and high-resolution collections for interactive visualization. *Voids* is a Python dictionary of the following lists (1) bounding box coordinates surrounding each void, (2) the binary voxel images representing the void, and (3) the center location (x, y, z) of the void and (4) size of the void (5) specific morphological measurements for each void (e.g., ellipticity, ferret diameter). These lists have length equal to the number of detected voids and each element S_{id} is indexed by a unique void id. This data structure can be easily extended to store additional information.

B. Filtered back-projection

1) Scheme: A typical reconstruction scheme starts with the raw projection data either streamed from the detector or saved to disk after acquisition. Consider such a dataset of projections

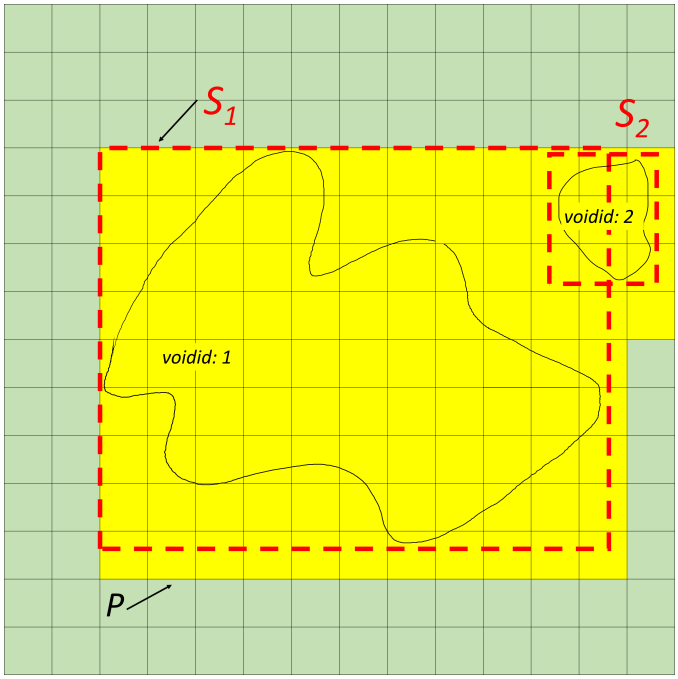

Fig. 2. Illustration of the implementation of the *Grid* (P) and *Voids* (S) data structures to store the subset of voxels within the neighborhood of detected voids. All patches on a regular grid constitute the object space V while the yellow colored patches constitute the subset stored as *Grid* P as an input to 3D U-net. Each patch contains $32 \times 32 \times 32$ voxels. By applying connected components labeling, the subset of the void neighborhood is additionally stored as a collection *Voids* of disconnected voids which are indexed by a unique *voidid* and the individual element only shows a 3D image of the void while excluding parts of other voids appearing in the bounding box.

$d(\theta_i, s, z)$ acquired for K angles (θ_i with $i = 0, \ldots, K-1$) over a span of 180 degrees. Each projection image is a discrete pixel array where each row is associated with a vertical position z and each column is associated with the signed distance s with respect to the rotation axis. This set of projections first needs to be pre-processed before applying the back-projection step (recovering a volume subset from projections). Pre-processing steps include (1) real-space filter to remove outlier pixels or detector noise, (2) correcting for flat-field deviations due to the irregular profile of the synchrotron beam, and (3) applying a convolutional filter such as Ram-Lak, Shepp-Logan or Parzen [24]. The filtering operation involves computing fast Fourier transform (FFT), multiplication by a convolutional filter given in the frequency domain, and inverse FFT. For convolutional filter H the procedure reads as follows,

$$\tilde{d}(\theta_i, s, z) = \mathcal{F}^{-1}(\mathcal{F}(d(\theta_i, s, z)) \cdot H) \quad \forall \theta_i \in i = 0, \ldots, K-1. \tag{1}$$

The next step is the back-projection operation that computes a voxel image for discrete 3D coordinates: $x_j, y_j, z_j \mapsto f_j$, where f_j is a floating point (grayscale) intensity value given for index $j \in V$ of a contiguous (or flattened) array constituting the set V of all voxels within the volume defining the reconstructed

object. The back-projection formula is defined as follows.

$$f_j(x_j, y_j, z_j) = \sum_{i=0}^{K-1} \tilde{d}(\theta_i, x_j \cos\theta_i + y_j \sin\theta_i, z_j) \quad \forall j \in v \tag{2}$$

This formula has previously been employed for real-time reconstruction of arbitrary orthogonal slices through the whole tomographic volume, see Buurlage et al [25] and Nikitin et al [12]. Filtered back-projection was done on a slice by fixing one of the coordinates x_j, y_j, z_j and computing formula (2) for all indexes in other two coordinates. As opposed to the methods with lower computational complexity for evaluating back-projection on the whole image volume [26]–[28], the direct evaluation scheme (2) is favorable when the number of required voxels to compute is significantly smaller than the total number of voxels in the volume. The scheme allows for computing the intensity for individual voxels with a linear computational complexity of $\mathcal{O}(KN)$ with respect to the number of voxels N and number of projection images K. Given a *Grid* P of cubic patches representing the voxel subset v, we may recover the intensity $f_j(x_j, y_j, z_j)$ for all voxels in v rather than the full volume V. The computational complexity of the back-projection step thus scales linearly with the number of voxels belonging to the neighborhood $N(v)$ rather than all the voxels $N(V)$. When this reconstruction scheme is used for recovering only a subset v as opposed to the complete object V, the speed-up in computational time would be very high if the object is largely empty ($r \to 0$). Hence we use the term "sparse volume reconstruction.

2) Implementation: The filtered back-projection (fbp) scheme was implemented on GPU with CuPy [29], a python library for accelerating regular algebraic operations, and with CUDA C++ library for accelerating the back-projection operator on the patches with the contiguous GPU memory access pattern. Only the sinograms required to reconstruct patches in P are transferred to GPU (CPU-GPU transfer time $t_{cpu\to gpu}$) in chunks and the filtering step is performed by using the CuPy FFT library (filter time t_{filt}). For a given P, the coordinates of voxels in it are sorted first along the global corner coordinates c_x, c_y, c_z in the reconstructed volume space, then along the local coordinates within each patch. But to ensure linear complexity for the back-projection, we need to first sort all voxel points along a contiguous array in the global coordinates of V. This introduces a small overhead (time for creating the mask of sorted coordinates t_{mask}) before performing back-projection. We implemented this by first extracting a Boolean array $p_j(x_j, y_j, z_j) \leftarrow P$ from the patches in the selected neighborhood P that acts a mask to identify the index j in the contiguous array where equation 2 needs to be evaluated (back-projection time t_{bp}). Finally, the reconstructed patches included in the voxel subset P are passed to the 3D U-net for binarization and then the binarized data is returned back to CPU memory (GPU-CPU transfer time $t_{gpu\to cpu}$). A expression of the overall execution time for the fbp scheme is shown in equation 3.

$$t_{fbp-gpu} = t_{mask} + t_{filt} + t_{bp} \tag{3a}$$

$$t_{fbp} = t_{cpu\to gpu} + t_{fbp-gpu} + t_{gpu\to cpu} \tag{3b}$$

C. Tiny 3D-U-net for binarization

1) Architecture: After recovering the grayscale intensity for the voxels in subset v using fbp, they are semantically segmented into binary labels (void = 1, metal = 0). This is done by passing cube-shaped grayscale patches from the *Grid* P through an architecture similar to the 3D U-net that outputs corresponding binarized patches. The architecture implemented here (see Fig. 3) is significantly smaller compared to the originally proposed architecture [30]. In the original architecture, 3 analysis and synthesis blocks were used with max-pooling and transpose convolution layers for down and upsampling respectively. Each synthesis and analysis blocks contains two convolutional layers where the second layer has twice the number of filters. Every subsequent synthesis block also has twice the number of filters in the first convolutional layer as compared to the first layer in the previous synthesis block. The first convolutional layer has 32 filters, followed by 64, 128 and so on. However, we found that the binarization of single-component tomography scans does not demand such a deep neural network. Parametric studies were conducted to optimize the architecture for lowest possible inference time without loss of accuracy in the binary labeling of voids. It was first determined that only 2 analysis and synthesis blocks each are sufficient to achieve good binarization accuracy and result in 14% shorter inference time per voxel. As shown in Fig. 3, the max pooling layer has stride 2 after the first analysis block while the one after the second block has stride 4. Furthermore, we minimized the number of filters in the convolutional blocks which led to a drastic speed-up. In our architecture, the total number of filters is reduced by a factor of 4. As shown in Fig. 3, our architecture has 8 filters in the first convolutional layer while the original 3D U-net has 32 [30]. The inference time to process a single voxel t^v_{seg}, (total time for U-net inference divided by the number of voxels), for the original 3D U-net was measured to be 205 nanoseconds while this measurement for our architecture was 28.1 nanoseconds, which is a 7.3X speed-up in binarization speed per voxel achieved by choice of architecture.

2) Implementation: As this is a fully-convolutional neural network, the size of patch passed through the network per inference step can be arbitrary [21]. From time measurements, we found that computation time per voxel is lowest for size 32 and approximately same for larger sizes. Small input sizes allow us to define tighter neighborhoods around the region of interest. Hence the choice of 32. The binarization scheme was implemented with tensorflow-keras [31]. The FBP step described in III-B outputs a CuPy array of shape $(n, 32, 32, 32)$ where n is the number of patches. Because the keras model prediction step is also implemented for GPU, we do a zero-copy operation and pass the data array to tensorflow (using DLPack [32]).After binarization, the list of patches with shape $(n_{patches}, 32, 32, 32)$ are assigned to the corresponding coordinates given by P.

3) Training: For training the network, cube-shaped patches were extracted from a reconstructed volume (separate from the

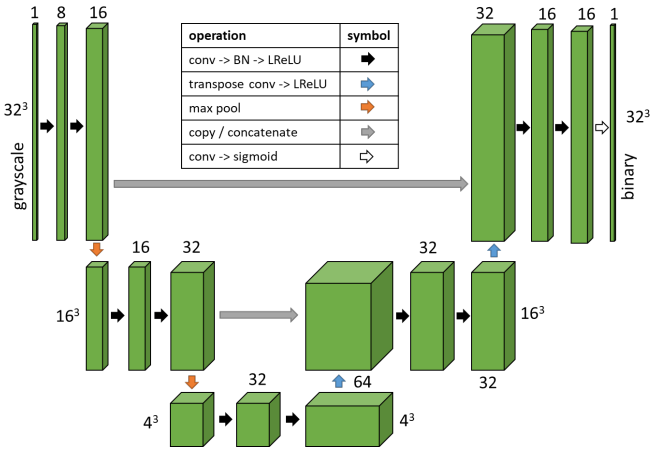

Fig. 3. Architecture diagram of a tiny 3D convolutional neural network similar to the 3D U-net.

test data used for performance analysis) with a corresponding manually segmented volume. These patches contain primarily empty metal regions with voids distributed in a sparse fashion. Hence, a selection criteria was applied to the extracted patches that eliminated empty regions which resulted in dramatic speed-up of convergence during model training.

4) Inference Time: As opposed to the FBP scheme, the complexity of the U-net binarization scales linearly even for very high sparsity and can be estimated by the characteristic t^v_{seg} for a given architecture as in equation 4.

$$t_{seg} = r \cdot N(V) \cdot t^v_{seg} \tag{4a}$$

$$t_{rec} = t_{seg} + t_{fbp} \tag{4b}$$

D. Time Measurements for Subset Reconstruction

For any step in the subset reconstruction scheme, we define speed-up as the ratio of time taken for the baseline case ($t_{subset|r=1}$) to the time taken for a given r. While the computational speed-up for the back-projection step itself would be linear (as discussed in III-B), other factors limit the overall speed-up for very high values of sparsity $1/r$. First, the filtering step of equation 1 is evaluated at once over the full projection width for a single row z_j even if there is just one patch passing through this row. Second, because the filtering step involves an FFT which is much faster on GPU, the incoming data transfer to the GPU would also require the full projection width and would also scale with the number of selected z coordinates regardless of the sparsity. Hence, both the filtering step and the CPU-GPU data transfer become the bottleneck in the case of very sparse volumes. Furthermore, the effect of sparsity on speed-up is not isotropic (equal in the three dimensions). The entire fbp scheme, including filtering and data transfer, will show exactly linear speed-up if the same portion of each sinogram (z-slice) was recovered and only a fraction r of the total number of sinograms was chosen. However, when patches are sparse along x and y, the filtering and data-transfer steps limit the speed-up. To understand this worst-case scenario, we analyzed computation times for different values of sparsity such that the sparsity was created by sampling the same number of patches along each sinogram. The computation times and speed-up with respect to the baseline time for $r = 1$ were measured for both the *2k* and *4k* volumes as shown in Fig. 4. Since the back-projection step is overall much slower than the filtering step, the computation time for overall reconstruction scheme does scale linearly except for very sparse volumes ($r \to 0$) where the filtering step and the CPU-GPU communication dominates. The total reconstruction time includes binarization using the 3D U-net in addition to the fbp step (equation 4). The U-net inference time is a bottleneck for lower values of sparsity. For sparsity greater than 20 ($r = 0.05$), the fbp time becomes the bottleneck since the filtering step dominates over the back-projection step. The total speedup is over 40X at $r = 0.01$ for both *2k* and *4k* volumes. The baseline subset computation time $t_{subset|r=1}$ for *2k* volume was measured to be 5.8 minutes while the same for the *4k* volume was measured to be 58 minutes. The practically realized speed-up may be greater than these reported values because these measurements assume no sparsity along z (all sinograms in the data were passed from cpu to gpu and filter was applied).

IV. REAL-TIME POROSITY MAPPING

In this work, our primary claim is that local reconstruction voxel subsets of interest provides significant speed-up to allow ”real-time” data reduction from raw projection data into void visualization and measurements (or void maps). The subset reconstruction scheme outlined in section III operates on some pre-selected voxel subset v of the full tomography object V and outputs binarized patches aligned with the coordinates of *Grid* P. As seen from the bottom left image of Fig. 1, much of the reconstructed volume contains metal with voids of varying size distributed sparsely throughout the volume. Suppose that the fraction of voxels belonging to the porous neighborhood is $r = N(v)/N(V)$, we may expect that the overall computation time should be reduced in proportion to r. However, the subset reconstruction scheme cannot proceed without first guessing the neighborhood of the voids and computing the coordinates of *Grid* P. Furthermore, the output of the subset reconstruction scheme is a list of sub-volumes aligned with P that do not actually distinguish between individual voids. This distinction is necessary for generating a void map showing individual voids colored by their morphological measurements. As such, there are two more implementation details to be covered for our algorithm. Here, we describe the full void mapping scheme including the ”coarse mapping” step that identifies the subset *Grid* P comprising of detected voids (i.e., excluding the empty metal region) and describe how the output from the subset reconstruction is then exported to the *Voids* data structure for visualization. We conclude the section with an overall performance analysis of the real-time void mapping code.

A. Porosity mapping scheme

1) Coarse reconstruction: To find the neighborhood where voids could exist, we first perform a coarse reconstruction of the entire object by down-sampling the row and height

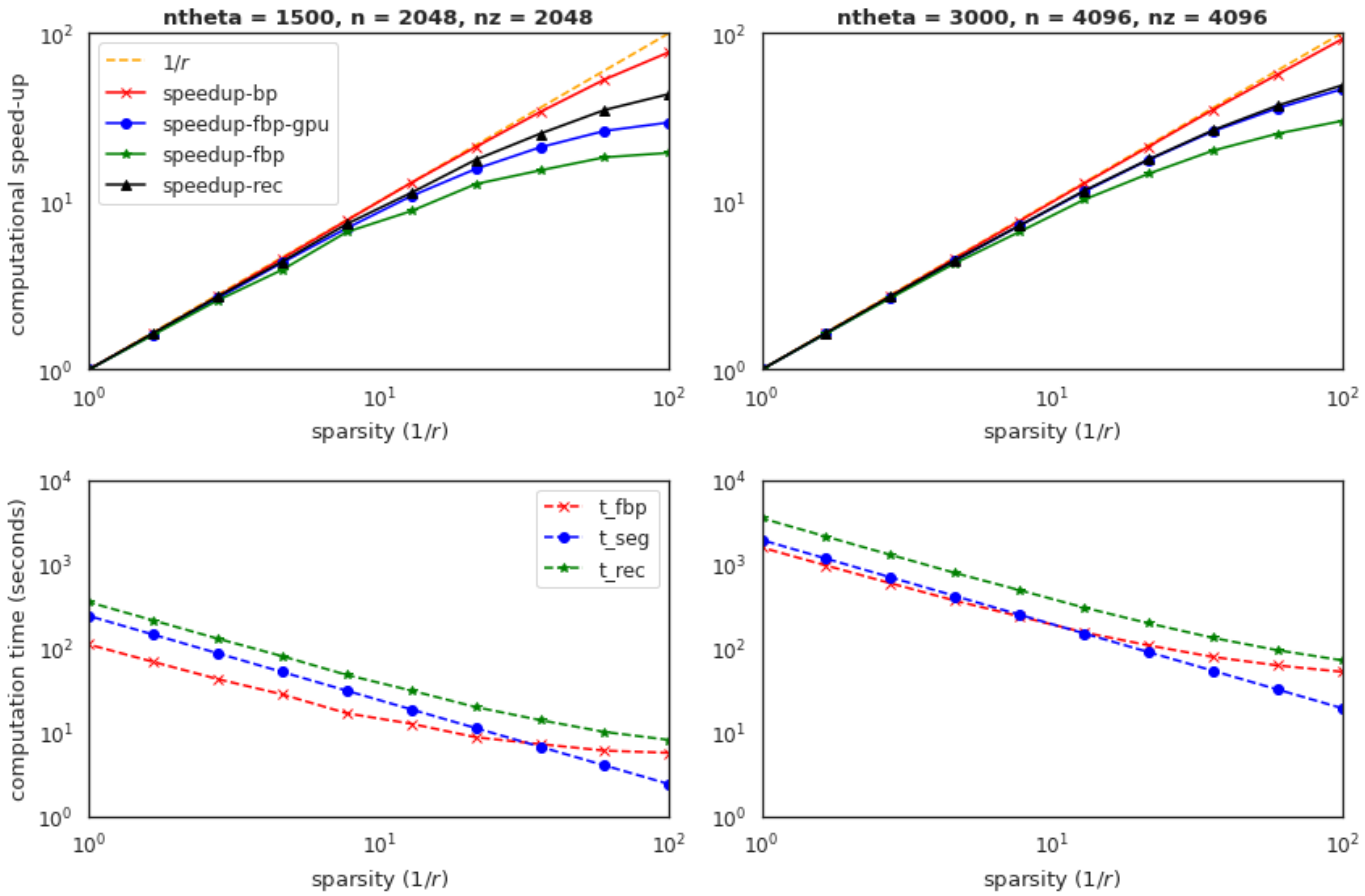

Fig. 4. Measurements of speed-up of the reconstruction scheme for different values of r for two volume sizes. The *2k* volume measurements are on the left while those for the *4k* are shown on the right. Here, *bp* refers to the time for back-projection, fbp refers to the total time for filtered-backprojection including the time for CPU-GPU communication, $fbp - gpu$ refers to the time for fbp step not counting for CPU-GPU communication, tot refers to the total time for fbp and binarization using the 3D U-net.

dimensions of the raw projection array by a factor b and the number of projections K by a factor b_K. Then, all voxels for this volume V_b are reconstructed by equations 1 and 2. To improve contrast, one may prefer average-pooling (or binning), i.e., averaging a bin of four pixels (b = 4) instead of down-sampling where we skip 3 pixels and use the intensity in the fourth pixel. We chose down-sampling because binning introduces additional computation of a mean value with no significant benefit in accuracy for computing a coarse map. From the discussion about time complexity of the fbp (section III-B), it is clear that the fbp time for V_b would be dramatically lower than the full volume V. Because down-sampling will increase the noise-floor in the data, individual voxels may be misclassified as voids of size 1 voxel in the connected components step. We apply a gaussian filter on the back-projected volume to minimize the number of noisy voxels. Finally, otsu thresholding is used to binarize the volume.

2) Connected components labeling: The connected components step on the binarized volume labels voxels in the down-sampled binary volume with a unique *voidid* by disconnected individual voids. From the labeled, we generate a list of bounding boxes that store the (1) position of individual voids, (2) meta-attributes such as total void volume, size and location of the void center and (3) the binary sub-volume that represents each void by implementing a data-structure that we refer to as *Voids* (see section III-A).

3) Selection criteria: We may optionally apply a selection criteria to *Voids* such as "*ignore voids below size of 10 micrometers*" or "*find all voids within 800 micrometers of the largest void*", etc. to down-select the number of voids. In *Tomo2Mesh*, we provide an easy programming interface to implement this selection criteria with a few lines of Python code.

4) Export Voids to Grid: From the bounding boxes S stored in *Voids*, we compute the coordinates for the *Grid* P, i.e., the corner locations of all cubic patches that contain any voids of interest. This is done by filling an empty 3D array with the shape of the full object with the binary images of the voids from the coarse reconstruction, then extracting coordinates P on a $32 \times 32 \times 32$ grid which contain the voids. This set of coordinates along with the complete raw projection data are used to reconstruct and segment the voxel subset as described in section III. The sparsity r for a given selection P depends on the specific criteria applied.

5) Import updated Voids from Grid (re-labeling): After recovering binarized data from the subset reconstruction scheme,

we create a new instance of *Voids* which extracts new binary volumes from the subset reconstruction for the corresponding *voidid* detected by the coarse mapping scheme. Hence, this new *Voids* collection contains the same voids visualized with improved level of detail.

B. Performance Considerations

We discuss performance trade-offs for a special visualization scenario where a coarse reconstruction and connected components step is performed to detect voids on a down-sampled object space (down-sampling factor b), then the surface detail of those voids is improved by the subset reconstruction scheme.

1) Time measurements for 2k volume: The table I shows some measured computation times and number of detected voids for a *coarse-to-fine* scenario on a *2k* volume. These time measurements are for a *2k* volume for different values of b where the original object space is of size $2048 \times 2048 \times 2048$. The coarse reconstruction time includes filtered back-projection followed by a median filter and Otsu thresholding. Labeling time includes the connected components algorithm and finding bounding boxes for the voids labeled after the connected components step. Both reconstruction and labeling could be done at once after transferring the projection data to GPU and only the patches containing voids were returned to CPU. The reconstruction time for subset scheme includes fbp for selected patches followed by 3D U-net for binarization. To ensure that the voids detected during subset reconstruction are aligned with the same void detected during the coarse mapping step, a re-labeling step is required. As seen from these measurements, the total time savings is significant especially when one requires to detect the largest few voids in the sample.

2) Smallest detectable void: It is not possible to detect all voids from a coarse reconstruction that would be otherwise detectable at the native voxel resolution (2.34 μm per voxel for *2k* volume and 1.17 μm for *4k* volume). Our void detection scheme would detect voids that are at least 3 voxels wide in each dimension, meaning that if the down-sampling factor $b = 4$, then the smallest detectable void would be $3 \times 4 \times 2.34 = 28.1 \mu m$ wide (shortest side) whereas the smallest possible void when detected directly from the *2k* volume would have its shortest side $\approx$ $7 \mu m$ wide. This results in a trade-off between void mapping time and minimum detectable void size. Fig. 5 shows histogram of void size obtained for coarse maps with $b = 2$ and $b = 4$ as well as for a void map at the native voxel size of 2.34 μm for the *2k* volume. Here, void size is defined as a cube-root of the total volume enclosed in the void.

3) Improvements from subset reconstruction: The subset reconstruction scheme results in more accurate determination of the void surface (boundary between metal and air) because of the higher voxel-resolution and the use of 3D U-net for binarization. The Fig. 6 shows some comparisons between void binarization obtained from the coarse reconstruction scheme and the subset reconstruction. For each void detected in the coarse mapping scheme, we retain a unique *void id* so that the corresponding detailed reconstruction can be retrieved for any given *void id*. Moreover, if it is determined during the subset reconstruction that a void detected from the coarse reconstruction was in fact a noisy pixel, then that *void id* is simply removed from the *Voids* collection (see for example *void id* 1822 and 2653 in Fig. 6). A 3D image corresponding to a specific *void id* only in the *Voids* collection only stores binary labels within a cropped bounding box for that void while excluding other voids that may appear in the bounding box. This is achieved by a second connected components step after subset reconstruction. An example of this is shown for *void id* 417 and 1734 in Fig. 6.

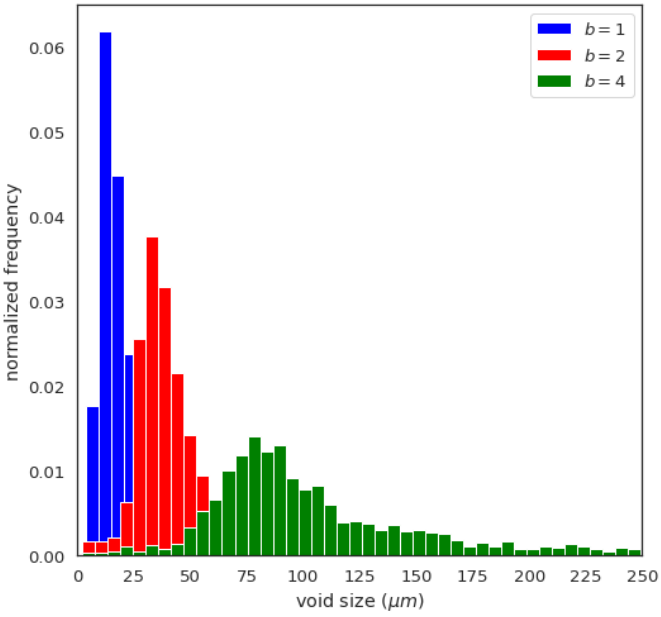

Fig. 5. Histogram plot of sizes of voids detected for the *2k* volume using $b = 1, 2, 4$. The right tail of the histogram was clipped to show the distribution of smaller voids.

C. Smart Visualization

1) Scenarios: As compared to most tomography reconstruction codes, a unique advantage of this subset reconstruction scheme is to recover arbitrarily shaped regions within the object space. To demonstrate this, we conceive of the following visualization scenarios (illustrated in Fig. 7: (1) show all voids larger than $\approx 14 \mu m$ detected with the "coarse-to-fine" described in section IV-B, (2) locate the largest void and show a cylindrical neighborhood of 1600 μm height around it's center and (3) locate the largest void and show a spherical neighborhood of 800 μm radius around it's center. To accomplish scenarios 2 and 3, all detected voids from the coarse reconstruction and further sorted by size and the *void id* of the largest void is identified. Then, a nearest neighbor search is performed on the *Voids* collection to find all voids that lie within the region (Euclidean distance for spherical neighborhood and Z-distance for cylindrical neighborhood). The table II reports measurements of time for the void mapping scheme for these scenarios for the *2k* and *4k* volumes. In this table, we additionally report the meshing time t_{mesh} and

TABLE I
TIME TO COMPUTE A COARSE MAP OF DETECTABLE VOIDS (*2k volume*)

				t_{coarse}		t_{subset}		$t_{mapping}$
b	**size**	**sparsity** $(1/r)$	**voids**	**reconstruction**	**labeling**	**reconstruction**	**re-labeling**	**total**
	pix.	ratio	num.	time (seconds)				
1	2048	1.00	3.59E+05	-	-	360	384	744
2	1024	3.08	3.82E+04	8.91	15.1	125	14.2	163
4	512	11.07	2.63E+03	0.797	3.45	39.3	6.3	49.8

the implementation of the mesh generation implementation is described next.

2) Marching cubes for mesh generation: From the binarized sub-volumes of voids stored in *Voids* (from both the full reconstruction with coarse resolution and subset reconstruction at full voxel resolution), we implemented marching cubes algorithm [6] to compute a triangular mesh object for each void which explicitly represents the surface of the void. A triangular mesh consists of (1) a list of vertices x_k, y_k, z_k that lie on the transition region between the metal and void space and (2) a connectivity map that stores a list of faces, each represented by three vertices and (3) a texture map that stores an RGB value for each vertex. From this mesh, we further reduce the face count by collapsing edges with length less than one pixel length. This make the mesh size manageable for better rendering. Since each void is meshed separately, we used a map-reduce scheme using Python's multiprocessing library to parallelize over the collection of voids. The complete mesh containing all voids is then exported to ".ply" format which is readable by most visualization software (we chose Paraview). For marching cubes, we included an implementation from scikit-image [33] and a subsequent step to reduce the face count by collapsing short edges from PyMesh library [34]. The reduced face count makes the mesh size manageable for better rendering.

3) Additional comments: If these scenarios were implemented with a typical fan-beam reconstruction code such as TomoPy [13], one would not see any time benefit from visualizing a cylindrical or spherical region. To reconstruct a subset of data, one simply select the relevant range of z-coordinates to select the subset of sinograms and reconstruct them. Reconstructing a spherical neighborhood would still require reconstruction of the full cylinder first followed by some cropping scheme and thus not result in any time savings. Because our subset reconstruction scheme performs reconstruction with fbp and U-net on arbitrary patches in the volume, we can realize further speed-up for the scenario involving a spherical neighborhood compared to a cylindrical neighborhood. Additionally, we note that the cylindrical or spherical regions reported in II are not the complete set of voxels in this region but only those voxels around the detected voids. Hence, the subset reconstruction scheme allows superior speed-up for intelligently visualizing porosity information in sparse volumes.

V. CONCLUSIONS AND FUTURE WORK

Our code allows selective visualization of porosity distribution in the vast majority of tomography data for engineering material applications within minutes of scan using a high-performance workstation available at most synchrotron tomography beamlines. Depending on the number of GPU devices available, we can now measure, visualize and automate acquisition of porosity data with feedback time for complete data-processing within a few minutes. With further code optimization, we hope to further reduce processing time. Some areas for for further improvements are (1) overlapping CPU-GPU data-transfer and computation, (2) optimized implementation of the connected components scheme and (3) improving the marching cubes step for generating a triangular mesh or exploring alternative data formats for visualizing void data. By processing raw projection data directly streamed from the detector, we may conceive of a novel tomography instrument that can provide immersive 3D visualization at the edge without ever having to save a byte of data. Infrastructure to stream raw projection data from tomography instruments is rapidly developing at some synchrotron beamlines [12]. With stream processing of porosity data, we can identify void clusters, cracks or other regions of interest for automation of subsequent high-resolution correlative measurements.

TABLE II
TIME (SECS) TO VISUALIZE NEIGHBORHOOD AROUND LARGEST VOID

2k volume, b = 2

void selection criteria	**sparsity** $(1/r)$	$t_{mapping}$	t_{mesh}
all voids with b = 2	3.079	162	40
cylindrical neighborhood	8.014	77.16	20.8
spherical neighborhood	70.09	37	12

4k volume, b = 4

void selection criteria	**sparsity** $(1/r)$	$t_{mapping}$	t_{mesh}
all voids with b = 4	6.496	792	241
cylindrical neighborhood	16.56	328	91
spherical neighborhood	119.4	109	59

VI. SUPPORTING INFORMATION

The open-source code "Tomo2Mesh" is available on github (github.com/aniketkt/Tomo2Mesh). On the project homepage, a link to publicly accessible globus endpoint is provided for retrieving the dataset and trained U-net models used for performance measurements. To reproduce time measurements, the code can be accessed via commit id 40df7048a93dc43eb16f53afc3ae95bda7ec1e3f.

VII. ACKNOWLEDGMENT

This effort was primarily funded through a Laboratory Directed Research and Development (LDRD) award titled

"AI-steer: Online steering of light source experiments" from Argonne National Laboratory. We wish to further thank Dr. Meimei Li for support of data and guidance. The data were acquired at beamline 1-ID of Advanced Photon Source (APS). Use of the APS is supported by the U.S. Department of Energy (DOE) under Contract No. DEAC0206CH11357.

REFERENCES

[1] X. Zhang, "Unveiling microstructure-property correlations in nuclear materials with high-energy synchrotron x-ray techniques," *Journal of Nuclear Materials*, p. 153572, 2022.

[2] D. B. Menasche, J. Lind, S. F. Li, P. Kenesei, J. F. Bingert, U. Lienert, and R. M. Suter, "Shock induced damage in copper: A before and after, three-dimensional study," *Journal of applied physics*, vol. 119, no. 15, 2016.

[3] D. B. Menasche, P. A. Shade, J. Lind, S. F. Li, J. V. Bernier, P. Kenesei, J. C. Schuren, and R. M. Suter, "Correlation of thermally induced pores with microstructural features using high energy x-rays," *Metallurgical and Materials Transactions A*, vol. 47, no. 11, pp. 5580–5588, 11 2016.

[4] M. B. Dixit, A. Verma, W. Zaman, X. Zhong, P. Kenesei, J. S. Park, J. Almer, P. P. Mukherjee, and K. B. Hatzell, "Synchrotron imaging of pore formation in Li metal solid-state batteries aided by machine learning," *ACS Applied Energy Materials*, 2020.

[5] J. Thomas, A. Figueroa Bengoa, S. T. Nori, R. Ren, P. Kenesei, J. Almer, J. Hunter, J. Harp, and M. A. Okuniewski, "The application of synchrotron micro-computed tomography to characterize the three-dimensional microstructure in irradiated nuclear fuel," *Journal of Nuclear Materials*, vol. 537, p. 152161, 2020.

[6] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, p. 163–169, aug 1987. [Online]. Available: https://doi.org/10.1145/37402.37422

[7] J. Kim, A. Slyamov, E. Lauridsen, M. Birkbak, T. Ramos, F. Marone, J. W. Andreasen, M. Stampanoni, and M. Kagias, "Macroscopic mapping of microscale fibers in freeform injection molded fiber-reinforced composites using x-ray scattering tensor tomography," *Composites Part B: Engineering*, p. 109634, 2022.

[8] M. A. Yakovlev, D. J. Vanselow, M. S. Ngu, C. R. Zaino, S. R. Katz, Y. Ding, D. Parkinson, S. Y. Wang, K. C. Ang, P. La Riviere *et al.*, "A wide-field micro-computed tomography detector: micron resolution at half-centimetre scale," *Journal of Synchrotron Radiation*, vol. 29, no. 2, 2022.

[9] H. Villarraga-Gomez, N. Kotwal, R. Parwani, D. Weiß, M. Krenkel, W. Kimmig, and C. G. vom Hagen, "Improving the dimensional accuracy of 3d x-ray microscopy data," *Measurement Science and Technology*, 2022.

[10] A. Tekawade, B. A. Sforzo, K. E. Matusik, A. L. Kastengren, and C. F. Powell, "High-fidelity geometry generation from CT data using convolutional neural networks," in *Developments in X-Ray Tomography XII*, B. Müller and G. Wang, Eds. SPIE, 9 2019, p. 67.

[11] J.-W. Buurlage, F. Marone, D. M. Pelt, W. J. Palenstijn, M. Stampanoni, K. J. Batenburg, and C. M. Schlepütz, "Real-time reconstruction and visualisation towards dynamic feedback control during time-resolved tomography experiments at tomcat," *Scientific Reports*, vol. 9, no. 1, pp. 1–11, 2019.

[12] V. Nikitin, A. Tekawade, A. Duchkov, P. Shevchenko, and F. De Carlo, "Real-time streaming tomographic reconstruction with on-demand data capturing and 3D zooming to regions of interest," *Journal of Synchrotron Radiation*, vol. 29, no. 3, May 2022. [Online]. Available: https://doi.org/10.1107/S1600577522003095

[13] D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "TomoPy: A framework for the analysis of synchrotron tomographic data." *Journal of Synchrotron Radiation*, vol. 21, no. Pt 5, pp. 1188–1193, 2014.

[14] M. Li, W.-Y. Chen, and X. Zhang, "Effect of heat treatment on creep behavior of 316 l stainless steel manufactured by laser powder bed fusion," *Journal of Nuclear Materials*, vol. 559, p. 153469, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022311521006899

[15] A. Du Plessis, I. Yadroitsev, I. Yadroitsava, and S. G. Le Roux, "X-ray microcomputed tomography in additive manufacturing: a review of the current technology and applications," *3D Printing and Additive Manufacturing*, vol. 5, no. 3, pp. 227–247, 2018.

[16] A. Du Plessis, B. J. Olawuyi, W. P. Boshoff, and S. G. Le Roux, "Simple and fast porosity analysis of concrete using x-ray computed tomography," *Materials and structures*, vol. 49, no. 1, pp. 553–562, 2016.

[17] H. D. Carlton, A. Haboub, G. F. Gallegos, D. Y. Parkinson, and A. A. MacDowell, "Damage evolution and failure mechanisms in additively manufactured stainless steel," *Materials Science and Engineering: A*, vol. 651, pp. 406–414, 2016.

[18] E. Maire, C. Le Bourlot, J. Adrien, A. Mortensen, and R. Mokso, "20 hz x-ray tomography during an in situ tensile test," *International Journal of Fracture*, vol. 200, no. 1, pp. 3–12, 2016.

[19] T. Bicer, D. Gursoy, R. Kettimuthu, I. T. Foster, B. Ren, V. De Andrede, and F. De Carlo, "Real-time data analysis and autonomous steering of synchrotron light source experiments," in *13th International Conference on e-Science*. IEEE, 2017, pp. 59–68.

[20] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo, and I. Foster, "Tomogan: low-dose synchrotron x-ray tomography with generative adversarial networks: discussion," *JOSA A*, vol. 37, no. 3, pp. 422–434, 2020.

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[22] Z. Su, E. Decencière, T.-T. Nguyen, K. El-Amiry, V. De Andrade, A. A. Franco, and A. Demortière, "Artificial neural network approach for multiphase segmentation of battery electrode nano-ct images," *npj Computational Materials*, vol. 8, no. 1, pp. 1–11, 2022.

[23] S. Niverty, H. Torbatissaraf, V. Nikitin, V. De Andrade, S. Niauzorau, N. Kublik, B. Azeredo, A. Tekawade, F. De Carlo, and N. Chawla, "Computational imaging in 3d x-ray microscopy: Reconstruction, image segmentation and time-evolved experiments," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3502–3506.

[24] F. Natterer, *The Mathematics of Computerized Tomography*. SIAM, 2001.

[25] J.-W. Buurlage, H. Kohr, W. J. Palenstijn, and K. J. Batenburg, "Real-time quasi-D tomographic reconstruction," *Measurement Science and Technology*, vol. 29, no. 6, p. 064005, 2018. [Online]. Available: https://doi.org/10.1088/1361-6501/aab754

[26] A. Dutt and V. Rokhlin, "Fast Fourier transforms for nonequispaced data," *SIAM Journal on Scientific computing*, vol. 14, no. 6, pp. 1368–1393, 1993. [Online]. Available: https://doi.org/10.1137/0914081

[27] G. Beylkin, "On applications of unequally spaced fast Fourier transform," *Mathematical Geophysics Summer School (Stanford Univ., Stanford, 1998)*, 1998. [Online]. Available: https://amath.colorado.edu/faculty/beylkin/papers/applusfft.pdf

[28] F. Andersson, M. Carlsson, and V. V. Nikitin, "Fast algorithms and efficient GPU implementations for the Radon transform and the back-projection operator represented as convolution operators," *SIAM Journal on Imaging Sciences*, vol. 9, no. 2, pp. 637–664, 2016. [Online]. Available: https://doi.org/10.1137/15M1023762

[29] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, "CuPy: A NumPy-compatible library for NVIDIA GPU calculations," *31st Conference on Neural Information Processing Systems*, vol. 151, 2017.

[30] O. Cicek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-net: Learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention*. Cham: Springer, 2016, vol. 9901, pp. 424–432.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation*. USENIX, 2016, pp. 265–283.

[32] "Dlpack: Open in memory tensor structure," 2022. [Online]. Available: https://github.com/dmlc/dlpack

[33] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453

[34] "Pymesh documentation," https://pymesh.readthedocs.io/en/latest/, accessed: 2022-07-01.

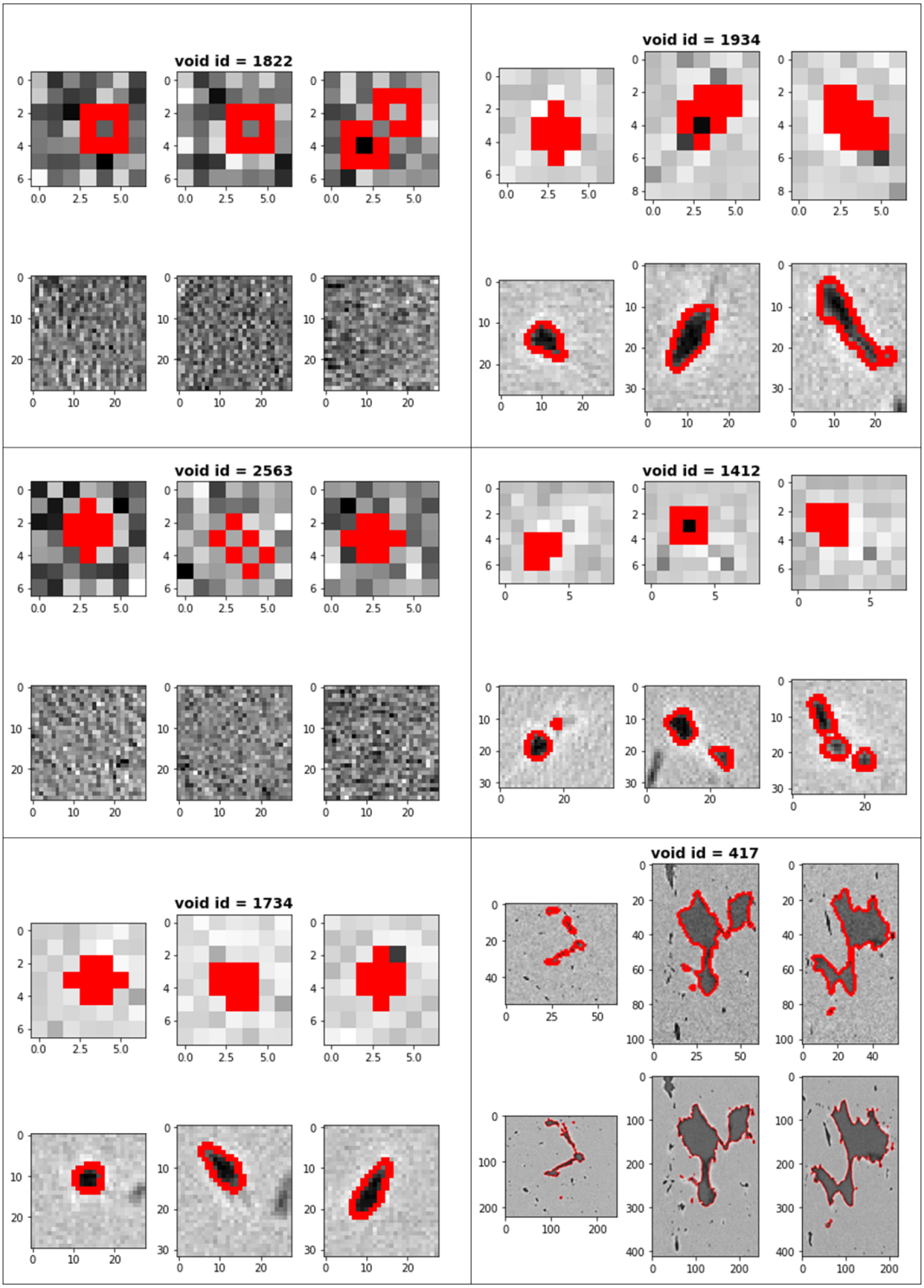

Fig. 6. Comparison of binarized output between coarse reconstruction (top) and the corresponding subset reconstruction (bottom) for six detected voids from a coarse map of the $2k$ volume with $b = 4$. Each image shows in red the edge-map of the binarized output against the corresponding grayscale intensity from the fbp output. The binarization for the coarse map is done on a down-sampled volume (factor $b = 4$ using Otsu thresholding) while the binarization for the detailed map is done by the subset reconstruction scheme which uses 3D U-net. For each void, the three orthogonal slices are shown. Regions which are incorrectly detected as voids are then found to have no void from the detailed reconstruction and the *Voids* data structure is updated accordingly.

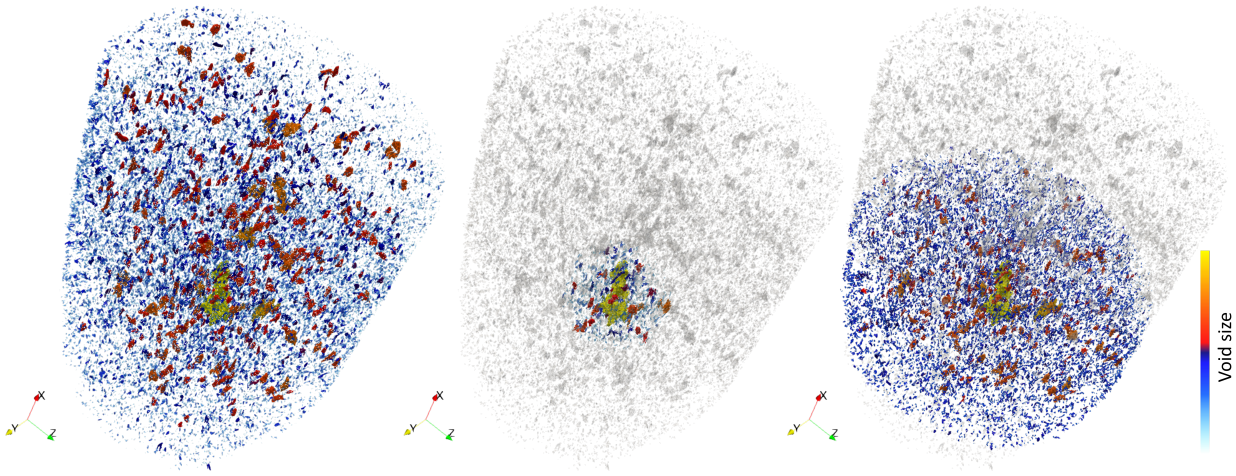

Fig. 7. Illustration of three smart visualization scenarios. In (1), all voids detectable from the *2k* volume with $b = 2$ ($\approx 14\mu m$ or greater) are shown and colored by size. In (2) and (3), cylindrical and spherical neighborhoods respectively are identified and shown around the largest detected void (seen in yellow color).