Electronic Supporting Information

Real-Time Tilt Undersampling Optimization during Electron Tomography of Beam Sensitive Samples using Golden Ratio Scanning and RECAST3D

Timothy M. Craig,[a] Ajinkya A Kadu,[a,b] Kees Joost Batenburg,[b,c] and Sara Bals*[a]

[a]Electron Microscopy for Materials Science and NANOlab Center of Excellence, University of Antwerp, Groenenborgerlaan 171, Antwerp 2020, Belgium.

[b]Centrum Wiskunde & Informatica, Science Park 123, Amsterdam 1098 XG, The Netherlands

[c]Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333CA Leiden, The Netherlands.

* Corresponding author. Email: sara.bals@uantwerpen.be

A Methods

A.1 RECAST3D

RECAST3D is a tool for real-time, quasi-3D reconstruction and visualization. When used at a microscope, it enables real-time viewing of three reconstructed orthoslices directly from collected images. The full code and instructions for operation and installation can be found at `https://github.com/cicwi/RECAST3D`. In this section, we describe the implementation and modifications made to RECAST3D for the purposes of this work.

RECAST3D consists of three libraries: the main library for visualizing orthoslice data, SliceRecon for performing reconstructions from projection data, and Tomopackets for transporting data between layers via packets (Figure A.1a). Buurlage et al. developed a framework for adding custom python scripts to increase the flexibility of this architecture.[1] One such script is the data adapter, which allows users to import saved projection data, create projection geometries, and perform alignments before orthoslice reconstruction by SliceRecon. Another potential add-on script is a post-reconstruction plugin, in which we developed `beam_analysis.py` to calculate quantitative indicators (SROD and SNR) as described in the main text.

To use RECAST3D for beam damage analysis (Figure A.1b), we made several changes to address two crucial issues in the RECAST3D

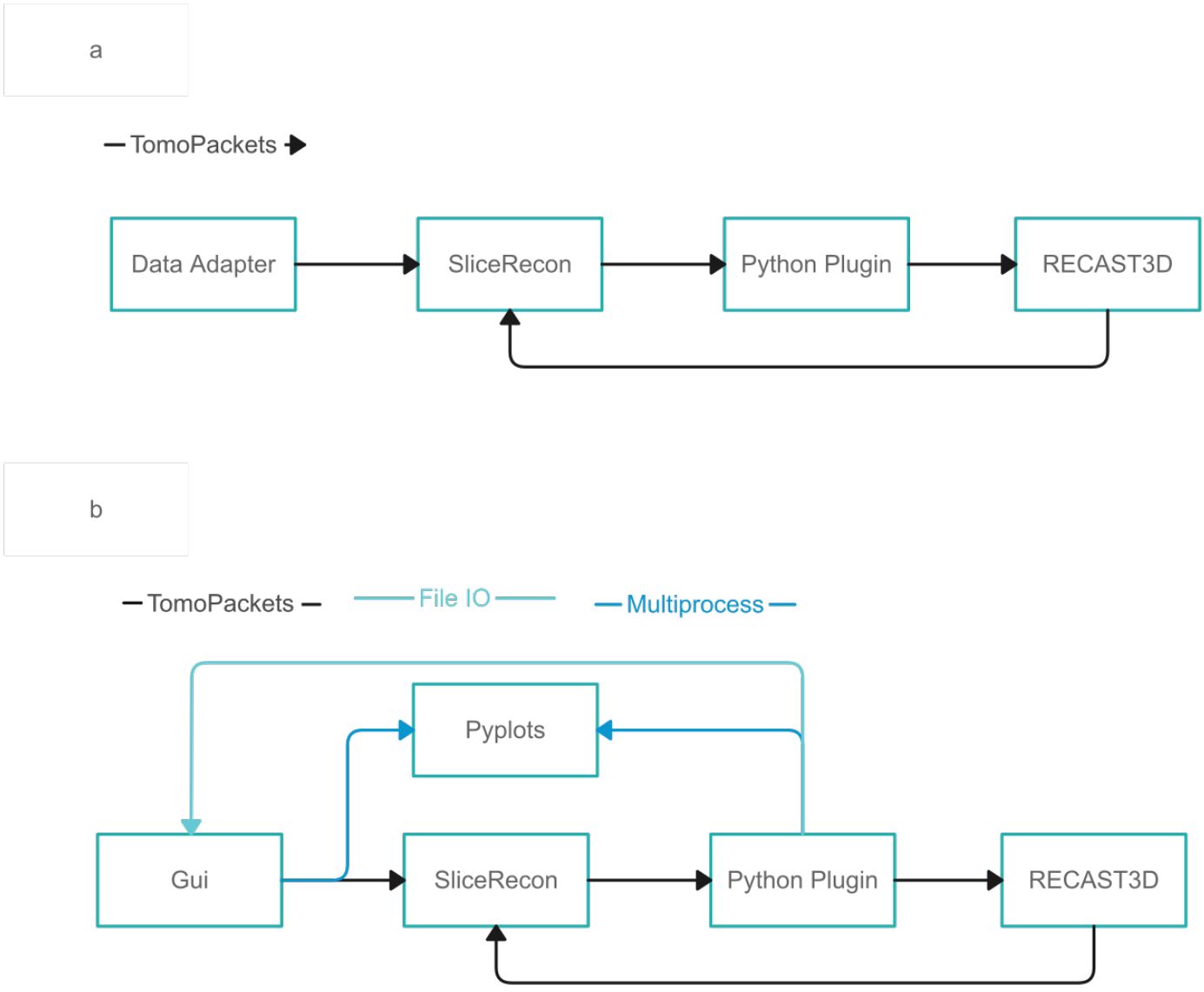

Fig. A.1 Illustration of RECAST3D workflow prior (a) and post modification (b). The primary modifications involved the substitution of the data adapter with a GUI, the implementation of a plotting library using multiprocessing for visualizing analysis and alignment data from the GUI and plug-in, and the introduction of a file input-output (IO)-based method for communication between the GUI and plug-in through the reading and writing of settings. These modifications aimed to enhance the functionality and usability of the RECAST3D workflow.

workflow. The first issue was the difficulty of performing alignments during real-time analysis. To address this, we added the Python Pyplots package for animating tilt-series images and inspecting alignment quality.[2]. We also implemented Python multiprocessing to view the tilt-series images, preventing the reconstruction from slowing down during the viewing process.[3] The tilt-series images were simultaneously piped to SliceRecon and Pyplots, and the Pyplots interface was also used to plot the SROD and SNR data from the plugin.

The second issue with RECAST3D for beam damage analysis was that the plugin could not access the slice orientation. This is because the plugin calculates the SNR and SROD by comparing the current orthoslices to the previous set of orthoslices. However, the user can change the slice orientation at will, which would result in comparing two sets of slices with different positions/orientations. To address this issue, we modified Tomopackets to pass the orientation to the plugin. In the case of a change in orientation, a request is sent to the data adapter to restart the reconstruction and SROD/SNR analysis. Finally, for convenience, we replaced the data adapter script with a graphical user interface.

A.2 Simulations

Simulations were conducted using an iterative routine as depicted in Figure A.2. The input parameters included an initial 3D volume (V_0), two deformation parameters (β_1, β_2), a set of collection times (t), and tilt angles (θ). The collection times correspond to the time elapsed between the collection of each projection in the tilt series. For instance, a set of $[1, 2.5]$ indicates a tilt series with two projections, the first of which was collected at a time of 1 arbitrary unit, and the second was collected 2.5 units after the first. The collection times were normalized by the minimum collection time and rounded to the nearest integer to determine an iteration number (N_I) according to

$$N_I = \left\lfloor \frac{t}{\min(t)} \right\rceil. \tag{A.1}$$

A deformation function (f_{def}), described in the subsequent subsection, was applied iteratively to the volume from $j = 0$ to the sum of N_I, as shown below:

$$V_j = f_{\text{def}}\left(V_{j-1}, \beta_1, \beta_2\right) \tag{A.2}$$

The set N_I defines the number of deformation iterations applied between projections. For example, the initial volume is deformed four times for the set $N_I = [1, 3]$, as the first projection is collected after 1 iteration of deformation, while the second projection is collected three iterations after the first. It is worth noting that the projections are obtained by forward projecting the deformed volume

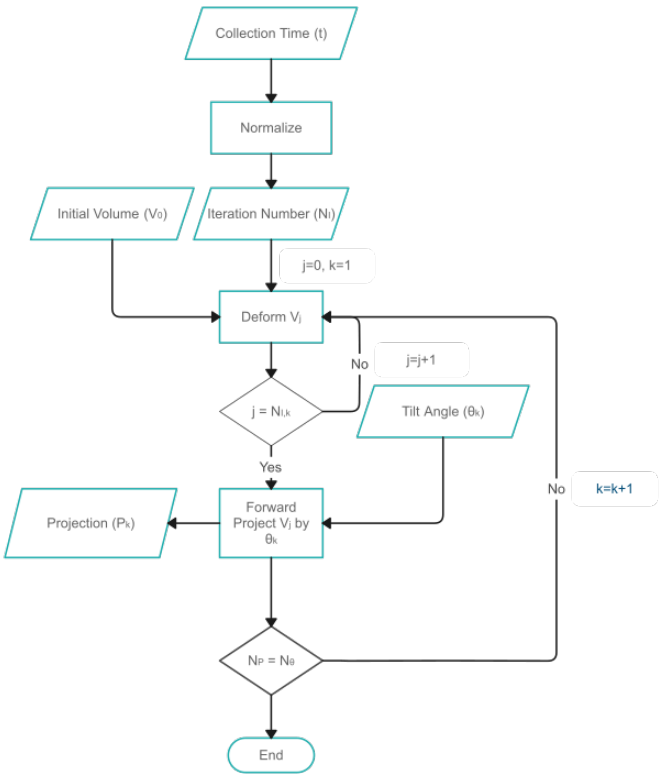

Fig. A.2 Simulations are performed by iteratively deforming an initial volume for a set number of iterations. An artificial tilt-series is generated from the simulations by iteratively forward projecting the deformed volumes by the angle of collection. The process lasts until a projection is acquired for every angle in the tilt-series.

V_j at the defined tilt angle θ_j, as described in the following:

$$P_j = \mathrm{FP}\left(V_j, \theta_j\right), \tag{A.3}$$

where FP represents forward projection operation.

Two assumptions were made about the acquisition times for the sake of convenience. Firstly, the time taken to collect each projection in a tilt series is the same, *e.g.*, $t = [1,2,3,\ldots]$. Secondly, the collection time for each projection during GRS and IS acquisition is the same. Specifically, only 1 iteration of deformation per projection was considered in all simulation experiments, regardless of the acquisition angle or collection method.

A.2.1 Beam Damage Deformation.

In order to simulate beam damage, we employ a two-step deformation process that is parameterized by β_1 and β_2. The first step involves applying elastic deformation to the sample through the use of a voxel mask (M) with random values ranging from -1 to 1. A gaussian filter is then applied to smooth the values of the mask based on the intensities of adjacent voxels, using a standard deviation of 10, which indicates that the random array is smoothed with the nearest 30 voxels for each voxel. The voxel mask is then scaled by β_1, and the voxels of the sample (V) are transformed according to the mask M using elementwise matrix multiplication, *i.e.*,

$$V_i = M \odot V_{i-1}, \tag{A.4}$$

where $\odot$ represents elementwise matrix multiplication. In the second step, we simulate knock-on damage. We first determine the number of non-zero neighboring voxels (NN) for every non-zero voxel in the sample. We then compute the probability P for each voxel based on NN and the deformation parameter β_2 using

$$P = \begin{cases} \beta_2^{NN/3} & \text{if } NN < 27 \\ 0 & NN = 27 \end{cases}. \tag{A.5}$$

If a random number between 0 and 1 is greater than probability P for a given voxel, the intensity of that voxel is set to 0.

A.3 Simulated and Experimental Samples

Table A.1 Collected tilt-series

Type	sample	name	$\beta 1 : \beta 2$	acquisition type	annular range [°]	tilt step [°]
simulation	nanocage	NC-1	0:0	GRS	±70	2, 5, 7, 10, 14, 35, 70
			0:0	IS	±70	
		NC-2	0.3:0.03	GRS	±70	2, 5, 7, 10, 14, 35, 70
			0.3:0.03	IS	±70	
		NC-3	0.55:0.055	GRS	±70	2, 5, 7, 10, 14, 35, 70
			0.55:0.055	IS	±70	
		NC-4	0.6:0.06	GRS	±70	2, 5, 7, 10, 14, 35, 70
			0.6:0.06	IS	±70	
real	nanostar	Au/Pd NS		GRS	±70	2, 5
				IS	±70	
	Au@NU-1000	Au@NU-1000		GRS	±70	
	Au/Pd@ZIF-8	Au/Pd@ZIF-8		GRS	±70	

B Experimental Data

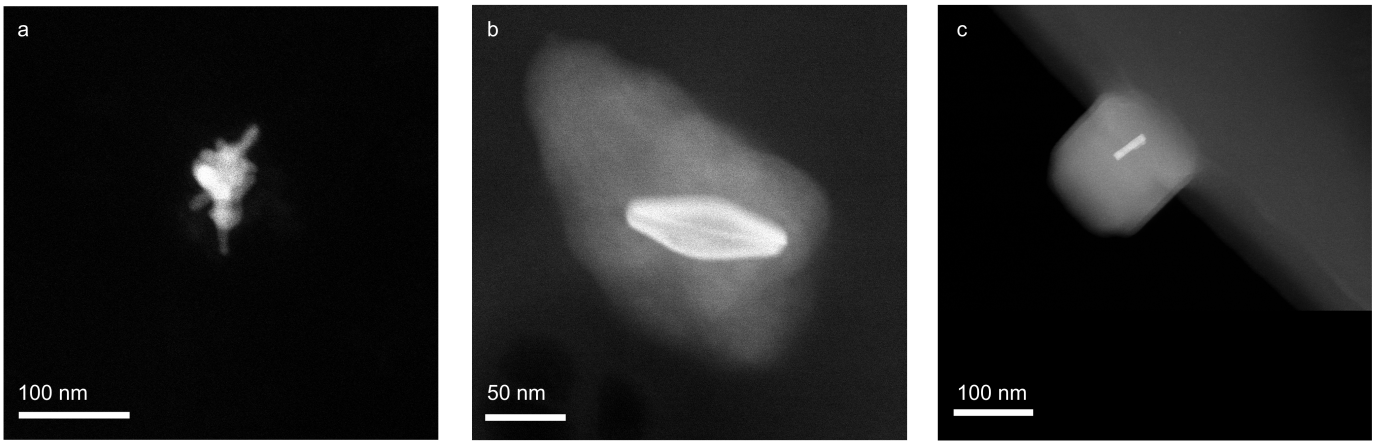

Fig. B.1 Projection of a Au/Pd Nanostars collected with HAADF (a) and Au@NU-1000 (b) and Au/Pd@ZIF-8 (c) NP@MOF complexes collected with ADF.

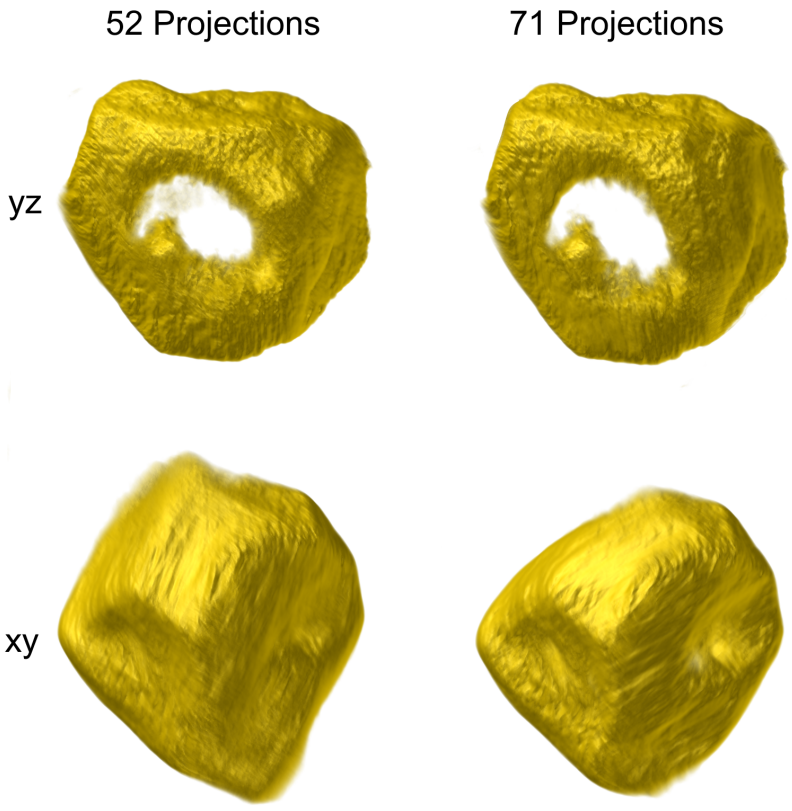

Fig. B.2 3D reconstruction of NS-4 acquired with IS (2° increment) terminated with 52 or 71 projections shown along the yz and xz plane. As more projections are collected from 52 to 71 projections, missing wedge artefacts in the xy plane are corrected but at a trade-off of increased beam damage artefacts visible in the yz plane.

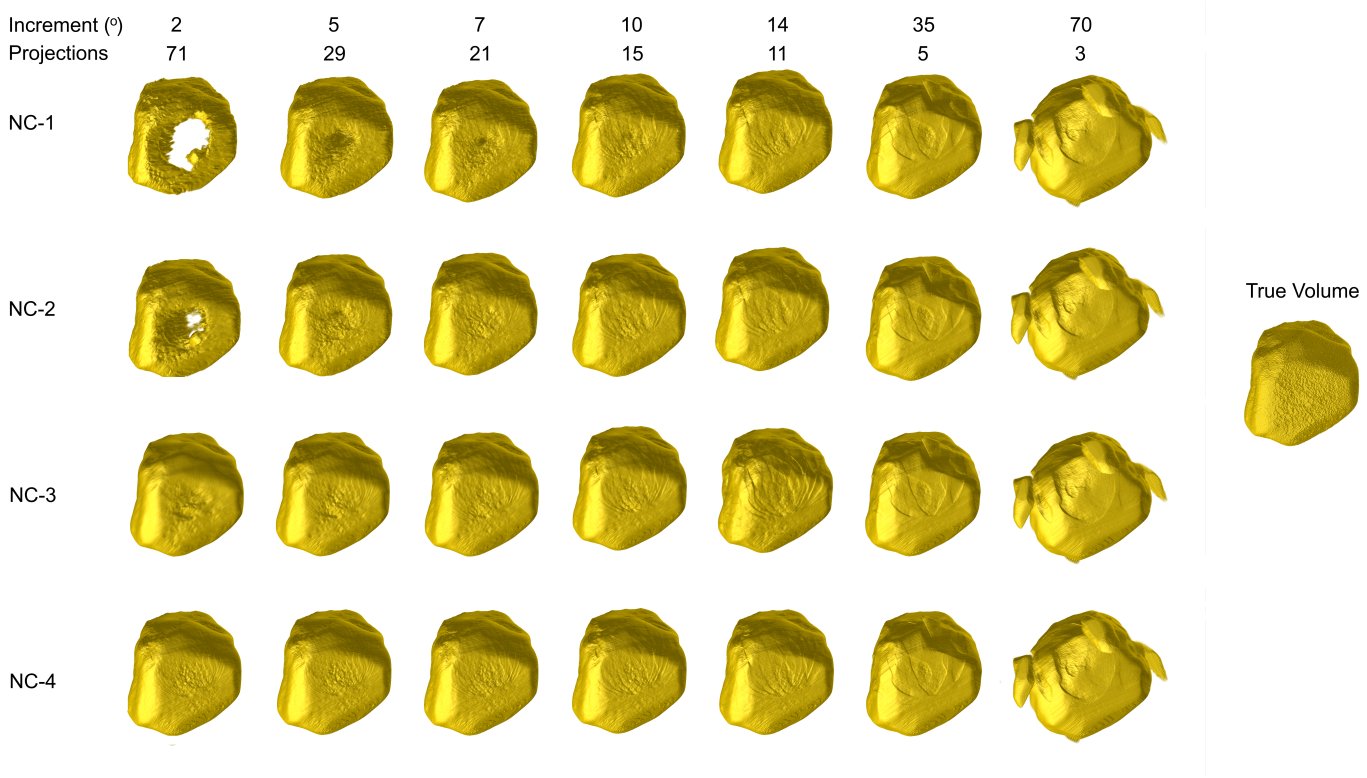

Fig. B.3 3D reconstruction of NC-1 to NC-4 collected using IS acquisition with a variable tilt increment.

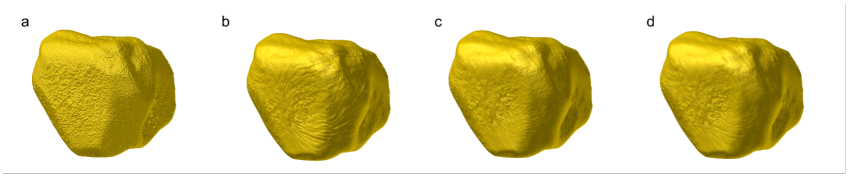

Fig. B.4 3D reference structure (a) along with NC-1 collected with 22 (b), 55 (c) and 71 (d) projections.

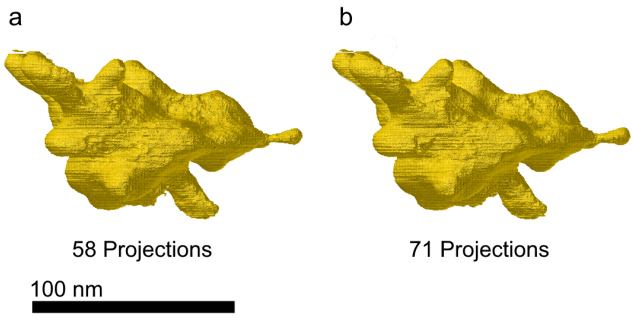

Fig. B.5 3D reconstruction of a Au/Pd nanostar acquired with GRS using 59 (a) and 71 projections (b).

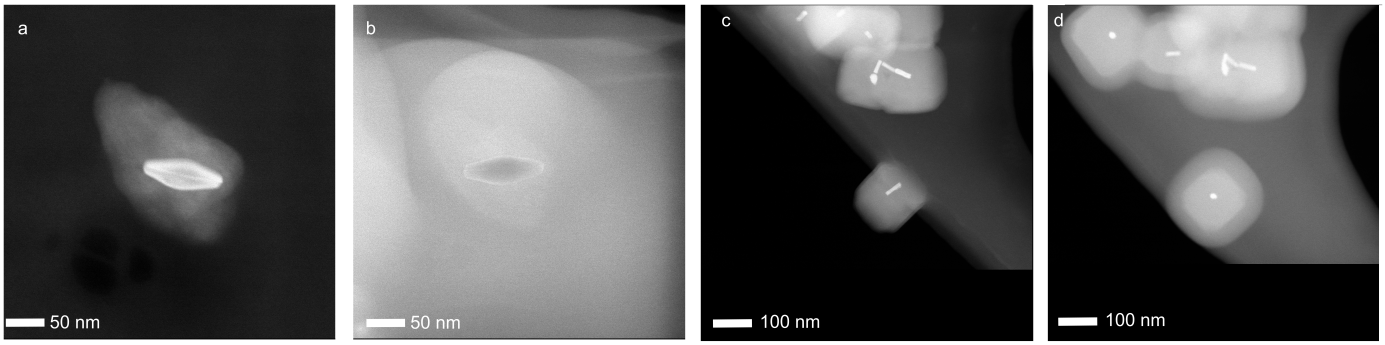

Fig. B.6 Electron microscopy image of Au@NU-100 collected before (a) and after (b) GRS acquisition along with before (c) and after (d) images collected for the GRS acquisition of Au/Pd@ZIF-8.

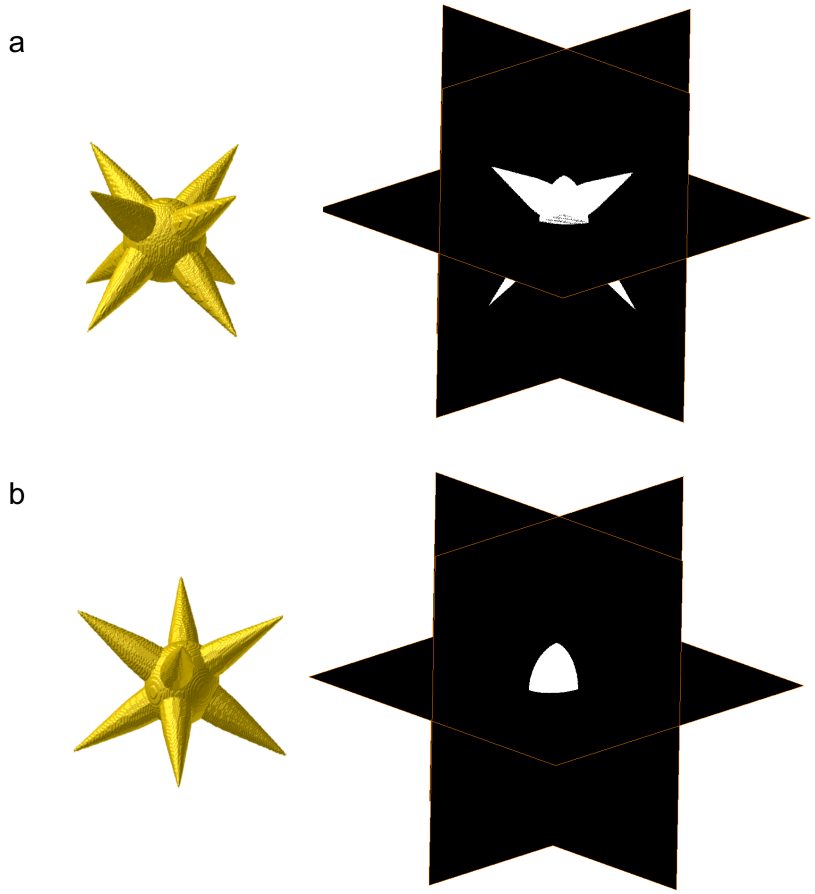

Fig. B.7 Nanostar volume and xy, yz, and xz orthoslices (a) prior to a 45° rotation around the y-axis (b). Dendrites that were apparent in the structure before rotation are no longer visible.

Notes and references

1 J.-W. Buurlage, H. Kohr, W. J. Palenstijn and K. J. Batenburg, *Measurement Science and Technology*, 2018, **29**, 064005.

2 J. D. Hunter, *Computing in science & engineering*, 2007, **9**, 90–95.

3 G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, 2009.