ETSpy: A HyperSpy extension package for electron tomography data processing and reconstruction

Andrew A. Herzing [a],[*],[1], Joshua A. Taillon [b],[2]

[a] Material Measurement Laboratory, National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD 20899, USA
[b] Material Measurement Laboratory, National Institute of Standards and Technology, 325 S Broadway, Boulder, CO 80305, USA

ARTICLE INFO

ABSTRACT

Electron tomography is a powerful tool for the three-dimensional characterization of materials at the nano- and atomic-scales. A typical workflow for tomography involves several pre-processing steps that may include spatial binning, image registration, and tilt-axis alignment depending upon the nature of the acquired data. Here we describe the capabilities of a new, open-source software package named ETSpy that builds upon the widely used HyperSpy package. The basic usage of the software and some specific applications are presented.

1. Introduction

Electron tomography (ET) is a process whereby a three-dimensional (3D) representation of an object is reconstructed from a series of two-dimensional (2D) projection images in the transmission electron microscope (Rosier and Klug, 1968; Crowther and DeRosier, 1970; Frank, 2006). ET has proven to be a valuable technique in both biological (McEwen and Marko, 2001; Lučić et al., 2005; Downing et al., 2007; Gan and Jensen, 2012; Turk and Baumeister, 2020; Young and Villa, 2023) and materials sciences (Midgley and Weyland, 2003; Weyland and Midgley, 2004; Ziese et al., 2004; Kübel, et al., 2005; Midgley and Dunin-Borkowski, 2009; Muller and Ercius, 2009; Grenier et al., 2014). Recent advances have even pushed the spatial resolution to the atomic-scale (Scott et al., 2012; Chen et al., 2013; Xu et al., 2015; Miao et al., 2016; Yang et al., 2017; Zhang et al.,2018; Wang et al., 2020a,b; Yang et al., 2021; Pelz et al., 2022) and have integrated spectroscopic methods such as energy dispersive x-ray (EDX) and electron energy-loss (EELS) spectroscopies with the more traditional brightfield or annular dark-field imaging techniques (Möbus et al., 2003; Gass et al., 2006; Yaguchi et al., 2008; Genc et al., 2013; Goris et al., 2013; Bals et al., 2014; Haberfehlner et al., 2014; Pfannmöller et al., 2015; Goris et al., 2016; Zanaga et al., 2016; Zhong et al., 2017, 2018; Bender et al., 2019; Baumann et al., 2020; Han et al., 2021). Further improvements in reconstruction resolution and quality have been demonstrated using a multi-modal approach combining imaging and spectral imaging during the tomographic acqusition (Schwartz et al., 2022b; Schwartz et al., 2024).

Though the technique is quite flexible and has many modalities, the basic concept remains the same throughout. In ET, we seek to recover an image of an object, $f_x(y,z)$, from a series of projections collected over a range of angles, $P_x(\theta,y)$. This can be accomplished in a number of ways, but often begins with the inverse Radon transform (R^T)(Radon, 1917) such that:

$$\hat{f}_n(y,z) = R^T\,(P_n(\theta,y))$$

$$\theta \in \theta_1,\theta_2,\theta_3,\ldots,\theta_k$$

$$0 \le n < n_x \qquad\qquad (1)$$

where:
θ_n: tilt angle for each projection image
k: number of projection images
n_x: number pixels in the horizontal image axis, x
y: vertical image axis

For the purposes of this paper and the software package which it describes, it is assumed that the tilt axis of the dataset is horizontal (*i.e.* parallel to the x direction of the images, see Fig. 1). Note that in Eq. 1, the reconstructed image is given as $\hat{f}$ rather than the true object, f. This is a result of the finite tilt increment between projections, artifacts arising from data processing errors, the noise content of the projection data,

and, in many cases, a hardware/specimen limit on the portion of the full tilt range (0 to π radians) that is accessible.

In practice, the workflow begins with the collection of projection images over a range of specimen tilt orientations. At each step, the stage goniometer is advanced by a given tilt increment, one or more new images are collected, and the data is stored for offline processing. If necessary, the stage and/or beam shift can be adjusted after tilting in order to account for mechanical movement of the specimen during tilting and the image can also be refocused. The entire process of tilting, adjusting shifts, focusing, and collecting data can be fully automated, semi-automated, or fully manual. The degree of user interaction that is necessary is typically set by the spatial resolution requirements of the analysis with higher resolution calling for more intervention.

Next, several pre-processing steps can be performed on as-needed basis. The full dataset may need to be truncated with some images being discarded due to image quality issues or because the specimen had shifted during acquisition. Often, the dataset will be down-sampled in the spatial domain both to decrease computational requirements in the subsequent steps or to reduce noise. Image filters may also be applied during pre-processing in order to remove noise or image artifacts (*e.g.* cosmic rays, camera defects, etc.). Prior to reconstruction, the projections must be spatially aligned to a common coordinate system and tilt axis. The former involves the calculation and application of a transformation matrix for each projection in the series, while the latter is accomplished via a global rotation and/or shift of the entire image stack. Failure to either properly register the individual projections to each other or to align the stack to the tilt axis will result in artifacts of varying severity and appearance in the reconstructed volume.

Once pre-processing is complete, the stack is provided as input to a reconstruction algorithm. The goal of the reconstruction process is to convert the input slices along x or the sinograms ($P_x(\theta,y)$) from the tilt series to two-dimensional images of the original object ($f_x(y,z)$). These algorithms fit into two broad classes: analytical methods and iterative methods. The most commonly used analytical method is filtered-backprojection (FBP) which applies a weighting filter to the projections prior to calculating the inverse Radon transform of the projection data (Ramachandran and Lakshminarayanan, 1971). The filter is designed to counteract the blurring effect which results from the finite

tilt increment used in acquiring the tilt series data and the nonuniform sampling of the frequency domain this produces. Iterative methods seek to improve upon an initial reconstruction by re-projecting it and comparing to the actual projection data. The two most widely used iterative methods are the simultaneous iterative reconstruction technique (SIRT, (Gilbert, 1972)) and the simultaneous algebraic reconstruction technique (SART, (Andersen and Kak, 1984)). In both cases, the goal is to minimize the difference between the measured projection data and the reprojection of the current reconstruction. In SIRT, the update step is carried out on the full reconstructed image by forward projecting it and comparing to the full input projection data. By contrast, in SART the update step is performed sequentially by forward projecting the current reconstruction estimate for each projection angle and comparing to the input projection data for just that angle. Many other reconstruction methods have been developed more recently for specific applications. These include (among others): (*i*) the discrete algebraic reconstruction technique (DART, (Batenburg and Sijbers, 2011)) where the intensity levels in the reconstructed image are iteratively assigned to one of several distinct grayscale values provided by the analyst, (*ii*) total-variation minimization (TVmin, (Rudin et al., 1992)) which is an iterative method that includes a regularization step to promote smoothness within distinct regions of the image, and (*iii*) compressed sensing electron tomography (CSET, (Saghi et al., 2011)) which produces high-fidelity reconstructions while remaining robust to under-sampling conditions in both the spatial and tilt-axis domains.

Finally, the reconstructed volume must be analyzed in some fashion depending on the nature of the features or properties being measured. In some cases, simply visualizing the 3D volume is sufficient to reveal the necessary detail and isosurface generation or volumetric rendering can meet these needs. More often, some features such as interfacial area or the volume fraction of a given phase needs to be measured. To do this, the reconstructed data is typically segmented so each voxel of the tomogram is assigned a discrete label, allowing for the quantification of various features.

Many software packages, both commercial and open-source, are available for handling some or all of these steps; from data acquisition to preprocessing and on to alignment and eventually reconstruction. It is beyond the scope of this paper to discuss all of these, but, limiting our
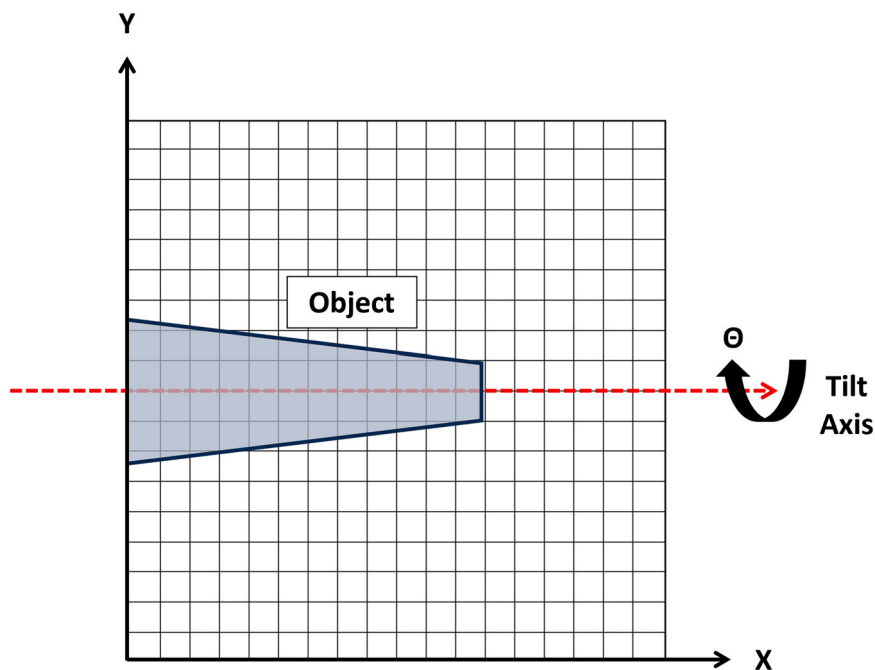

Fig. 1. Diagram of the axes convention used in this paper for electron tomography data. The object is rotated about the tilt axis which is parallel to the horizontal axis of the image.

focus to just the non-commercial open source options, the most widely used package for data acquisition is SerialEM (Mastronarde, 2005, 2024b) which can be paired with the IMOD package (Kremer et al., 1996; Mastronarde, 2024a) for preprocessing, alignment, and reconstruction. Another widely used open-source tool for tomography data processing and visualization is tomviz (Schwartz et al., 2022a; tomviz, 2024) which offers a fully developed UI with intuitive workflow options for all stages of the tomography pipeline. Another well-developed open-source option is TomoPy (Gürsoy et al., 2014) which originated for X-ray based tomography work but is readily adapted to ET as well. Finally, a number of plugins for ImageJ, too many to list here, are available, each of which handles some portion of the tomography workflow.

In this paper, we describe the ETSpy package (Herzing, 2024) which is an ET-focused extension of HyperSpy (de la Peña, et al., 2017; de la Peña et al., 2024), a popular open-source, Python-based package for the processing of multidimensional data, including electron microscopy and spectroscopy data. ETSpy offers a compact, command line driven interface for pre-processing, alignment, reconstruction, and basic 2D visualization of ET data. Additionally, when used through the Jupyter Notebook or Jupyter Lab interfaces (Beg et al., 2021), ETSpy promotes scientific reproducibility through automatic documentation of all processing steps used to achieve a result. These Jupyter documents can also be easily annotated and shared with collaborators or included alongside any resulting publication. As a HyperSpy extension, ETSpy offers a number of attractive options in terms of the ET workflow. Since the parent package already offers high quality widgets for plotting and visualization of higher dimensional datasets and has embedded a large number of relevant data processing methods, ETSpy can draw on these existing capabilities and the efforts of the large network of active HyperSpy users and developers. In this way, ETSpy joins the growing number of HyperSpy extensions available to analysts all over the world, which as of this publication includes tools for X-ray energy dispersive and electron energy loss spectroscopies (*eXSpy*, de la Peña et al., 2024), diffraction and 4D-STEM (*pyxem*, Johnstone et al., 2024), EBSD (*kikuchipy*, Ånes et al., 2024), cathodoluminescence (*lumiSpy*, Lähnemann et al., 2023), atomic resolution imaging (*Atomap*, Nord et al., 2017), holography (*holoSpy*, Prestat et al., 2024a), and particle analysis (*ParticleSpy*, Slater et al., 2021).

2. Test data

Two test datasets are bundled with ETSpy. These include a simulated tilt series of a catalyst particle consisting of a sphere with small particles of varying size present on the surface and an experimental STEM-HAADF image series collected from a needle-shaped sample from NIST SRM-2135c (NIST, 1999). The latter consists of a silicon substrate with alternating layers of nickel and chromium deposited on the free surface. These datasets can be loaded using the following code:

```
import etspy.api as etspy
from etspy import datasets as ds
s =ds.get_needle_data()  # loads experimental test
data
s2 =ds.get_catalyst_data()   # loads simulated cata-
lyst data
```

The experimental data is provided in two versions: one which has not been aligned (as it was collected) and another that has already been aligned. The aligned parameter of ds.get_needle_data() controls the version that is loaded (the default is non-aligned):

```
s = ds.get_needle_data()
s_aligned = ds.get_needle_data(aligned=True)
```

The model catalyst data (accessed via ds.get_catalyst_data()) is fully aligned but misalignment can be applied as the data is read into memory in order to simulate real experimental data. The provided Jupyter notebook uses these datasets to demonstrate the rest of the ETSpy functionality.

3. Signal classes

Two new Python classes are defined in ETSpy, each of which is a child of HyperSpy's Signal2D parent class. The first of these new classes is the TomoStack, which contains the tilt series data, metadata, and all of the methods used for data manipulation, alignment, and reconstruction. As in the Signal2D parent class, the TomoStack has navigation axes and signal axes. For an ET image tilt series, the signal axes will be the x and y image axes, while the navigation axis will be the θ tilt angle dimension. The data can be visualized readily using the HyperSpy plot functionality and, when using the Jupyter widget backend for matplotlib (Hunter, 2007), the stack can be interactively viewed in the Jupyter interface as a function of tilt angle.

Relevant metadata for tomographic processing is included in a new node of the standard HyperSpy metadata structure. This metadata can be accessed from the Jupyter command line by typing:

```
s.metadata.Tomography
```

and it includes the projection angles, a boolean indicator as to whether or not the data has been previously cropped, the angle of the tilt axis relative to the horizontal axis of the images, and any applied global shifts. The projection angles for the tilt series and the translational shifts of each image are stored as attributes of the TomoStack class as HyperSpy signals. They can be directly accessed as s.tilts and s.shifts, respectively. To view the underlying data values for these signals, the following commands can be used:

```
s.tilts.data
s.shifts.data
```

There is an important distinction between the projection angles stored in s.tilts and those referenced in the HyperSpy AxesManager: The values stored in s.tilts are the "true" values used for several alignment processes as well as the reconstruction. The values stored in the HyperSpy AxesManager (which assume a uniform scale between tilts) are used for visualization purposes and should be referenced with caution, as they often may differ from the actual projection angles stored in s.tilts. This distinction allows for arbitrary projections to be removed from a TomoStack via use of the s.remove_projections() method, which is not possible with HyperSpy's standard axes-handling tools. Future versions of ETSpy may make use of HyperSpy's "non-uniform data axis" feature, where the axis values are not evenly spaced, but HyperSpy's current implementation imposes limitations on other functionality needed for tomographic analysis (such as spatial binning). At the present state, ETSpy will continue to store and use the projection angles in the s.tilts attribute.

In addition to the TomoStack, there is a second class called a RecStack which is used to handle the reconstructed data. At the time of publication, this class mostly provides a way to distinguish between objects containing tilt series data and those containing reconstructed data. It also includes a basic 3D slice plotting function to view images of reconstructed data along the Z-X, Y-X, and Z-Y directions. Additional functionality will be added to this class in the future.

4. Data I/O

ETSpy contains a load function for reading data into memory. It can be used in the following ways:

```
s = etspy.load("filename.ext")
image_series = ["image1.ext", "image2.ext", "image3.
ext"]
s2 = etspy.load(image_series)
```

Hyperspy (de la Peña *et al.*, 2024) and the associated RosettaSciIO package (Prestat et al., 2024b) already offer robust data reading capabilities for a wide array of ET relevant file formats. ETSpy relies on this functionality to access image data and the required metadata for several formats such as Gatan Digital Micrograph (DM3 or DM4),[†] HDF5 (including the Hyperspy specific HSPY derivative), and MRC. When loaded, HyperSpy reader functions handle the calibration data and ETSpy collects the projection angle data for inclusion with the tilts attribute. Additional ETSpy code is included for handling sets of individual image series in either DM3/DM4 format or MRC/MDOC pairs generated using SerialEM. Finally, if a tilt series already resides in memory as a NumPy array or a HyperSpy Signal2D, it can be directly converted to a TomoStack in the following way (using random data as an example):

```
import etspy.api as etspy
import hyperspy.api as hs
import numpy as np
tilts = np.arange(0,180,2)
s = np.random.random([90,100,100])
s = etspy.TomoStack(s, tilts)
s = hs.signals.Signal2D(np.random.random([90,100,
100]))
s = etspy.TomoStack(s, tilts)
```

Any TomoStack or RecStack can be saved at any point during data processing and analysis. This is done by using the save method and is best accomplished using the HyperSpy hspy format which is a custom version of the HDF5 standard format. When using the hspy format, the data and all of the metadata is stored and easily accessed in the future using the etspy.load() method.

5. Basic visualization

Both the TomoStack and RecStack classes inherit the plotting functionality of the HyperSpy Signal2D class. For example, the code

```
from etspy import datasets as ds
s = ds.get_catalyst_data()
s.plot()
```

will load the test catalyst data and plot the first projection image in the stack. When used interactively with ipywidgets a slice navigator can be used to navigate through the stack in the tilt dimension and update the display in real time. By employing the swap_axes() method, navigation can also be done over the x-axis which will show the sinogram view. Finally, the TomoStack objects can be used as input to all other plotting functionality of Hyperspy as well. For example, to compactly plot a projection image alongside a sinogram, the following code can be used:

```
import hyperspy.api as hs
hs.plot.plot_images([s.inav[0], s.extract_sinogram
(128)],
```

```
colorbar=None)
```

which produces the output shown in Fig. 2.

6. Pre-processing

Spatial sub-sampling can be performed by slicing the signal axes as described in the HyperSpy documentation. In the first line of the code shown below, the result will be to remove the first 5 and last 10 columns and the first 15 and last 20 rows from the input tilt series:

```
s = ds.get_needle_data()
print(s)
<TomoStack, title: , dimensions: (77|256, 256)>
s_cropped = s.isig[5:-10, 10:-5]
print(s_cropped)
<TomoStack, title: , dimensions: (77|241, 241)>
```

The tilt axis can be similarly sliced using the navigation axis of the TomoStack, and the tilts and shifts attributes will be sliced to match:

```
# each ofthe following has a navigation size of 77
pixels:
print(s)
<TomoStack, title: , dimensions: (77|256, 256)>
print(s.tilts)
<TomoTilts, title: Image tilt values, dimensions:
(77|1)>
print(s.shifts)
<TomoShifts, title: Image shift values, dimensions:
(77|2)>
s_cropped = s.inav[5: -10]
# afterslicing, each of the following has a navigation
size of 62 # pixels:
print(s_cropped)
<TomoStack, title: , dimensions: (62|256, 256)>
print(s_cropped.tilts)
<TomoTilts, title: Image tilt values, dimensions:
(62|1)>
print(s_cropped.shifts)
<TomoShifts, title: Image shift values, dimensions:
(62|2)>
```

ETSpy also supports removing projections images from the Tomo-Stack in a less uniform way. For example, projections may suffer from irrecoverable alignment or focus issues or otherwise contain artifacts that would degrade the reconstruction quality if included. In that case, a dedicated method named remove_projections() for the TomoStack class has been provided for removal of poor-quality projections. For example, imagine that the tenth projection was severely out of focus while in the twentieth projection the sample had drifted too far out of the field of view. The stack can be corrected by removing these images in the following way:

```
print(s)
<TomoStack, title: , dimensions: (77|256, 256)>
s_new = s.remove_projections([10, 20])
print(s_new)
<TomoStack, title: , dimensions: (75|256, 256)>
print(s_new.tilts)
<TomoTilts, title: Image tilt values, dimensions:
(75|1)>
print(s_new.shifts)
<TomoShifts, title: Image shift values, dimensions:
(75|2)>
```
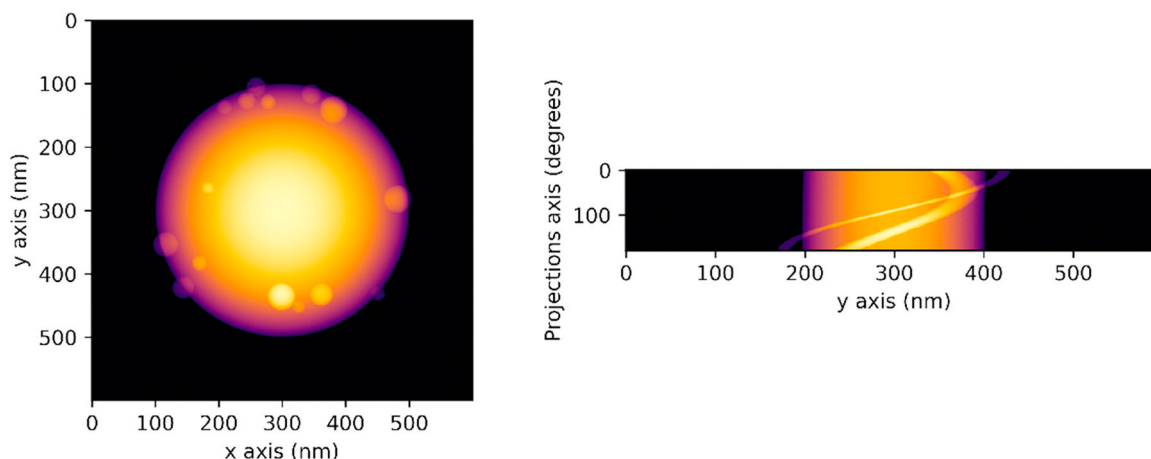
Fig. 2. Plotting functionality of the TomoStack class shown for the simulated catalyst tilt series. The central projection (left) is shown alongside the central sinogram slice (right).

In both cases, the relevant tilt angles are also removed from the tilts and shifts attributes to ensure the relationship between the projection images and the projection angles/shifts is correct. As a word of caution, after using the remove_projections() method, the navigation axis for the signal (s_new.axes_manager["Projections"]) will no longer be synchronized with the actual tilt values stored in s_new.tilts, due to the limitations described in Section 3.

Filtering of the stack can also be performed using the filter() method of the TomoStack class. Currently, Sobel, median, and band-pass filters are implemented along with a combined median and Sobel option. Additional filters can be easily added or applied directly to the underlying data array of the TomoStack. This can be useful for improving the accuracy of the stack alignment methods. Translational shifts can be calculated on the filtered stack and applied to the unfiltered stack using the align_other() method, as discussed below.

7. Image registration

Multiple options exist for calculating and applying translational shifts for aligning the images of the stack using the register_stack() method of the TomoStack class. The available options are:

- Phase correlation ("PC")
- StackReg ("StackReg")
- Center of Mass ("COM")
- Combined center of mass and common line ("COM-CL")

Phase correlation (PC) (Guizar-Sicairos et al., 2008) analysis, a normalized form of Fourier cross-correlation, is carried out using the *scikit-image* (Walt et al., 2014) implementation. The StackReg option uses the Python implementation (Lichtner, 2023) of an algorithm first described by Thévenaz et al. (1998) and originally implemented in *ImageJ* (Thévenaz, 2011). In this method, the transformation is iteratively calculated by minimization of a cost function based on the sum squared difference of intensities between two images. The COM option employs a Python version of a set of algorithms developed by Sanders et al. (2015) which were first implemented in *Matlab*. The method traces center of mass of the object at several locations along the tilt axis and minimizes the error between the measurements and the path expected for such an object over the set of projection angles that were used for data collection. Finally, the *COM-CL* option employs a two-step alignment first described by Scott et al. (2012). In this case, shifts are first calculated in the direction perpendicular to the tilt axis by center of mass tracking. Then, the shifts parallel to the tilt axis are determined via the common-line method which involves projecting the image along the direction perpendicular to the tilt axis and serially cross-correlating the

resulting profiles. All of these methods have advantages and disadvantages. PC is straightforward and fast, although it can be challenged by noisy data, by samples which exhibit regularly repeating features, or when the differences in image features between two images is large (*e.g.* when a large tilt increment is used). StackReg will often outperform PC in these cases but is computationally more expensive. Finally, both COM and COM-CL incorporate the geometry of the tilt series acquisition and are very effective for particulate or pillar shaped samples since little or no additional material enters the field of view during tilting in these cases. However, they will often fail when slab type specimen geometries are used or when significant amounts of extraneous material enters and leaves the field of view over the course of the series. Further detail about all of these methods can be found in the provided citations.

Regardless of the method chosen, translation shifts are calculated between each set of images in the stack. The shifts are then composed such that they are made relative to the previous shift in the stack starting from a user defined projection number. If the user does not define this starting projection number, the midpoint of the stack is used.

Finally, the shifts can be calculated using any of these methods and then applied to another stack using the align_other() method. This approach is useful when multiple image signals are collected during a single tilt series acquisition and when one of the signals has superior characteristics for calculating the alignment shifts compared to another. For example, shifts can be calculated on a high signal-to-noise ratio medium-angle annular dark field (MAADF) image stack and applied to a noisier high-angle annular dark field (HAADF) stack. Since both signals are acquired simultaneously, the shifts for each will be the same. This can also be used to calculate shifts using an image signal collected alongside a hyperspectral EDX or EELS dataset and then apply those shifts to the spectral images extracted from the datacube (see Section 10 for more detail).

8. Tilt-axis alignment

In order ensure that the tilt axis of the dataset is centered and made as close to horizontal as possible prior to reconstruction, a global shift and rotation of the stack can also be calculated and applied using the tilt align method. Two options are available in this case:

- Center of mass tracking ("COM")
- Maximum projection image analysis ("MaxImage")

The first of these is based on a method described by Wolf (2012) and involves tracking the center of mass at multiple locations along the tilt axis and fitting these to the motion expected for an ideal cylinder. This method works very well for pillar- or needle-shaped specimen
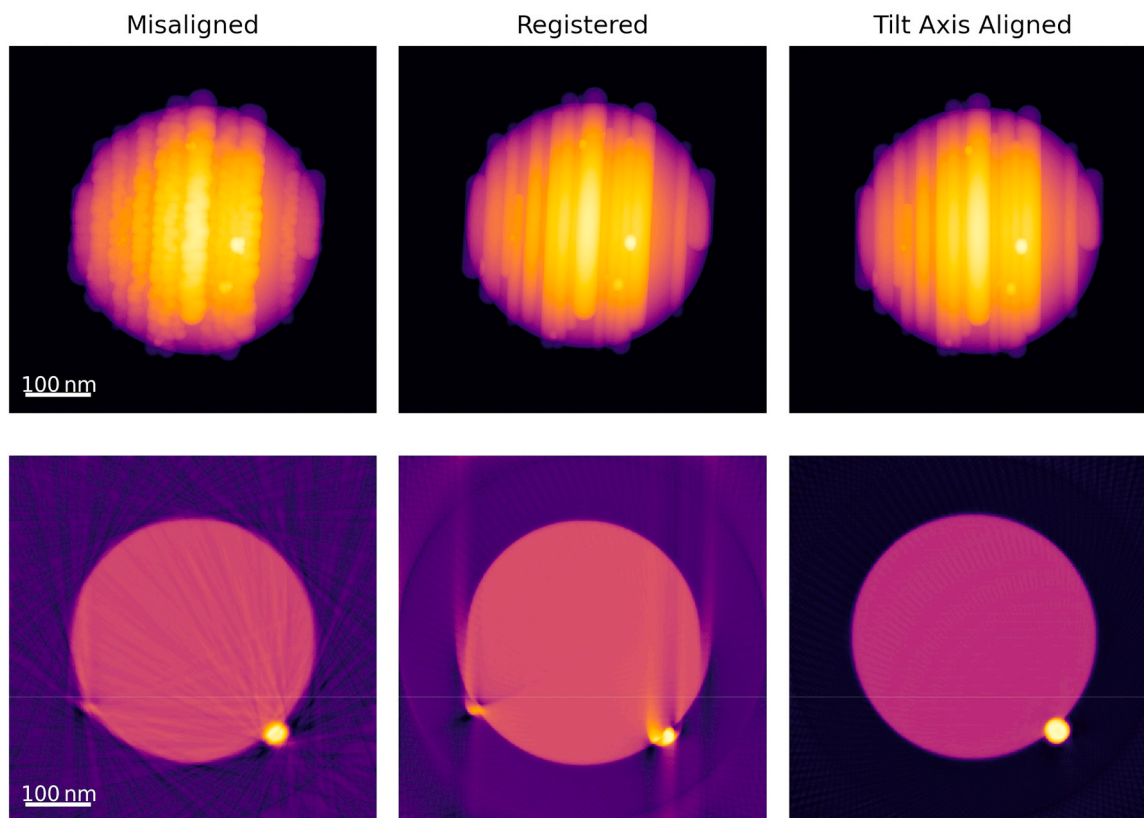
Fig. 3. Alignment results. Maximum projection images of the tilt series (top series) are shown for the misaligned stack, the registered stack, and the fully aligned stack along with the resulting reconstructions (SIRT, 100 iterations) from each (bottom row).

geometries. Alternatively, the MaxImage method analyzes features in the projected maximum image of a spatially registered stack using a combination of edge-detection and Hough transform analysis in order to determine the tilt axis rotation. This method is particularly effective when particles are present in the sample which trace linear paths in the projected maximum image. Optionally, the global shift of the tilt axis can also be calculated by minimization of the sum of the reconstruction. In both cases, the measured shift and rotation are stored in the Tomography node of the metadata structure and can be applied to other stacks using the align_other() method described in the previous section.

An example of the full, automated alignment process is illustrated in Fig. 3 which shows the maximum projection images from the simulated catalyst tilt series prior to alignment, after stack registration via "StackReg" and after a tilt axis alignment via the "MaxImage" method. The reconstruction at each stage is also shown with varying artifacts due to misalignment.

Finally, where the automated approaches fail for a particular dataset, a manual approach is also available. This is done using the test_align() method which displays three reconstructed sinograms extracted from user-defined locations in the stack. If the user does not define these locations, positions at one quarter, one half, and three quarters of the stack width are chosen. By visual inspection of the features in these reconstructions, it is often possible to iteratively determine the best shift and tilt and tilt values by altering one or both parameters and re-executing test_align() multiple times.

9. Reconstruction

There are four reconstruction algorithms currently available in ETSpy and each relies on the ASTRA toolbox (Aarle et al., 2015) to perform the reconstruction. The available options are:

- Simple backprojection ("BP")
- Filtered backprojection ("FBP", default)
- Simultaneous iterative reconstruction technique ("SIRT")
- Simultaneous algebraic reconstruction technique ("SART")
- Discrete algebraic reconstruction technique ("DART")

Reconstruction is initiated by calling the reconstruct() method of the TomoStack. The ASTRA toolbox offers CPU-based and CUDA-based GPU-accelerated options for the BP, FBP, SIRT, and SART methods. ETSpy will attempt to determine whether a CUDA-compatible GPU is available and use it for faster reconstruction when possible. Alternatively, the reconstruction will be carried out using parallel computation on the CPU using the Python multiprocessing package. The DART implementation employed in ETSpy is based on custom Python code and no CUDA-acceleration is available at the present time. The reconstruction returns a RecStack object which contains the 3D reconstructed image data.

For the iterative SIRT and SART reconstruction methods, another function (recon_error()) is provided to perform a reconstruction on a single slice of the tilt series data and return a series of reconstructions showing the progress at each iteration along with a measure of the error at each step. This can be useful in determining the number of iterations to perform on the entire stack. Once completed, the resulting stack can be interactively plotted to see how the reconstructed image changes with iteration. The error is returned as a HyperSpy Signal2D array which can be interrogated numerically or graphically. For CUDA-capable systems, ASTRA offers a way to do this directly using astra.algorithm. get_res_norm(). For systems without CUDA, ETSpy will use the numpy. linalg.norm() function to calculate the error between the sinogram and the reconstruction at each iteration. An example of this process is shown in Fig. 4.
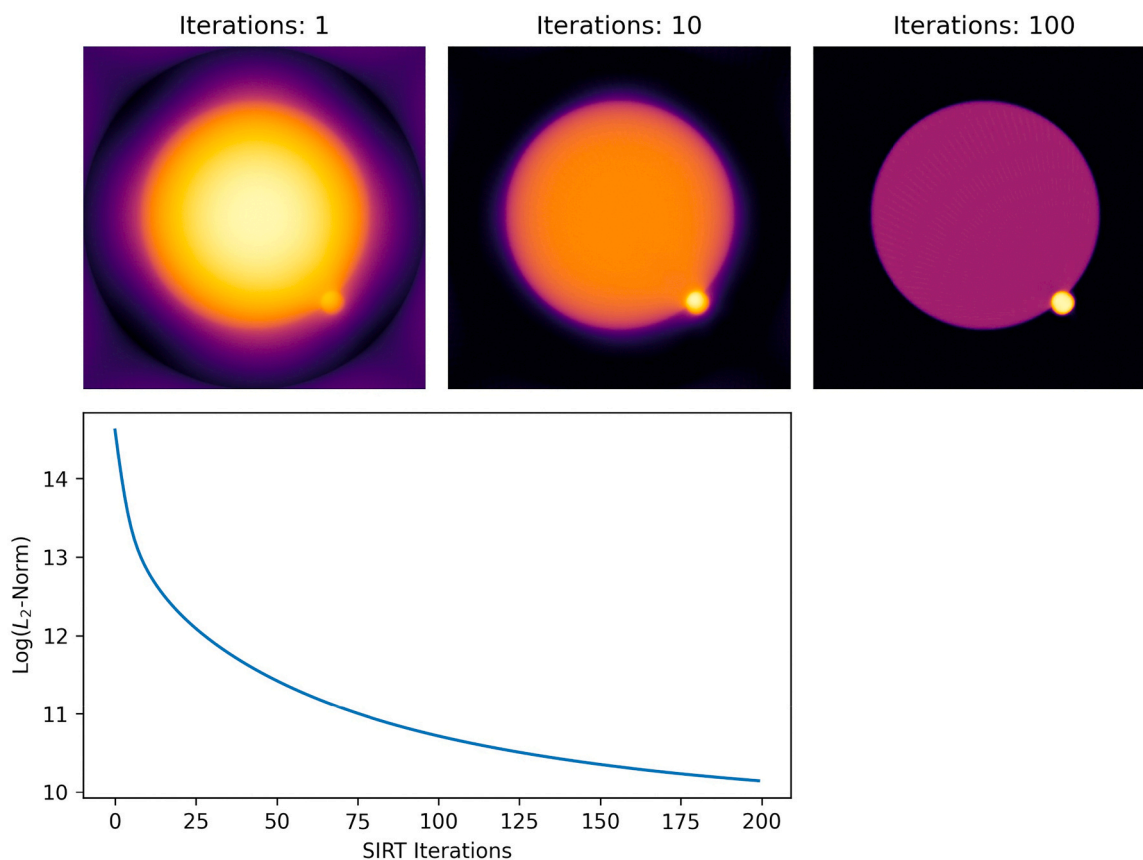
Fig. 4. Iterative reconstruction convergence analysis. Reconstructed slice from the simulated catalyst tilt series shown after an increasing number of SIRT iterations (top row). Also shown is the L_2-norm of the difference between the reconstruction and the input projection data (log scale) as a function of SIRT iteration (bottom).

10. Hyperspectral tomography

To demonstrate the use of ETSpy for processing hyperspectral tomography, data was collected from another needle-shaped sample of the NIST SRM-2135c depth-profiling standard. Several EDX spectrum images (SIs) were collected every 5° over a 180° tilt range using an on-axis style tomography holder (Fischione Model 2050). The data were collected using an EDAX Octane-T silicon drift detector and the acquisition was controlled by the EDAX TEAM software. The SIs were 162 x 128 pixels in size with 3 nm square pixels. The probe current was approximately 0.5 nA and the per pixel dwell time was 200 ms. At each new tilt, the sample was manually aligned to an image at the previous tilt using a combination of beam and stage shifting. The sum spectrum for the entire spectral tomography dataset is shown along with the sum image calculated from a single projection angle in Fig. 5.

Once acquired, the TEAM SPD files were converted to HDF5 format using HyperSpy. All that remains to utilize ETSpy for reconstructing the chemical maps is to extract the required 2D elemental maps at each tilt and converting each of the resulting HyperSpy Signal2D instances to a TomoStack. Using HyperSpy, this can be done either via straightforward background subtraction methods, curve fitting, or via machine learning algorithms. Examples of background subtraction and the non-negative matrix factorization (NMF) machine learning algorithm (Paatero and Tapper, 1994) are shown in the following code:

```
import hyperspy.api as hs
import etspy.api as etspy
# Load EDX datasets and define tilts
edx  =  hs.load("EDX_SI_*.hdf5",  reader="HSPY",
stack=True)
tilts = np.linspace(0, 180, 37)
```

```
## Background Subtraction Method
# Define lines to use and background windows
lines = ["Si_Ka", "Ni_Ka", "Cr_Ka","Pt_La"]
bckg = [[1.2,1.4,3.2,3.4],[4.5,4.8,11.9,12.5],
        [4.5,4.8,11.9,12.5],[4.5,4.8,11.9,12.5]]
# Extract elemental maps
maps = edx.get_lines_intensity(lines,
background_windows=bckg)
SiKa, NiKa, CrKa, PtLa = maps
# Convert maps to TomoStack
SiKa = etspy.TomoStack(SiKa, tilts)
NiKa = etspy.TomoStack(NiKa, tilts)
CrKa = etspy.TomoStack(CrKa, tilts)
PtLa = etspy.TomoStack(PtLa, tilts)
## NMF decomposition
# Perform initial singular value decomposition
edx.decomposition(True)
# Refine the decomposition using NMF
edx.decomposition(True,
  algorithm="nmf",
  output_dimension=6)
# Extract phase maps and convert to TomoStacks
loadings = edx.get_decomposition_loadings()
Ni_NMF = etspy.TomoStack(loadings.inav[0], tilts)
Si_NMF = etspy.TomoStack(loadings.inav[1], tilts)
Cr_NMF = etspy.TomoStack(loadings.inav[2], tilts)
Pt_NMF = etspy.TomoStack(loadings.inav[3], tilts)
```

Once the TomoStack's have been created using either method, it is then just a matter of following the usual tomographic workflow of stack registration, tilt-axis alignment and reconstruction. Since the elemental
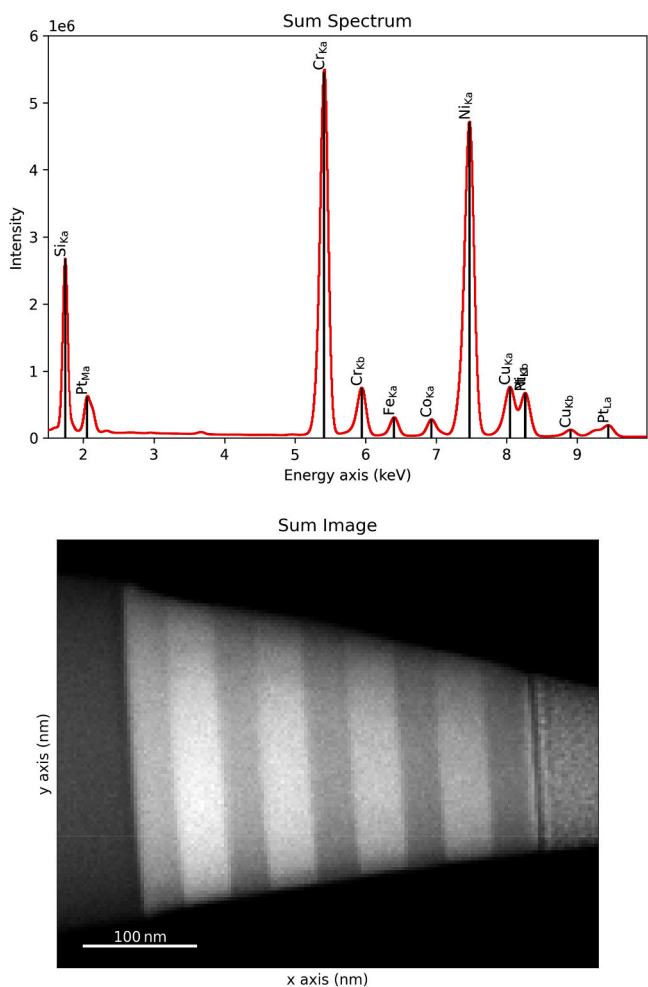
Fig. 5. Sum spectrum of the fully spectral tomography dataset (top) and sum image extracted from a single spectrum image projection (bottom).

maps are often not well suited for calculating alignment transformations, we can either use a tilt series of an image signal collected simultaneously (*e.g.* HAADF, etc.) or by the spectral sum images for alignment followed by using the align_other() method of the TomoStack to apply the calculated alignment to each spectral tilt series. The latter option using the spectral sum image is shown below:

```
## Calculate alignment on the spectral sum image series
# Calculate sum image. This results in a HyperSpy
# Signal2D class:
edx_sum = edx.sum(3).as_signal2D((0,1))
# Convert to a TomoStack
edx_sum = etspy.TomoStack(si_sum, tilts)
edx_reg = edx_sum.stack_register("StackReg")
edx_ali = edx_reg.tilt_align("CoM")
# Apply alignments to the phase maps
Ni_NMF = edx_ali.align_other(Ni_NMF)
Si_NMF = edx_ali.align_other(Si_NMF)
Cr_NMF = edx_ali.align_other(Cr_NMF)
Pt_NMF = edx_ali.align_other(Pt_NMF)
# Reconstruct the datasets
Ni_rec = Ni_NMF.reconstruct()
Si_rec = Si_NMF.reconstruct()
Cr_rec = Cr_NMF.reconstruct()
Pt_rec = Pt_NMF.reconstruct()
```

The component images produced by the NMF decomposition for one projection of the full hyperspectral tomography dataset are shown in Fig. 6. In this case, the first four components show the spatial distribution of the nickel, silicon, chromium, and platinum signals, respectively. Components five and six (not shown), were related to the absorption of the nickel and chromium *L*-lines.

Since the data underlying these images was collected simultaneously with that which is used to generate the sum images described previously, the alignments calculated in the latter case can also be applied here. Finally, the individual NMF image series are reconstructed independently and slices from each are displayed in Fig. 7.

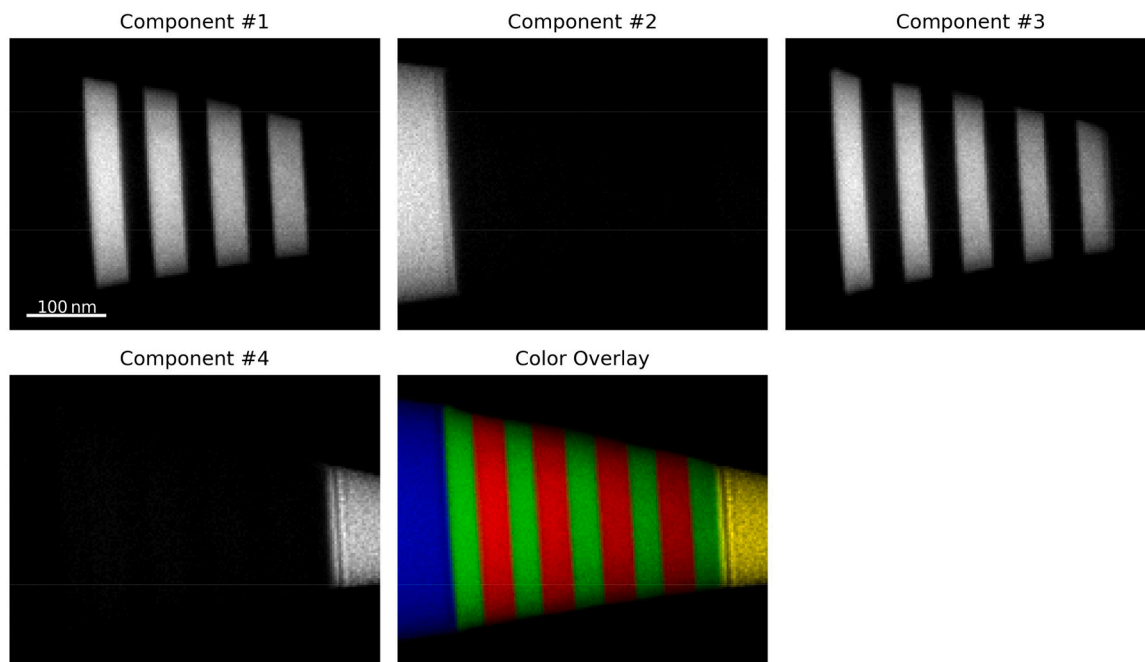At this point, the reconstructions can be quantified using packages

Fig. 6. Results of NMF decomposition of spectral tomography dataset. Components 1 through 4 are shown along with a color overlay emphasizing their spatial extent.
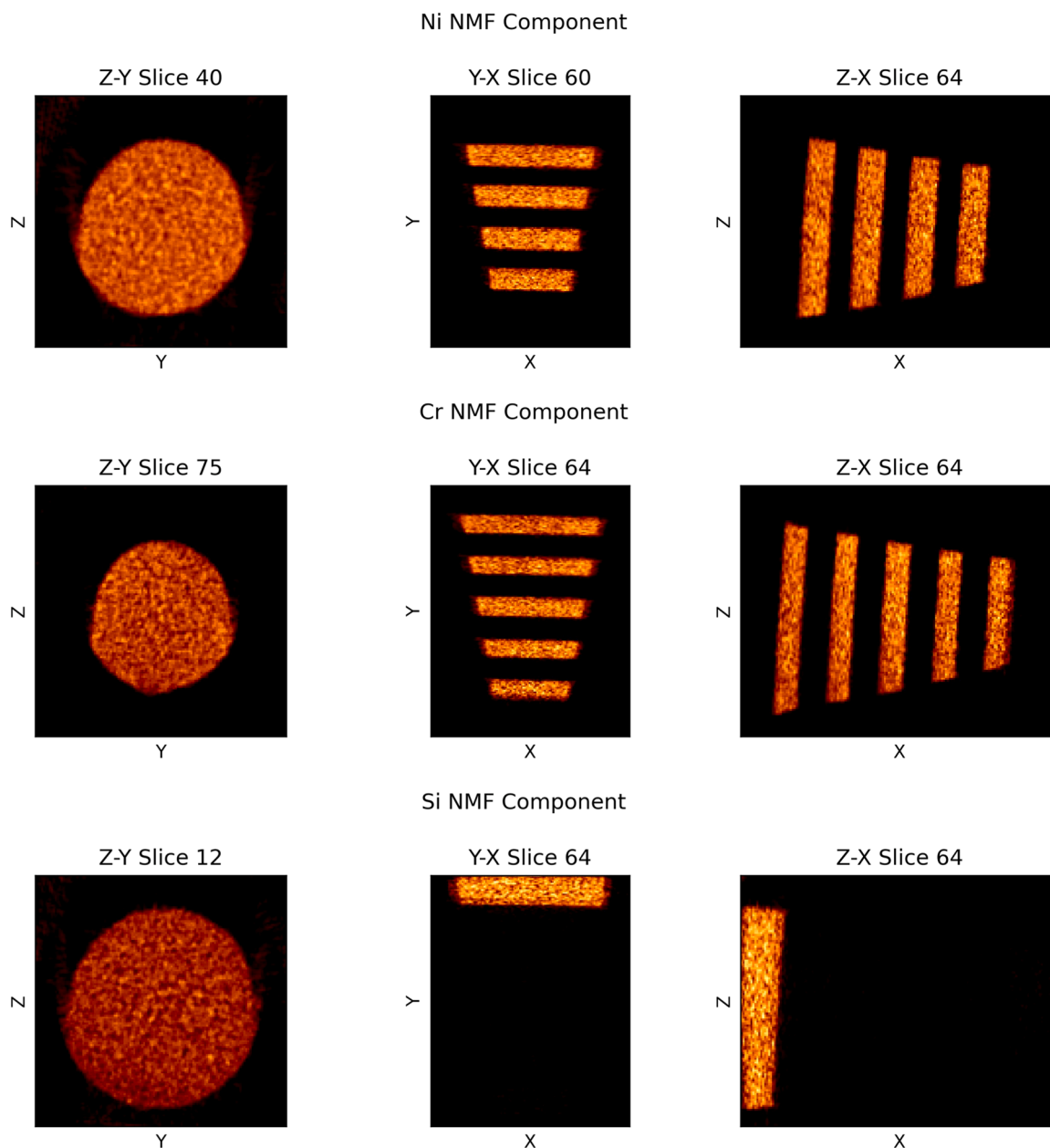
Ni NMF Component

Z-Y Slice 40   Y-X Slice 60   Z-X Slice 64

Cr NMF Component

Z-Y Slice 75   Y-X Slice 64   Z-X Slice 64

Si NMF Component

Z-Y Slice 12   Y-X Slice 64   Z-X Slice 64

Fig. 7. Slices extracted from reconstructions of the NMF components associated with nickel (top), chromium (middle), and silicon (bottom).

such as *NumPy*, *SciPy*, *scikit-image*, etc., all of which can be accessed directly from the same Jupyter Notebook used for ETSpy processing. For example, it is very straightforward to binarize the reconstructions of the nickel and chromium layers and calculate volumetric ratios of each:

```
pixel_size =Ni_rec.axes_manager[0].scale
Ni_binary =Ni_rec.deepcopy()
Ni_binary[Ni_binary>0] = 1
Cr_binary =Cr_rec.deepcopy()
Cr_binary[Cr_binary>0] = 1
Ni_volume = pixel_size**3 * Ni_binary.sum()
Cr_volume = pixel_size**3 * Cr_binary.sum()
volume_ratio = Ni_volume/Cr_volume
```

This is just one simple example to illustrate the way in which tomographic reconstructions generated using ETSpy can be directly interrogated within the Jupyter environment.

Finally, 3D visualization can be carried out in one of two ways. First,

the reconstructions can be saved to disk and then read into dedicated visualization software. Alternatively, for those who wish to remain in the Python environment used for the rest of the data processing, packages such as ipyvolume or mayavi can be used from some basic visualization. An example of using ipyvolume is shown in Fig. 8.

11. Conclusions

In this paper, we have described the ETSpy software package, a new Hyperspy extension that is specifically tailored to enhance the processing and reconstruction of electron tomography data. The package provides a comprehensive suite of tools designed to streamline the workflow from data input to final reconstruction.

The package leverages the already powerful capabilities of its parent package and adds a host of application specific functionality for electron tomography data. This includes data pre-processing, image registration, tilt axis alignment, 3D reconstruction, and basic visualization. Since it is Python-based, the package is easily scriptable and when used within the
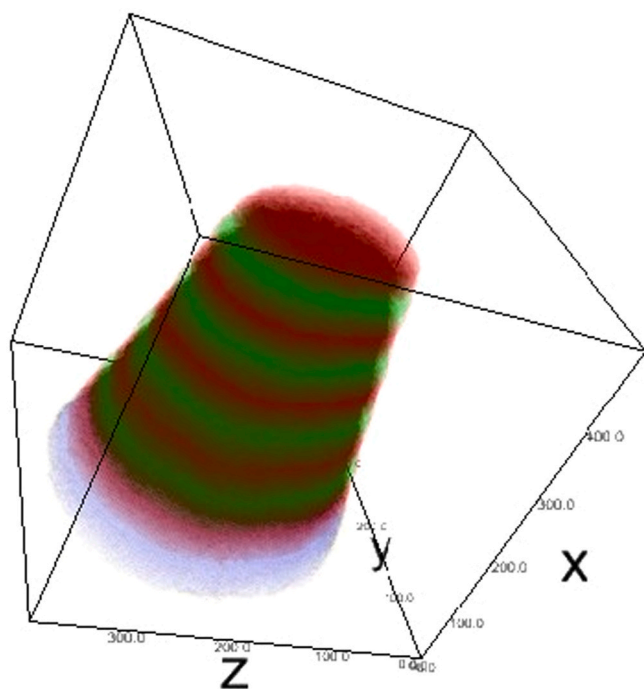
Fig. 8. Volumetric rendering using ipyvolume of the Cr-K$_\alpha$, Ni-K$_\alpha$, and Si-K$_\alpha$ tomographic reconstructions (red, green, and blue, respectively).

Jupyter Lab interface can be thoroughly documented and easily shared. The combination of HyperSpy's core functionality with the new capabilities of ETSpy is most apparent in the processing of hyperspectral tomography datasets since initial tasks such as background subtraction and machine learning decomposition are seamlessly integrated with tomographic alignment and reconstruction.

CRediT authorship contribution statement

Andrew A. Herzing: Conceptualization, Methodology, Project administration, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Joshua A. Taillon:** Methodology, Software, Validation, Writing – review & editing.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andrew Herzing reports financial support was provided by National Institute of Standards and Technology. Joshua Taillon reports financial support was provided by National Institute of Standards and Technology. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

Appendix A. Code and Data Availability

The ETSpy package is hosted on GitHub under the usnistgov organization: https://github.com/usnistgov/ETSpy and can be cited using the following DOI: https://doi.org/10.18434/mds2-3616. Extensive documentation is available at the project's homepage: https://pages.nist.gov/etspy.

In addition to the code itself, a Jupyter notebook is provided to demonstrate the basic functionality described in this paper, which should be available at https://pages.nist.gov/etspy/examples/etspy_demo.html (in the event that link changes, it is also available from the GitHub repository).

The EDX tomography dataset can be accessed at: https://doi.org/10.18434/mds2–3631.

Appendix B. Package Installation

Due to the integration of compiled and GPU-accelerated features, installation is most easily accomplished by using Anaconda (Anaconda Software Distribution, 2024) to create a new environment and to install the package from the conda-forge repository:

```
conda create -n etspy
conda activate etspy
conda install -c conda-forge etspy
```

Data availability

Data will be made available on request.

References

Aarle, Wv, Palenstijn, W.J., Beenhouwer, J.D., Altantzis, T., Bals, S., Batenburg, K.J., Sijbers, J., 2015. The ASTRA toolbox: a platform for advanced algorithm development in electron tomography. Ultramicroscopy 157, 35–47. https://doi.org/10.1016/j.ultramic.2015.05.002.

Anaconda Software Distribution, 2024. URL: ⟨https://docs.anaconda.com/⟩. accessed: 2024-07-11.

Andersen, A.H., Kak, A.C., 1984. Simultaneous algebraic reconstruction technique (SART): a superior implementation of the art algorithm. Ultrason. Imaging 6, 81–94. https://doi.org/10.1016/0161-7346(84)90008-7.

Ånes, H.W., Lervik, L.A.H., Natlandsmyr, O., Bergh, T., Prestat, E., Bugten, A.V., Østvold, E.M., Xu, Z., Francis, C., Nord, M., 2024. Pyxem/kikuchipy: Kikuchipy 0.10.0. Zenodo. doi:10.5281/zenodo.11432173.

Bals, S., Goris, B., Liz-Marzán, L.M., Van Tendeloo, G., 2014. Three-dimensional characterization of noble-metal nanoparticles and their assemblies by electron tomography. Angew. Chem. Int. Ed. 53, 10600–10610. https://doi.org/10.1002/anie.201401059.

Batenburg, K.J., Sijbers, J., 2011. DART: a practical reconstruction algorithm for discrete tomography. IEEE Trans. Image Process. 20, 2542–2553. https://doi.org/10.1109/tip.2011.2131661.

Baumann, F.H., Popielarski, B., Lu, Y., Mitchell, T., 2020. Extension of CD-TEM towards 3D elemental mapping. IEEE Trans. Semicond. Manuf. 33, 346–351. https://doi.org/10.1109/TSM.2020.2990588 (conference Name: IEEE Transactions on Semiconductor Manufacturing).

Beg, M., Taka, J., Kluyver, T., Konovalov, A., Ragan-Kelley, M., Thiéry, N.M., Fangohr, H., 2021. Using jupyter for reproducible scientific workflows. Comput. Sci. Eng. 23, 36–46. https://doi.org/10.1109/MCSE. 2021.3052101.

Bender, H., Richard, O., Kundu, P., Favia, P., Zhong, Z., Palenstijn, W.J., Batenburg, K.J., Wirix, M., Kohr, H., Schoenmakers, R., 2019. Combined STEM-EDS tomography of nanowire structures. Semicond. Sci. Technol. 34, 114002. https://doi.org/10.1088/1361-6641/ab4840.

Chen, C.C., Zhu, C., White, E.R., Chiu, C.Y., Scott, M.C., Regan, B.C., Marks, L.D., Huang, Y., Miao, J., 2013. Three-dimensional imaging of dislocations in a nanoparticle at atomic resolution. Nature 496, 74–77. https://doi.org/10.1038/nature12009.

Crowther, R., DeRosier, D., 1970. The reconstruction of a three-dimensional structure from projections and its application to electron microscopy. Proc. R. Soc. A 317, 319–340. https://doi.org/10.1098/rspa.1970. 0119.

de la Peña, F., Ostasevicius, T., Tonaas Fauske, V., Burdet, P., Jokubauskas, P., Nord, M., Sarahan, M., Prestat, E., Johnstone, D.N., Taillon, J., et al., 2017. Electron microscopy (big and small) data analysis with the open source software package HyperSpy. Microsc. Microanal. 23, 214–215. https://doi.org/10.1017/S1431927617001751.

de la Peña, F., Prestat, E., Burdet, P., Lähnemann, J., MacArthur, K.E., Fauske, V.T., Sarahan, M., Francis, C., Johnstone, D.N., Ostasevicius, T., Migunov, V., Furnival, T., Nord, M., Mazzucco, S., Eljarrat, A., Caron, J., Aarholt, T., Poon, T., Jokubauskas, P., actions-user, Winkler, F., Taillon, J., Slater, T., pquinn-dls, Guzzinati, G., Myers, J.C., Tappy, N., Garmannslund, A., 2024. Hyperspy/exspy: V0.2. Zenodo. doi:10.5281/zenodo.10953535.

de la Peña, F., Prestat, E., Fauske, V.T., Lähnemann, J., Burdet, P., Jokubauskas, P., Furnival, T., Francis, C., Nord, M., Ostasevicius, T., MacArthur, K.E., Johnstone, D. N., Sarahan, M., Taillon, J., Aarholt, T., pquinn dls, Migunov, V., Eljarrat, A., Caron, J., Nemoto, T., Poon, T., Mazzucco, S., actions user, Tappy, N., Cautaerts, N., Somnath, S., Slater, T., Walls, M., pietsjoh, Ramsden, H., 2024. Hyperspy. doi:10.5281/zenodo.11148112.

Downing, K.H., Sui, H., Auer, M., 2007. Electron tomography: a 3D view of the subcellular world. Anal. Chem. 79, 7949–7957. https://doi.org/10.1021/ac071982u.

Frank, J., 2006. Electron Tomography. Springer, New York. https://doi.org/10.1007/978-0-387-69008-7.

Gan, L., Jensen, G.J., 2012. Electron tomography of cells. Q. Rev. Biophys. 45, 27–56. https://doi.org/10.1017/S0033583511000102.

Gass, M.H., Koziol, K.K., Windle, A.H., Midgley, P.A., 2006. Fourdimensional spectral tomography of carbonaceous nanocomposites. Nano Lett. 6, 376–379. https://doi.org/10.1021/nl052120g.

Genc, A., Kovarik, L., Gu, M., Cheng, H., Plachinda, P., Pullan, L., Freitag, B., Wang, C., 2013. XEDS STEM tomography for 3D chemical characterization of nanoscale particles. Ultramicroscopy 131, 24–32. https://doi.org/10.1016/j.ultramic.2013.03.023.

Gilbert, P., 1972. Iterative methods for the three-dimensional reconstruction of an object from projections. J. Theor. Biol. 36, 105–117. https://doi.org/10.1016/0022-5193(72)90180-4.

Goris, B., Backer, A., Aert, S., Gómez-Graña, S., Liz-Marzán, L.M., Tendeloo, G., Bals, S., 2013. Three-dimensional elemental mapping at the atomic scale in bimetallic nanocrystals. Nano Lett. 13, 4236–4241. https://doi.org/10.1021/nl401945b.

Goris, B., Meledina, M., Turner, S., Zhong, Z., Batenburg, K.J., Bals, S., 2016. Three dimensional mapping of Fe dopants in ceria nanocrystals using direct spectroscopic electron tomography. Ultramicroscopy 171, 55–62. https://doi.org/10.1016/j.ultramic.2016.08.017.

Grenier, A., Duguay, S., Barnes, J., Serra, R., Haberfehlner, G., Cooper, D., Bertin, F., Barraud, S., Audoit, G., Arnoldi, L., Cadel, E., Chabli, A., Vurpillot, F., 2014. 3d analysis of advanced nano-devices using electron and atom probe tomography. Ultramicroscopy 136, 185–192. https://doi.org/10.1016/j.ultramic.2013.10.001.

Guizar-Sicairos, M., Thurman, S.T., Fienup, J.R., 2008. Efficient subpixel image registration algorithms. Opt. Lett. 33, 156–158 doi:10.1364/OL.33.000156.

Gürsoy, D., De Carlo, F., Xiao, X., Jacobsen, C., 2014. TomoPy: A framework for the analysis of synchrotron tomographic data. J. Synchrotron Radiat. 21, 1188–1193. https://doi.org/10.1107/S1600577514013939.

Haberfehlner, G., Orthacker, A., Albu, M., Li, J., Kothleitner, G., 2014. Nanoscale voxel spectroscopy by simultaneous EELS and EDS tomography. Nanoscale 6, 14563–14569. https://doi.org/10.1039/c4nr04553j.

Han, Y., Jang, J., Cha, E., Lee, J., Chung, H., Jeong, M., Kim, T.G., Chae, B.G., Kim, H.G., Jun, S., Hwang, S., Lee, E., Ye, J.C., 2021. Deep learning STEM-EDX tomography of nanocrystals. Nat. Mach. Intell. 3, 267–274. https://doi.org/10.1038/s42256-020-00289-5.

Herzing, A., 2024. usnistgov/etspy: ETSpy package. URL: ⟨https://github.com/usnistgov/etspy⟩. doi: 10.18434/mds2-3616.

Hunter, J.D., 2007. Matplotlib: a 2d graphics environment. Comput. Sci. Eng. 9, 90–95. https://doi.org/10.1109/MCSE.2007.55.

Johnstone, D., Crout, P., Francis, C., Nord, M., Laulainen, J., Høgås, S., Opheim, E., Prestat, E., Martineau, B., Bergh, T., Cautaerts, N., Ånes, H.W., Smeets, S., Femoen, V.J., Ross, A., Broussard, J., Huang, S., Collins, S., Furnival, T., Jannis, D., Hjorth, I., Jacobsen, E., Danaie, M., Herzing, A., Poon, T., Dagenborg, S., Bjørge, R., Iqbal, A., Morzy, J., Doherty, T., Ostasevicius, T., Thorsen, T.I., von Lany, M., Tovey, R., Vacek, P., 2024. Pyxem/pyxem: V0.19.1. Zenodo. doi:10.5281/zenodo.11585254.

Kremer, J.R., Mastronarde, D.N., McIntosh, J., 1996. Computer visualization of three-dimensional image data using IMOD. J. Struct. Biol. 116, 71–76. https://doi.org/10.1006/jsbi.1996.0013.

Kübel, C., Voigt, A., Schoenmakers, R., Otten, M., Su, D., Lee, T.C., Carlsson, A., Bradley, J., 2005. Recent advances in electron tomography: TEM and HAADF-STEM tomography for materials science and semiconductor applications. Microsc. Microanal. 11, 378–400. https://doi.org/10.1017/s1431927605050361.

Lähnemann, J., Orri, J.F., Prestat, E., Ånes, H.W., Johnstone, D.N., Migrator, L.G.T.M., Tappy, N., 2023. LumiSpy/lumispy: V0.2.2. Zenodo. doi:10.5281/zenodo.7747350.

Lichtner, G., 2023. Pystackreg github repository. URL: ⟨https://github⟩. ⟨com/glichtner/pystackreg⟩.

Lučić, V., Förster, F., Baumeister, W., 2005. Structural studies by electron tomography: from cells to molecules. Annu. Rev. Biochem. 74, 833–865. https://doi.org/10.1146/annurev.biochem.73.011303.074112.

Mastronarde, D.N., 2005. Automated electron microscope tomography using robust prediction of specimen movements. J. Struct. Biol. 152, 36–51. https://doi.org/10.1016/j.jsb.2005.07.007.

Mastronarde, D.N., 2024a. IMOD: Image processing and 3d reconstruction. URL: ⟨https://bio3d.colorado.edu/imod/⟩.

Mastronarde, D.N., 2024b. SerialEM: A program for automated electron microscope tomography. URL: ⟨https://bio3d.colorado.edu/SerialEM/⟩ ⟨index.html⟩.

McEwen, B.F., Marko, M., 2001. The emergence of electron tomography as an important tool for investigating cellular ultrastructure. J. Histochem. Cytochem. 49, 553–563. https://doi.org/10.1177/002215540104900502.

Miao, J., Ercius, P., Billinge, S.J.L., 2016. Atomic electron tomography: 3d structures without crystals. Science 353, aaf2157. https://doi.org/10.1126/science.aaf2157.

Midgley, P.A., Dunin-Borkowski, R.E., 2009. Electron tomography and holography in materials science. Nat. Mater. 8. https://doi.org/10.1038/nmat2406.

Midgley, P., Weyland, M., 2003. 3d electron microscopy in the physical sciences: the development of z-contrast and EFTEM tomography. Ultramicroscopy 96, 413–431. https://doi.org/10.1016/s0304-3991(03)00105-0.

Möbus, G., Doole, R.C., Inkson, B.J., 2003. Spectroscopic electron tomography. Ultramicroscopy 96, 433–451. https://doi.org/10.1016/s0304-3991(03)00106-2.

Muller, D., Ercius, P., 2009. Electron tomography in materials science. Microsc. Microanal. 15, 1534–1535. https://doi.org/10.1017/s1431927609098262.

NIST, 1999. NIST SRM 2135c Certificate of Analysis Ni/Cr Thin Film Depth Profile Standard. Technical Report. National Institute of Standards and Technology. Gaithersburg, MD USA.

Nord, M., Vullum, P.E., MacLaren, I., Tybell, T., Holmestad, R., 2017. Atomap: a new software tool for the automated analysis of atomic resolution images using two-dimensional Gaussian fitting. Adv. Struct. Chem. Imaging 3, 9. https://doi.org/10.1186/s40679-017-0042-5.

Paatero, P., Tapper, U., 1994. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. Environmetrics 5, 111–126. https://doi.org/10.1002/env.3170050203.

Pelz, P.M., Groschner, C., Bruefach, A., Satariano, A., Ophus, C., Scott, M.C., 2022. Simultaneous successive twinning captured by atomic electron tomography. ACS Nano 16, 588–596. https://doi.org/10.1021/acsnano.1c07772 (publisher: American Chemical Society).

Pfannmöller, M., Heidari, H., Nanson, L., Lozman, O., Chrapa, M., Offermans, T., Nisato, G., Bals, S., 2015. Quantitative tomography of organic photovoltaic blends at the nanoscale. Nano Lett. 15, 6634–6642. https://doi.org/10.1021/acs.nanolett.5b02437.

Prestat, E., de la Peña, F., Lähnemann, J., Jokubauskas, P., Fauske, V.T., pietsjoh, Ostasevicius, T., Nemoto, T., Francis, C., Johnstone, D.N., Furnival, T., Cautaerts, N., Somnath, S., pquinn dls, Caron, J., MacArthur, K.E., Nord, M., Burdet, P., Tappy, N., Aarholt, T., Poon, T., Taillon, J., Slater, T., Migunov, V., DENSmerijn, Sarahan, M., Ånes, H.W., 2024b. hyperspy/rosettasciio: v0.5. URL: ⟨https://doi.org/10.5281/zenodo⟩. 12208017, doi:10.5281/zenodo.12208017.

Prestat, E., de la Peña, F., Migunov, V., Lähnemann, J., Caron, J., Winkler, F., Francis, C., Slater, T., MacArthur, K.E., Ostasevicius, T., Taillon, J., pquinn-dls, Furnival, T., Johnstone, D.N., Aarholt, T., Poon, T., Nord, M., Jokubauskas, P., 2024a. Hyperspy/holospy: V0.3. Zenodo. doi:10.5281/zenodo.10999344.

Radon, J., 1917. Uber die bestimmung von funktionen durch ihre integralwerte langs gewissez mannigfaltigheiten. Ber. Verh. Sachs. Akad. Wiss. Leipz., Math. Phys. Kl. 69.

Ramachandran, G.N., Lakshminarayanan, A.V., 1971. Three-dimensional Reconstruction from Radiographs and Electron Micrographs: Application of Convolutions instead of Fourier Transforms. Proceedings of the National Academy of Sciences 68, 2236–2240. URL: ⟨https://www.pnas.org/doi/⟩ 10.1073/pnas.68.9.2236, doi:10.1073/pnas.68.9.2236.

Rosier, D.D., Klug, A., 1968. Reconstruction of three dimensional structures from electron micrographs. Nature 217, 130–134. https://doi.org/10.1038/217130a0.

Rudin, L.I., Osher, S., Fatemi, E., 1992. Nonlinear total variation based noise removal algorithms. Phys. D: Nonlinear Phenom. 60, 259–268. https://doi.org/10.1016/0167-2789(92)90242-F.

Saghi, Z., Holland, D.J., Leary, R., Falqui, A., Bertoni, G., Sederman, A.J., Gladden, L.F., Midgley, P.A., 2011. Three-dimensional morphology of iron oxide nanoparticles with reactive concave surfaces. a compressed sensingelectron tomography (CS-ET) approach. Nano Lett. 11, 4666–4673 doi:10.1021/nl202253a.pMID: 21950497.

Sanders, T., Prange, M., Akatay, C., Binev, P., 2015. Physically motivated global alignment method for electron tomography. Adv. Struct. Chem. Imaging 1, 4. https://doi.org/10.1186/s40679-015-0005-7.

Schwartz, J., Di, Z.W., Jiang, Y., Fielitz, A.J., Ha, D.-H., Perera, S.D., El Baggari, I., Robinson, R.D., Fessler, J.A., Ophus, C., Rozeveld, S., Hovden, R., 2022b. Imaging atomic-scale chemistry from fused multi-modal electron microscopy. npj Comput. Mater. 8 (16). https://doi.org/10.1038/s41524-021-00692-5.

Schwartz, J., Di, Z.W., Jiang, Y., Manassa, J., Pietryga, J., Qian, Y., Gee Cho, M., Rowell, J.L., Zheng, H., Robinson, R.D., Gu, J., Kirilin, A., Rozeveld, S., Ercius, P., Fessler, J.A., Xu, T., Scott, M., Hovden, R., 2024. Imaging 3D chemistry at 1 nm resolution with fused multi-modal electron tomography. Nat. Commun. 15, 3555. https://doi.org/10.1038/s41467-024-47558-0.

Schwartz, J., Harris, J., Pietryga, J., Zheng, H., Kumar, P., Visheratina, A., Kotov, N.A., Major, B., Avery, P., Ercius, P., Ayachit, U., Geveci, B., Muller, D.A., Genova, A., Jiang, Y., Hanwell, M., Hovden, R., 2022a. Real-time 3D analysis during electron tomography using tomviz. Nat. Commun. 13, 4458. https://doi.org/10.1038/s41467-022-32046-0.

Scott, M.C., Chen, C.C., Mecklenburg, M., Zhu, C., Xu, R., Ercius, P., Dahmen, U., Regan, B.C., Miao, J., 2012. Electron tomography at 2.4angström resolution. Nature 483, 444–447. https://doi.org/10.1038/nature10934.

Slater, T., CameronGBell, Mohsen, 2021. ePSIC-DLS/particlespy: V0.6.0. Zenodo. doi:10.5281/zenodo.5094360.

Thévenaz, P., 2011. Turboreg: An ImageJ plugin for the automatic alignment of a source image or a stack to a target image. URL: ⟨https://bigwww⟩. epfl.ch/thevenaz/turboreg/.

Thévenaz, P., Ruttimann, U., Unser, M., 1998. A pyramid approach to subpixel registration based on intensity. IEEE Trans. Image Process. 7, 27–41. https://doi.org/10.1109/83.650848.

tomviz, 2024. Tomviz: Open-source 3d visualization for electron tomography. URL: ⟨https://tomviz.org/⟩.

Turk, M., Baumeister, W., 2020. The promise and the challenges of cryo-electron tomography. FEBS Lett. 594, 3243–3261. https://doi.org/10.1002/1873-3468.13948.

van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T., Contributors, 2014. Scikit-image: image processing in Python. e453 PeerJ 2. https://doi.org/10.7717/peerj.453.

Wang, C., Ding, G., Liu, Y., Xin, H.L., 2020a. 0.7 Å resolution electron tomography enabled by deep-learning-aided information recovery. Adv. Intell. Syst. 2, 2000152. https://doi.org/10.1002/aisy.202000152.

Wang, C., Duan, H., Chen, C., Wu, P., Qi, D., Ye, H., Jin, H.J., Xin, H.L., Du, K., 2020b. Three-dimensional atomic structure of grain boundaries resolved by atomic-resolution electron tomography. Matter 3, 1999–2011. https://doi.org/10.1016/j.matt.2020.09.003.

Weyland, M., Midgley, P.A., 2004. Electron tomography. Mater. Today 7, 32–40. https://doi.org/10.1016/s1369-7021(04)00569-3.

Wolf, D., 2012. Accurate tilt series alignment for single axis tomography by sinogram analysis. Proc. 15th Eur. Microsc. Congr.

Xu, R., Chen, C.C., Wu, L., Scott, M.C., Theis, W., Ophus, C., Bartels, M., Yang, Y., Ramezani-Dakhel, H., Sawaya, M.R., Heinz, H., Marks, L.D., Ercius, P., Miao, J., 2015. Three-dimensional coordinates of individual atoms in materials revealed by electron tomography. Nat. Mater. 14, 1099–1103. https://doi.org/10.1038/nmat4426.

Yaguchi, T., Konno, M., Kamino, T., Watanabe, M., 2008. Observation of three-dimensional elemental distributions of a Si device using a 360°tilt FIB and the cold field-emission STEM system. Ultramicroscopy 108, 1603–1615. https://doi.org/10.1016/j.ultramic.2008.06.003.

Yang, Y., Chen, C.C., Scott, M.C., Ophus, C., Xu, R., Pryor, A., Wu, L., Sun, F., Theis, W., Zhou, J., Eisenbach, M., Kent, P.R.C., Sabirianov, R.F., Zeng, H., Ercius, P., Miao, J., 2017. Deciphering chemical order/disorder and material properties at the single-atom level. Nature 542, 75–79. https://doi.org/10.1038/nature21042.

Yang, Y., Zhou, J., Zhu, F., Yuan, Y., Chang, D.J., Kim, D.S., Pham, M., Rana, A., Tian, X., Yao, Y., Osher, S.J., Schmid, A.K., Hu, L., Ercius, P., Miao, J., 2021. Determining the three-dimensional atomic structure of an amorphous solid. Nature 592, 60–64. https://doi.org/10.1038/s41586-021-03354-0.

Young, L.N., Villa, E., 2023. Bringing structure to cell biology with cryoelectron tomography. Annu. Rev. Biophys. 52, 573–595. https://doi.org/10.1146/annurev-biophys-111622-091327.

Zanaga, D., Altantzis, T., Polavarapu, L., Liz-Marzán, L.M., Freitag, B., Bals, S., 2016. A new method for quantitative xeds tomography of complex heteronanostructures. Part. Part. Syst. Charact. 33, 396–403. https://doi.org/10.1002/ppsc.201600021.

Zhang, Q., Kusada, K., Wu, D., Yamamoto, T., Toriyama, T., Matsumura, S., Kawaguchi, S., Kubota, Y., Kitagawa, H., 2018. Selective control of FCC and HCP crystal structures in Au–Ru solid-solution alloy nanoparticles. Nat. Commun. 9, 510. https://doi.org/10.1038/s41467-018-02933-6.

Zhong, Z., Goris, B., Schoenmakers, R., Bals, S., Batenburg, J.K., 2017. A bimodal tomographic reconstruction technique combining EDS-STEM and HAADF-STEM. Ultramicroscopy 174, 35–45. https://doi.org/10.1016/j.ultramic.2016.12.008.

Zhong, Z., Palenstijn, W.J., Adler, J., Batenburg, K.J., 2018. EDS tomographic reconstruction regularized by total nuclear variation joined with HAADF-STEM tomography. Ultramicroscopy. https://doi.org/10.1016/j.ultramic.2018.04.011.

Ziese, U., de Jong, K.P., Koster, A.J., 2004. Electron tomography: a tool for 3d structural probing of heterogeneous catalysts at the nanometer scale. Appl. Catal. A: Gen. 260, 71–74. https://doi.org/10.1016/j.apcata.2003.10.014.