Ultramicroscopy 157 (2015) 35–47

Contents lists available at ScienceDirect

Ultramicroscopy

journal homepage: www.elsevier.com/locate/ultramic

The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography

Wim van Aarle [a,*,1], Willem Jan Palenstijn [a,b,1], Jan De Beenhouwer [a], Thomas Altantzis [c], Sara Bals [c], K. Joost Batenburg [a,b,d], Jan Sijbers [a]

[a] iMinds-Vision Lab, University of Antwerp, Universiteitsplein 1, B-2610 Wilrijk, Belgium
[b] Centrum Wiskunde & Informatica, Science Park 123, NL-1098 XG Amsterdam, The Netherlands
[c] Electron Microscopy for Materials Science, University of Antwerp, Groenenborgerlaan 171, B-2020 Wilrijk, Belgium
[d] Mathematical Institute, Leiden University, P.O. Box 9512, NL-2300 RA Leiden, The Netherlands

ARTICLE INFO

Article history:
Received 19 December 2014
Received in revised form
28 April 2015
Accepted 4 May 2015
Available online 6 May 2015

Keywords:
Electron tomography
Reconstruction
ASTRA Toolbox
Dual-axis

ABSTRACT

We present the ASTRA Toolbox as an open platform for 3D image reconstruction in tomography. Most of the software tools that are currently used in electron tomography offer limited flexibility with respect to the geometrical parameters of the acquisition model and the algorithms used for reconstruction. The ASTRA Toolbox provides an extensive set of fast and flexible building blocks that can be used to develop advanced reconstruction algorithms, effectively removing these limitations. We demonstrate this flexibility, the resulting reconstruction quality, and the computational efficiency of this toolbox by a series of experiments, based on experimental dual-axis tilt series.

1. Introduction

In recent years, electron tomography has proven itself as a powerful technique for imaging the 3D structure of nanomaterials. The ability to compute images of the interior of electron microscopy samples from a series of (S)TEM images has served as a catalyst for materials modelling, simulation, and synthesis. In electron tomography, a 3D image of a microscopic sample is computed from a series of projection images (the so-called *tilt series*), acquired along a range of tilt angles. The series of images is first aligned to correct for geometrical distortions during acquisition, such as sample drift, and subsequently processed by a *tomographic reconstruction algorithm. Weighted Backprojection* (*WBP*) and the *Simultaneous Iterative Reconstruction Technique* (*SIRT*) [1] are the most common algorithms used for this reconstruction task.

Electron tomography can be used to reconstruct 3D images for a broad variety of electron microscopy techniques, including *Bright-Field Transmission Electron Microscopy* (*BF-TEM*) [2], *Annular Dark-Field Scanning TEM* (*ADF-STEM*) [3–6], and *Energy Filtered TEM* (*EF-TEM*) [7,8]. The main requirement is that the acquired projection images depend monotonically on some physical property of the sample, integrated along a set of parallel lines.

Most electron tomography users do not deal with reconstruction algorithms directly. Rather, these methods are implemented in a software package that provides the user with an interface to set certain reconstruction parameters, while performing the computations internally. A range of software packages exist for reconstruction in electron tomography, each with their own advantages and drawbacks, and their own user base. Some microscope vendors offer a software package along with the microscope, such as the Inspect3D software of FEI, the Digital Micrograph software from the Gatan Company, and the Hitachi Tomography plugin [9]. Within the academic community, several free packages have been developed including IMOD [10,11], EFTET-J [7], Protomo [12], UCSF tomography [13,14], TomoJ [15], TOM Toolbox [16] and TxBR [17].

* Corresponding author.
E-mail addresses: wim.vanaarle@uantwerpen.be (W. van Aarle), willemjan.palenstijn@uantwerpen.be (W.J. Palenstijn), jan.debeenhouwer@uantwerpen.be (J. De Beenhouwer), thomas.altantzis@uantwerpen.be (T. Altantzis), sara.bals@uantwerpen.be (S. Bals), joost.batenburg@cwi.nl (K.J. Batenburg), jan.sijbers@uantwerpen.be (J. Sijbers).
[1] Joint first authors.

http://dx.doi.org/10.1016/j.ultramic.2015.05.002
0304-3991/© 2015 Elsevier B.V. All rights reserved.

Key advantages of using established software packages for electron tomography are the reliability implied by the software's proven track record and the user friendliness that such packages can provide. On the other hand, the implemented reconstruction tools typically provide only limited flexibility to the user of the software. Here, we mention three such limitations: (i) the set of reconstruction algorithms that can be used is limited, typically only offering one or two alternatives (e.g., WBP and SIRT); (ii) the software assumes a fixed geometrical setup for the experiment (e.g., single-axis tilting, or dual-axis tilting with an angle of 90° between the two series) without the flexibility to change the experiment; (iii) the computational efficiency of the implemented algorithms is sometimes limited, requiring a long time to reconstruct a large dataset.

For a routine user of electron tomography, the functionality offered by the established software packages is often sufficient. However, for a *technique developer*, either on the experimental or on the algorithmic side, the limitations imposed by using a fixed software package can be an obstacle in experimenting with new concepts and ideas. At present, no software platform is available that suits the development of advanced, efficient algorithms capable of dealing with various geometries and constraints.

The *All Scale Tomographic Reconstruction Antwerp (ASTRA) Toolbox* is a software platform developed at the University of Antwerp, Belgium, and at the Centrum Wiskunde Informatica (CWI), Amsterdam, The Netherlands, to address the need for a fast, flexible development platform for tomography algorithms [18,19]. It provides a set of building blocks that can deal with various geometrical setups and incorporate a variety of constraints in an efficient manner. The toolbox is accessible through MATLAB and Python, providing a powerful platform for algorithm prototyping, and is available as open source software under a GPLv3 license [20].

Due to its flexible nature, the ASTRA Toolbox is suitable for addressing a wide range of computational problems in many tomographic applications such as medical CT, biomedical [21] or industrial micro-CT, synchrotron tomography [22], and electron tomography [23]. It offers full 3D flexibility for modelling misalignments and multi-directional tilt series and allows us to perform parallelized computations using such complex geometrical setups on *Graphics Processing Units* (*GPUs*). Through its integration in MATLAB and Python, advanced numerical code such as regularized reconstruction algorithms (e.g. total variation minimization (TV-min) [24]), can be directly applied to large experimental datasets.

In this paper, we provide an overview of the ASTRA Toolbox and its design. We demonstrate the flexibility of the toolbox by constructing a complex reconstruction algorithm for a dual-tilt geometry with just a few lines of MATLAB code. We investigate the resulting reconstruction quality and the computational efficiency of the toolbox by a series of experiments, based on experimental dual-axis tilt series.

2. Basic concepts and framework design

We start by describing the architecture of the ASTRA Toolbox and its key components. Internally, the software consists of three layers (Fig. 1): (i) the first, low-level layer provides efficiently implemented algorithm building blocks such as projection and backprojection operators that are GPU accelerated using NVIDIA CUDA. (ii) The second, middle-level C++ layer contributes a range of algorithms, such as reconstruction algorithms, that make use of these building blocks. (iii) The third, top-level presents an easy to use interface of these algorithms and building blocks to the end user. Two options are provided: a MATLAB interface implemented using the MATLAB MEX framework and a Python interface. Both offer the same features and design philosophy and differ only slightly in their syntax. In the remainder of this work, we will focus on the conceptual structure as experienced by a user running MATLAB scripts.

Three main concepts are involved in using the toolbox (Fig. 2): (i) the projection and volume data (Section 2.1), (ii) the spatial geometry of the experimental setup (Section 2.2), and (iii) the algorithms to be executed on the data (Section 2.3).

2.1. Projection and volume data

Data objects are used to store projection or volume data within the toolbox. Typically, input projection data is first loaded into the

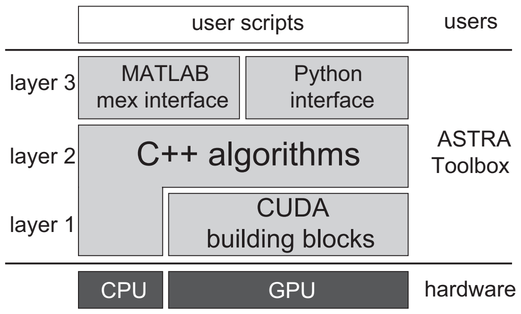

Fig. 1. Schematic overview of the ASTRA Toolbox design.

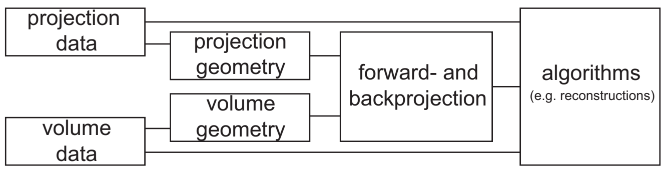
projection data
projection geometry
forward- and backprojection
algorithms (e.g. reconstructions)
volume geometry
volume data

Fig. 2. Schematic overview of the basic ASTRA Toolbox objects and their relations.

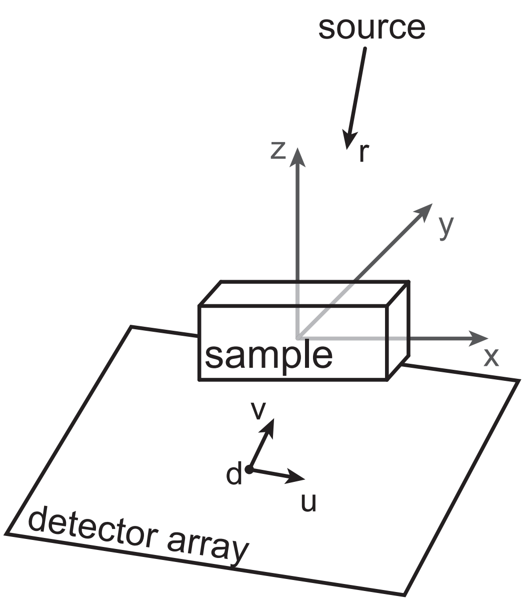

Fig. 3. Scheme depicting the projection geometry of a single projection direction.

MATLAB environment in the form of a double precision matrix. However, for the ASTRA algorithms to use this data it has to be accessible from the lower (CPU or GPU) layer of the toolbox. Simple functions are provided to copy data from the interface to the toolbox and to create an empty dataset in system memory. Once this is done, each data object is referred to by a unique identifier or handler, much like how file I/O is handled in MATLAB. Later on, the user can use this identifier to use the data in an algorithm or to copy the data from system memory back into the MATLAB environment.

2.2. *Spatial geometry of the setup*

Each data object is linked to its corresponding *volume geometry* or *projection geometry*, specifying the setup of the scanning system. The volume geometry describes the pixel or the voxel grid on which the object is represented. This volume has the shape of a rectangle or box centered around the origin. The projection geometry describes the source and the detector setup relative to the volume geometry. The ASTRA Toolbox supports different types of geometry (parallel, fan-beam and cone-beam) used in different types of tomography. For electron tomography, a parallel beam projection geometry is the most relevant. In this geometry, the position and orientation of the detector and the electron beam can be fully specified in 3D, for each projection direction separately.

In practice, electron microscopes used for tomography contain a stationary source and detector and a tilting sample. We, however, define the geometry in a frame of reference where the sample remains stationary. This means that instead of a tilting sample, we define a rotating source and a detector setup, moving around the stationary sample. A 3D parallel beam projection geometry can be regarded as a series of projections, each defined by the four following 3D vectors (also refer to Fig. 3):

- The direction of the beam (r_x, r_y, r_z).
- The center of the detector plane, specified by 3D coordinates (d_x, d_y, d_z). For a parallel beam geometry, moving the (virtual) detector along the direction of the beam does not have any impact on the computation, as projected lines extend indefinitely in that direction (even behind the detector). Therefore, in the example given here, we center the detector plane at the origin for all projections.
- The principal axes of the detector plane (typically horizontal and vertical), specified as 3D vectors (v_x, v_y, v_z) and (u_x, u_y, u_z). The length of these vectors corresponds with the size of one detector pixel and their direction determines the 3D orientation of the detector. By changing their lengths, detector pixels of smaller or larger sizes can be modelled.

By specifying the projection geometry in this way, not only single-axis acquisition schemes can be modelled, but also dual-axis or even multi-axis schemes. Moreover, structural sources of misalignment problems such as a tilt of the detector with respect to the electron beam can be very accurately modelled and incorporated in all reconstruction algorithms built upon the basic structure of the ASTRA Toolbox. Detector shift, for example, can be modelled with sub-pixel precision, forsaking the need of interpolation on the measured data.

Automatic alignment correction of a recorded tilt series is a very difficult problem, even more so for a dual-axis tilt series. In [25], alignment correction is done by regarding it as an optimization procedure for all parameters of the projection geometry over a certain objective function reflecting the reconstruction quality (e.g., the projection difference). Such a technique is only possible when flexible projection geometries are available, such as in the ASTRA Toolbox.

2.3. *Algorithms*

The actual computations are implemented in the algorithm objects. At the time of writing, the toolbox provides efficient implementations for the most popular reconstruction algorithms such as WBP (slice by slice), SIRT and CGLS, a Krylov subspace least squares congruent gradients solver analytically equivalent to LSQR [26,27]. These algorithms are built upon basic GPU accelerated projection and backprojection building blocks. Configuration of an algorithm can be done in the MATLAB interface by linking it to the correct data identifiers and by setting some algorithm specific options (e.g., enabling a minimum constraint in a SIRT reconstruction). Examples of this will be provided in Section 3. It is important to note that the *forward projection (FP)* and *backprojection (BP)* building blocks itself can also be directly accessed from within the MATLAB interface. This means that a user can easily develop new tomographic algorithms or prototypes in which a substantial portion of the computational burden is offloaded to a much faster GPU card. An example of this is provided in Section 3.5. In case GPU cards are unavailable, the ASTRA Toolbox also provides OpenMP accelerated CPU implementations of these building blocks, but only for 2D datasets and slice-by-slice 3D reconstructions.

3. Concrete example: implementing an advanced method for dual-axis reconstruction

In this section we demonstrate the key features of the ASTRA Toolbox — and how to use them — by constructing an advanced reconstruction algorithm in a step-wise manner. Firstly, in Section 3.1, we introduce some notation that will be used to describe these advanced iterative methods. In Section 3.2 we describe the common workflow for creating a three dimensional reconstruction with the ASTRA Toolbox. As an example, we demonstrate how a dual-axis acquisition geometry can be defined, how projections of a given voxel volume can be computed and how a reconstruction can be created using the *Simultaneous Iterative Reconstruction Technique* (*SIRT*) technique.

Subsequently, we describe how this workflow can be extended by relatively small number of MATLAB code lines to include more advanced reconstruction methods. In Section 3.3, we demonstrate how a recently proposed method for dense particle segmentation [23] called *Partially Discrete Algebraic Reconstruction Technique (PDART)* algorithm can be implemented with certain image processing operations within MATLAB, combined with optimized tomographic reconstruction steps (projection and backprojection) of the ASTRA Toolbox. Next, in Section 3.4 we highlight the fact that the voxel size of the reconstructed volume can be chosen independently of the detector pixel size, which provides the ability to reconstruct a subset of the volume at lower resolution than the central region of interest, leading to improved performance. Finally, in Section 3.5, we discuss how the ASTRA building blocks can be used in third party libraries and scripts such as in an existing *Total Variation minimization (TVmin)* script.

3.1. Notation

Let n denote the total number of voxels in the volume and let $\boldsymbol{v} \in \mathbb{R}^n$ denote a vector describing the voxel values of a certain 3D volume containing the scanned sample. Assume a square detector with t^2 being the total number of detectors in a single projection. With l being the total number of projections, the total number of measurements is then $m = lt^2$. Let $\boldsymbol{p} \in \mathbb{R}^m$ denote a vector that contains all tilt series measurements of $\boldsymbol{v}$. We define the matrix $\boldsymbol{W}$ as the *projection matrix*, a linear operator that describes a forward projection of the scanned object:

$$\boldsymbol{W}\boldsymbol{v} = \boldsymbol{p}. \tag{1}$$

The matrix $\boldsymbol{W}$ describes how the projection data $\boldsymbol{p}$ depends on the image volume $\boldsymbol{v}$, i.e., it maps the volume geometry onto the projection geometry. The multiplication of $\boldsymbol{W}$ with a vector $\boldsymbol{v}$ is called a *forward projection (FP)*, the multiplication of $\boldsymbol{W}^T$ with a vector $\boldsymbol{p}$ is called a *backprojection (BP)*. Both operations are of crucial importance in tomographic reconstruction as they take up nearly all the computation time of iterative reconstruction methods. An efficient implementation of these two operations is thus key to any set of tomographic software tools. Moreover, the accuracy of the projection and backprojection can be a defining influence in the accuracy of the reconstruction. The values of $\boldsymbol{W}$ must therefore be specified accurately, and depend on the geometry of the scanning system. Many different types of projection geometries exist: fan beam, cone beam, helical scan, etc. In the remainder of this work we only consider a three-dimensional parallel beam geometry, as it is the most common in electron tomography applications.

A *reconstruction algorithm* takes a volume geometry and projection geometry as inputs, and is then used to compute a reconstructed image $\boldsymbol{v}$ from the measurement $\boldsymbol{p}$ for the given geometrical setup. Many such algorithms exist in the literature. In this paper, we mainly consider the *Simultaneous Iterative Reconstruction Technique (SIRT)*, an iterative solver whose update step in each iteration (k) is

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)} + \boldsymbol{C}\boldsymbol{W}^T\boldsymbol{R}(\boldsymbol{p} - \boldsymbol{W}\boldsymbol{v}^{(k)}), \tag{2}$$

in which $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ is a diagonal matrix denoting the inverse column sums of $\boldsymbol{W}$, i.e., $c_{ii} = 1/\sum_j w_{ij}$, and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ is a diagonal matrix denoting the inverse row sum of $\boldsymbol{W}$, i.e., $r_{jj} = 1/\sum_i w_{ij}$. In case of noiseless data, this update scheme is guaranteed to converge to a weighted least squares solution $\boldsymbol{v}^*$:

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \|\boldsymbol{W}\boldsymbol{v} - \boldsymbol{p}\|_R^2, \tag{3}$$

with $\|\boldsymbol{x}\|_R^2 = \boldsymbol{x}^T\boldsymbol{R}\boldsymbol{x}$ [1].

3.2. Dual-axis reconstruction

Despite the fact that the acquisition of dual-axis tomography datasets is nowadays very common in microbiology applications (where the tilt range per axis is quite limited) and is also being used for materials science applications, sophisticated reconstruction methods for this geometry are generally not supported by the available software. Reconstructions are often obtained by reconstructing both tilt series independently and then averaging the results, which leads to sub-optimal reconstruction quality as only a portion of the data is used in both reconstruction steps. Some methods have been proposed for combined reconstruction [28], but the implementation of such methods can be challenging.

As the ASTRA Toolbox supports the definition of highly flexible geometrical setups for the acquisition procedure, the dual-axis geometry can be modelled in a straightforward way. Moreover, reconstruction algorithms built with the toolbox' building blocks can be used

```
% Define a reconstruction mask object, fill with ones for the first iteration
mask_id = astra_mex_data3d('create', '-vol', vol_geom, 1);

% Configure the SIRT algorithm
cfg = astra_struct('SIRT3D_CUDA');
cfg.ProjectionDataId = proj_id;
cfg.ReconstructionDataId = vol_id;
cfg.option.ReconstructionMaskId = mask_id;
alg_id = astra_mex_algorithm('create', cfg);

for i = 1:150
    % Run (masked) iterative algorithm
    astra_mex_algorithm('iterate', alg_id, 1);
    reconstruction = astra_mex_data3d('get', vol_id);

    % Update the reconstruction mask
    q = reconstruction < tau;
    astra_mex_data3d('set', mask_id, q);

    % Segment and update the reconstruction
    segmentation = double(~q) * rho;
    reconstruction(~q) = rho;
    astra_mex_data3d('set', vol_id, reconstruction);

    % Update the projection data by substracting the projection of the fixed area
    [tmp_id, Ws] = astra_create_sino3d_cuda(segmentation, proj_geom, vol_geom);
    astra_mex_data3d('set', proj_id, p - Ws);
end
```

Fig. 4. MATLAB code for 150 iterations of the PDART algorithm, built upon the framework provided by the ASTRA Toolbox. In Fig. 12, this script is applied on an experimental dataset.

in combination with a dual-axis geometry without any changes to the algorithm itself, making it unnecessary to develop algorithms specifically designed for a particular geometry, such as in [28].

In what follows, we demonstrate the common workflow for creating a three dimensional reconstruction with the ASTRA Toolbox by providing code samples that implement a simple dual-axis reconstruction setup.

1. The projection data is read into the MATLAB environment and stored as a 3D matrix. This step is obviously dependent on the measuring system and its output file formats. At the time of writing, the ASTRA toolbox does not provide methods for reading in specific file formats, but instead relies on the user to be able to do this.

```
p = readdata(filename); % this function is created by the user
```

2. The projection geometry is specified. As discussed in Section 2.2, this geometry is defined by four vectors: $\boldsymbol{r}$, $\boldsymbol{d}$, $\boldsymbol{u}$, and $\boldsymbol{v}$. The required information for this (detector size, projection direction, etc.) is typically found in log files generated alongside of the projection data. The following code sample demonstrates how to construct a projection geometry for a dual-axis projection setup, starting from a single list of tilt angles. We assume here that all projection images of both tilt series have been correctly aligned. Note, however, that it is possible to correct for misalignments by altering the vectors specified in the geometry object, but this greatly beyond the scope of this paper [25].
   In the ASTRA Toolbox, a projection geometry is specified by a 12-column matrix where each row represents a projection image and contains the corresponding vectors $\boldsymbol{r}$, $\boldsymbol{d}$, $\boldsymbol{u}$, and $\boldsymbol{v}$.

```
angles = [...] % vector with the projection angles in each of the tilt series
l = numel(angles);
vectors = zeros(2*l,12); % matrix in which each row contains the vector r,d,u,v,
                         % that define a single projection (Fig.3)

% Projections from the first tilt series
for i = 1:l
    a = angles(i);
    vectors(i,1:3) = [sin(a), 0, -cos(a)]; % ray direction, r
    vectors(i,4:6) = [0, 0, 0];            % centre of detector, d
    vectors(i,7:9) = [cos(a), 0, sin(a)];  % vector from det (0,0) to (0,1), u
    vectors(i,10:12) = [0, 1, 0];          % vector from det (0,0) to (1,0), v
end

% Projections from the second tilt series
for i = (l+1):(2*l)
    a = angles(i-l);
    vectors(i,1:3) = [0, -sin(a), -cos(a)];
    vectors(i,4:6) = [0, 0, 0];
    vectors(i,7:9) = [0, -cos(a), sin(a)];
    vectors(i,10:12) = [1, 0, 0];
end

% Create the ASTRA projection geometry
d = ...; % number of detectors in each projection image
proj_geom = astra_create_proj_geom('parallel3d_vec', [d d], vectors);
```

3. The volume geometry is specified, defining the size, location and number of voxels in the volume (i.e., reconstruction) domain. In electron tomography one typically defines a box-shaped slab that extends outside the zero-degree field-of-view in the x- and

y-directions, while confining the thickness of the sample within the slab. Here, we use a cubic voxel grid centered round the origin, with s^3 voxels of unit length. In Section 3.4 we demonstrate more advanced settings.

```
% Create the ASTRA volume geometry
s = ...; % number of voxels in the volume in each dimension
vol_geom = astra_create_vol_geom([s s s]);
```

4. With the geometry setup, the next step is to load the projection data into the toolbox memory and to allocate some space to store the reconstruction in. The result of these operations is two data identifiers which are used in subsequent steps of the workflow. Note that each data object is linked to its corresponding geometry object.

```
% Create ‘zero’ volume and load the measurements ‘p’ into the toolbox
vol_id = astra_mex_data3d('create', '-vol', vol_geom, 0);
proj_id = astra_mex_data3d('create', '-proj3d', proj_geom, p);
```

5. Once the data is loaded into the toolbox memory, a reconstruction algorithm object can be configured. Here, we use a CUDA accelerated

```
dims_LR = [64 64 64];    % volume size of the LR volume
dims_HR = [40 40 40];    % volume size of the HR volume
c = [...];               % centre of the ROI
a = 4;                   % downsampling factor in the LR volume

% Create the volume geometries
vol_geom_LR = astra_create_vol_geom(dims_LR);
vol_geom_HR = astra_create_vol_geom(dims_HR);

% LR projection geometry: scale u and v to adjust for larger voxels
proj_geom_LR = proj_geom;
proj_geom_LR.Vectors(:,7:12) = proj_geom_LR.Vectors(:,7:12) * a;

% HR projection geometry: translate the centre of the detector to c
proj_geom_HR = proj_geom;
proj_geom_HR.Vectors(:,4:6) = proj_geom_HR.Vectors(:,4:6) - c;
```

Fig. 5. MATLAB code that defines a mixed resolution geometry.

```
function Y = FP(X)
    % X is a vector, containing the HR parts and LR parts concatenated.
    % Extract into two parts, project, and add
    v_HR = reshape(X(1:prod(dims_HR)), dims_HR);
    v_LR = reshape(X((prod(dims_HR)+1):end), dims_LR);
    [~,p_HR] = astra_create_projection3d_cuda(v_HR, proj_geom_HR, vol_geom_HR);
    [~,p_LR] = astra_create_projection3d_cuda(v_LR, proj_geom_LR, vol_geom_LR);
    Y = p_HR + a * p_LR;
    Y = Y(:);
end

function Y = BP(X)
    % X is a vector of the projection data
    % Backproject to LR and HR parts, and combine
    X = reshape(X, size(projs));
    [~,d_HR] = astra_create_backprojection3d_cuda(X, proj_geom_HR, vol_geom_HR);
    [~,d_LR] = astra_create_backprojection3d_cuda(X, proj_geom_LR, vol_geom_LR);

    % zero the HR part of the LR vol
    a = dims(1,1); b = dims(1,2);
    c = dims(2,1); d = dims(2,2);
    e = dims(3,1); f = dims(3,2);
    d_LR(a:b, c:d, e:f) = 0;

    Y = [d_HR(:); d_LR(:)];
end
```

Fig. 6. The forward- and backprojection operators for this mixed resolution volume can be written by combining two single-resolution operators.

SIRT implementation for 3D data problems. We also limit the gray level values allowed in the reconstruction to the interval $[0, 0.07]$, a simple form of prior knowledge about the scanned sample that can lead to improved reconstruction quality. The result of this

```
function v = sirt(p, m, n, iters)
    RW = FP(ones(m, 1));          % inverse row and column sums
    CW = BP(ones(n, 1));
    RW(RW < 1e-4) = Inf;
    CW(CW < 1e-4) = Inf;
    v = zeros(n, 1);
    for k = 1:iters
        r = (p(:) - FP(v)) ./ RW;  % weighted residual computation
        v = v + BP(r) ./ CW;       % weighted backprojection
        v(v<0) = 0;                % non-negativity constraint
    end
end
```

Fig. 7. MATLAB implementation of the SIRT algorithm, with a non-negativity constraint. In Fig. 13, this script is applied on an experimental dataset.

```
# Define the ASTRA Spot operator
W = opTomo('cuda', proj_geom, vol_geom);

# Run 50 iterations of TVmin in an external script
lambda = 10;
D = spdiags([[-ones(s^3-1,1);0], ones(s^3,1)], [0,1], s^3, s^3);
TV = [kron(speye(s^3), D); kron(D, speye(s^3))];
reconstruction = chambollePock3D(W, TV, p(:), 50, lambda);
```

Fig. 8. MATLAB code that shows how the ASTRA Spot Toolbox (opTomo) can be used to use the building blocks in existing script. In Fig. 14, this script is applied on an experimental dataset.

configuration is again a certain identifier.

```
% Configure the algorithm
cfg = astra_struct('SIRT3D_CUDA');
cfg.ProjectionDataId = proj_id;
cfg.ReconstructionDataId = vol_id;
cfg.option.MinConstraint = 0;
cfg.option.MaxConstraint = 0.07;
alg_id = astra_mex_algorithm('create', cfg);
```

6. The algorithm identifier is used to perform 150 iterations on the data that it was provided with.

```
% Run the algorithm
astra_mex_algorithm('iterate', alg_id, 150);
```

7. Finally, the reconstruction data is retrieved into the MATLAB memory, ready for subsequent analysis.

```
% Retrieve the result
reconstruction = astra_mex_data3d('get', vol_id);
```

In Section 4.1, this script is applied on an experimental dataset.

3.3. PDART algorithm

Accurate segmentation of dense nanoparticles within various materials is a challenging problem in electron tomography because of reconstruction artifacts that hinder segmentation (e.g., missing wedge artifacts). Recently, the *Partially Discrete Algebraic Reconstruction Technique* (*PDART*) has been proposed to simultaneously reconstruct and segment dense homogeneous particles that are embedded in (possibly) non-homogeneous material [23]. With PDART, accurate reconstructions of these dense particles can be obtained by exploiting prior knowledge about their gray level $\rho \in \mathbb{R}$. In this section, we show how PDART can be implemented easily and efficiently using the ASTRA Toolbox.

The PDART algorithm interleaves conventional SIRT iterations with segmentation steps during which all pixels of which the gray value exceeds a certain threshold $\tau \in \mathbb{R}$ are assumed to be part of a dense, homogeneous particle. Afterwards, the pixels whose value exceeds the threshold are therefore fixed at the known gray level and are no longer allowed to be changed in subsequent SIRT iterations. This can be achieved by removing these pixels from the reconstruction equation (1). Consider iteration (k). Define $\boldsymbol{q}^{(k)} \in \{0, 1\}^n$ as a vector specifying all pixels that are not fixed, i.e., that do not belong to a dense particle in iteration (k). Define $\boldsymbol{s}^{(k)} \in \{0, \rho\}^n$ as a vector specifying the segmentation of the dense particles:

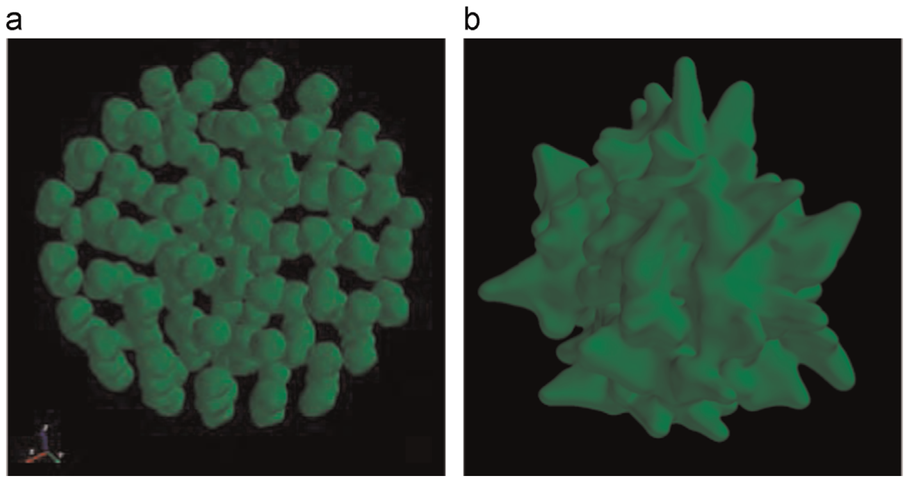

Fig. 9. Surface renderings of the samples used in the experimental setup. (a) Au nanoassembly and (b) Au nanostar.

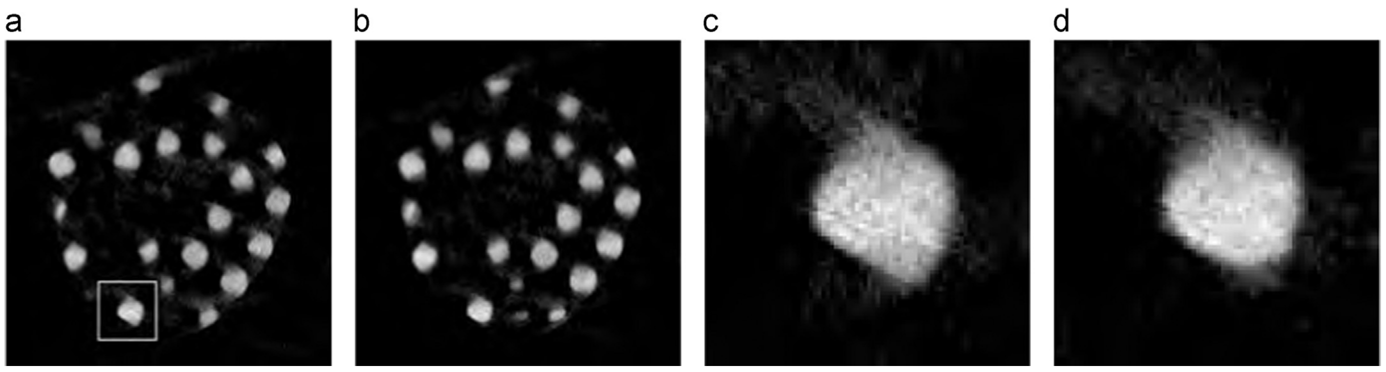

Fig. 10. (a, b) Cross-section of SIRT reconstructions with a single and a dual-axis set of projections. (c, d) Zoomed in region of a single particle (noted by the box in (a)). (b) is the result of the code provided in Section 3.2.

$$q_j^{(k)} = \begin{cases} 1 & \text{if } v_j^{(k)} < \tau \\ 0 & \text{if } v_j^{(k)} \geq \tau \end{cases}, \quad s_j^{(k)} = \begin{cases} 0 & \text{if } v_j^{(k)} < \tau \\ \rho & \text{if } v_j^{(k)} \geq \tau \end{cases}, \quad \forall j \in \{1, \ldots, n\}. \tag{4}$$

In iteration $(k+1)$, the contribution of all pixels with values larger then the threshold is subtracted from the projection data:

$$\boldsymbol{p}^{(k+1)} = \boldsymbol{p} - \boldsymbol{Ws}. \tag{5}$$

This residual projection is then used in a following SIRT iteration, which is restricted to all pixels j that satisfy $q_j^{(k)} = 1$. This operation comes down to the removal from $\boldsymbol{W}$ of the columns corresponding to the dense particle pixels. In the ASTRA toolbox, this can be easily achieved by specifying a reconstruction mask, a data object that defines which pixels are to be considered in the reconstruction (i.e., $\boldsymbol{q}^{(k)}$). In Fig. 4, MATLAB code is provided that performs 50 iterations of the PDART algorithm.

3.4. Mixed resolution reconstructions

Rapid advances in the field of detector technology are enabling reconstructions with an ever increasing resolution, i.e., with an ever increasing volume size. While this improved resolution can obviously be crucial for analysis, it is typically only beneficial in a small region

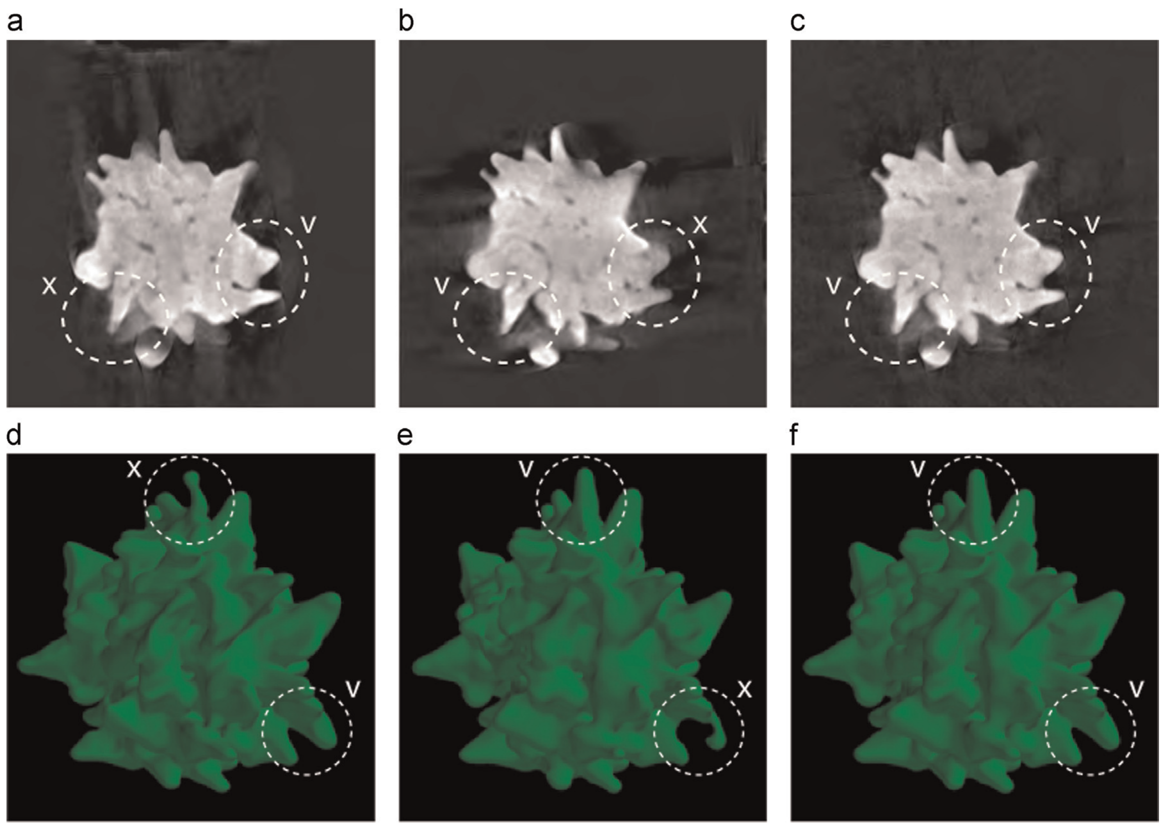

Fig. 11. (a–c) Cross-section of the SIRT reconstructions with a single and a dual-axis set of projections. (d–f) Surface rendering of the SIRT reconstructions. (a) Single-axis 1, (b) single-axis 2, (c) dual-axis, (d) single-axis 1, (e) single-axis 2 and (f) dual-axis.

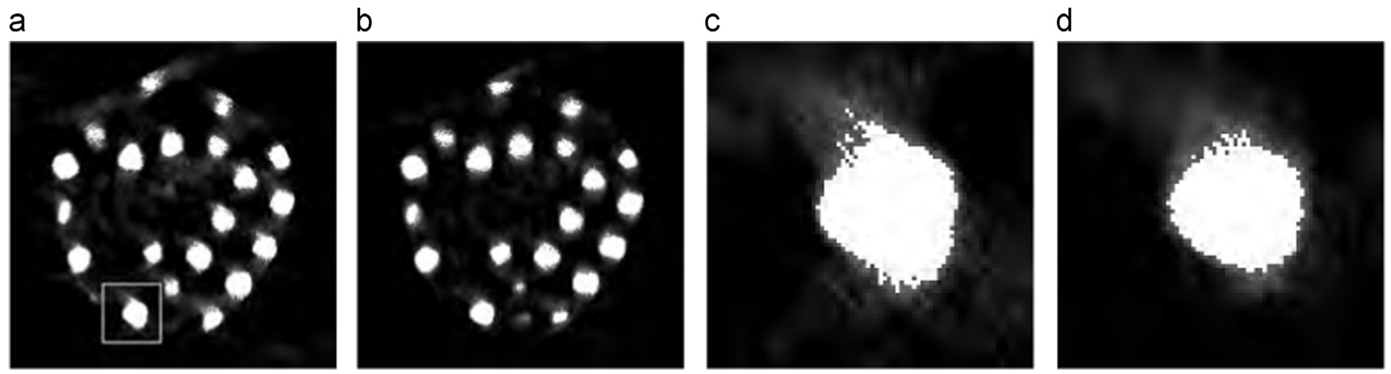

Fig. 12. (a, b) Cross-section of PDART reconstructions with a single and a dual-axis set of projections. (c, d) Zoomed in region of a single particle (noted by the box in (a)). (b) is the result of the code provided in Fig. 4. (a) Single-axis, (b) dual-axis, (c) single-axis and (d) dual-axis.

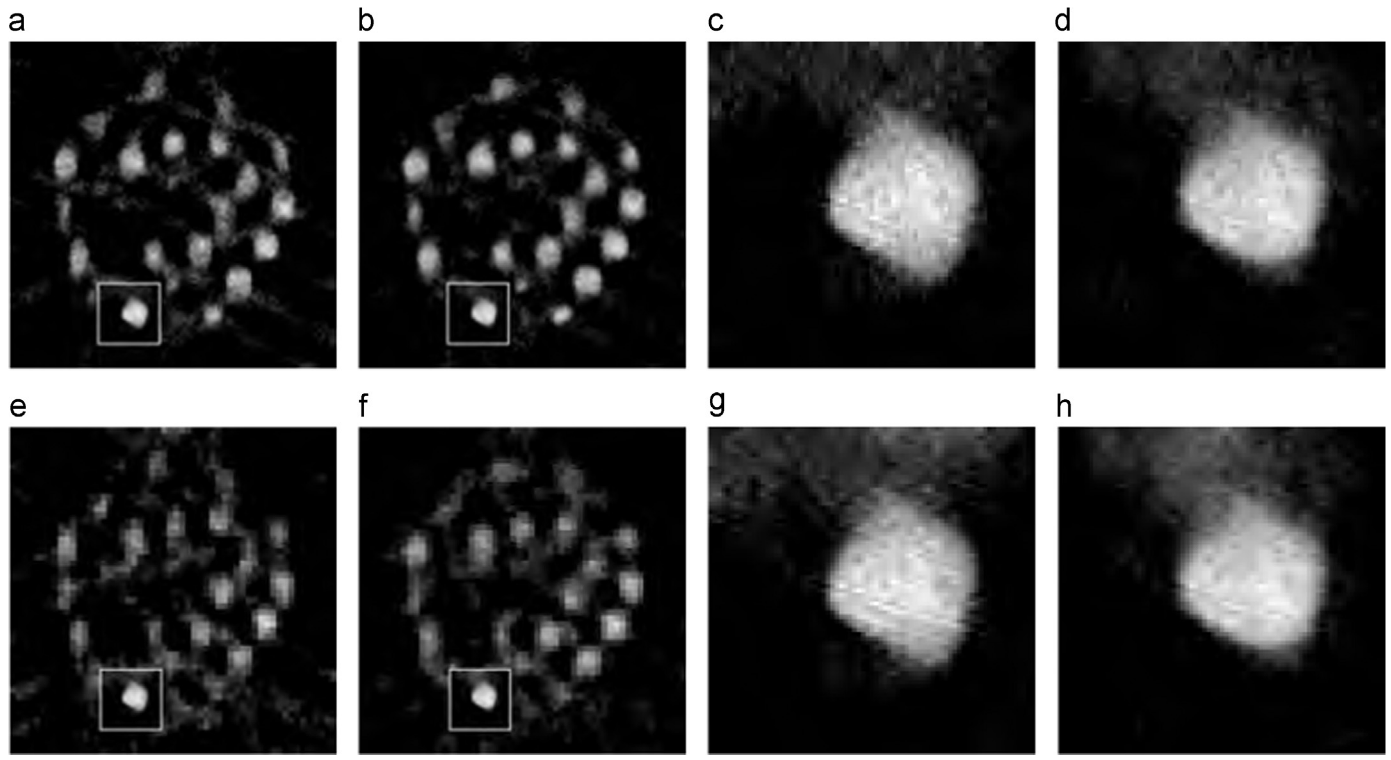

Fig. 13. Cross-section of mixed resolution reconstructions with a single and a dual-axis set of projections, where the background was downsampled by a factor $a=4$ (a–d) and $a=8$ (e–h). (b) and (f) are the result of the code provided in Figs. 5–7. (a) $a=4$, single-axis ROI rmse=0.0319, (b) $a=4$, dual-axis ROI rmse=0.0238, (c) $a=4$, single-axis, (d) $a=4$, dual-axis, (e) $a=8$, single-axis ROI rmse=0.0489, (f) $a=8$, dual-axis ROI rmse=0.0427, (g) $a=8$, single-axis, (h) $a=8$, dual-axis.

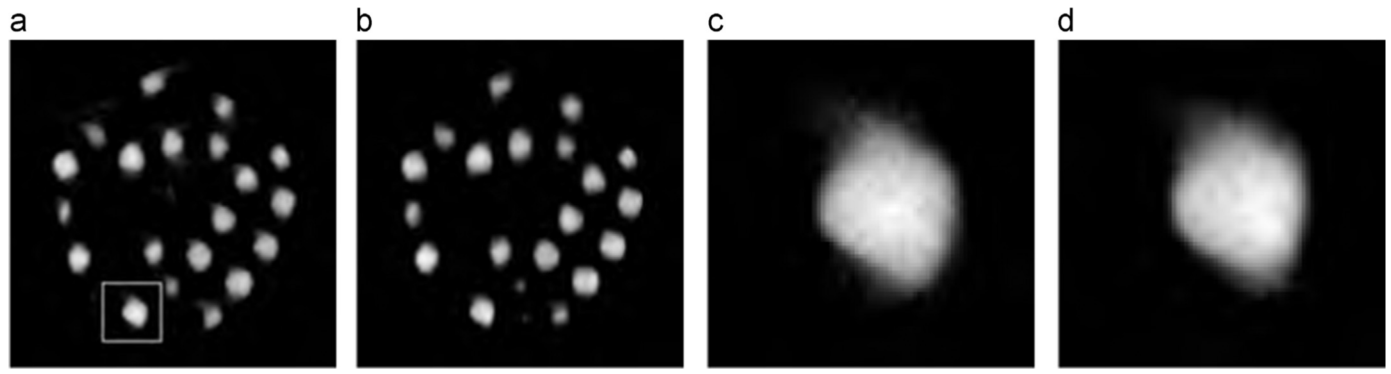

Fig. 14. (a, b) Cross-section of Total Variation Minimization reconstructions with a single and a dual-axis set of projections. (c, d) Zoomed in region of a single particle (noted by the box in (a)). (b) is the result of the code provided in Fig. 8. (a) Single-axis, (b) dual-axis, (c) single-axis, (d) dual-axis.

Table 1
Computation timings for the various reconstructions presented in Section 4.

	SIRT	*PDART*	*TVmin*	*Mixed $a=4$*	*Mixed $a=8$*
Iterations	150	50	50	150	150
Single-axis	134 s	228 s	939 s	59 s	54 s
Dual-axis	323 s	402 s	1027 s	107 s	100 s

of interest (ROI). Consequently, as larger volumes also lead to a vastly increased computational burden (even with modern improvements in computational hardware), one would ideally do a reconstruction with a mixed resolution, combining a slow high resolution (HR) reconstruction inside the ROI with a fast low resolution (LR) reconstruction outside of it.

As the ASTRA projection geometry allows specifying different pixel sizes for the reconstruction and the projection data, mixed resolution reconstructions can be easily achieved by defining two separate volume/projection geometry pairs; one for the global low resolution volume and one for the high resolution region of interest.

Consider the case where a certain object can be expressed in a volume, centered around the origin, of $256 \times 256 \times 256$ voxels of size $1\ \text{nm} \times 1\ \text{nm} \times 1\ \text{nm}$, but where one is only interested in a small $40 \times 40 \times 40$ nm region of interest, centered around a certain point (d_x, d_y, d_z). One could then opt to downsample the reconstruction volume outside of this ROI, for example by a certain factor $a=4$. The LR volume will then contain $64 \times 64 \times 64$ voxels of size $4\ \text{nm} \times 4\ \text{nm} \times 4$ nm. In the ASTRA Toolbox, the unit of the coordinate system is defined by the size of the voxels in the volume geometry, i.e., the voxel size is fixed to 1. To encode the low resolution reconstruction volume, we can shrink the size of the detector pixels in the projection geometry (specified by the unit vectors u and v, see Section 3.2):

$$u_{LR} = a_* u_{HR}, \quad v_{LR} = a_* v_{HR}. \tag{6}$$

Similarly, the origin of the coordinate system is always at the center of the reconstruction volume in the current version of the ASTRA Toolbox. This can, however, be easily circumvented by shifting the projections relatively to the location of the center of the region of interest by a vector. Fig. 5 concerns the creation of this multi-resolution geometry.

The forward- and backprojection operators for this mixed-resolution volume can be written by combining two single-resolution operators, as is demonstrated in Fig. 6. To perform the reconstruction, these custom operators can for example be used in a MATLAB implementation of SIRT shown in Fig. 7.

Note that, alternatively, one could use the same principles to reconstruct the background in a detector native resolution while increasing the resolution in the ROI even further by applying a super-resolution scheme [29]. This is, however, not pursued further in this work.

3.5. Total variation minimization

In the literature, many reconstruction techniques can be found, each with its own merit in particular use cases. Often, they are created and distributed in the form of a MATLAB script or function, in which the tomographic projection model is specified by providing the sparse matrix $\boldsymbol{W}$. The projection operations are thus performed by applying a sparse matrix–vector product, which can be slow and which requires a lot of system memory. It is therefore desirable to plug the efficient ASTRA projector operators into these existing code frameworks. This can be achieved with the separately available ASTRA Spot tools. These tools provide the so-called Spot wrapper around the projection operation and make them appear and act like any other MATLAB matrix. In this way, it is possible to use the fast building blocks inside existing code without even altering it.

As an example, we consider a framework for *Total Variation Minimization (TVmin)*, a reconstruction technique that can, like PDART, be used to counter the effect of missing wedge artifacts [30]. It does so by finding the most sparse solution, or the solution with the most sparse gradient, corresponding to the measurements:

$$\boldsymbol{v}^* = \arg\ \min_v \|\boldsymbol{W}\boldsymbol{v} - \boldsymbol{p}\| + \lambda \|TV(\boldsymbol{v})\|_1, \tag{7}$$

in which λ is a regularization parameter, and $TV(\boldsymbol{v})$ is a function that describes a metric for 'total variation' for each voxel, e.g., its gradient.

Assume that we have a MATLAB script that implements the Chambolle–Pock optimization strategy [31] for solving (7). As arguments, it takes a projection matrix $\boldsymbol{W}$, a *TV*-function expressed as a matrix, the projection data and the number of iterations. By creating an ASTRA Spot 'opTomo' object and linking it to our projection and volume geometry, we can run this algorithm using the ASTRA building blocks (Fig. 8). This way, each time a vector is multiplied by $\boldsymbol{W}$ or $\boldsymbol{W}^T$, the efficiently implemented and memory extensive forward- or backprojection buildings blocks are called.

4. Experimental samples

In this section, we apply the techniques and MATLAB scripts described in the previous section on two experimental dual-tilt series. The tilt series were recorded using a FEI Tecnai G2 electron microscope available at the EMAT laboratory at the University of Antwerp, Belgium. Both datasets correspond to experiments in the field of materials science, but the same code can be used for other applications using dual-axis electron tomography (e.g., life sciences), or indeed other types of tomography (e.g., μCT and synchrotron).

The first dataset, depicted in Fig. 9(a), is acquired from an assembly of Au nanoparticles, embedded in a polymeric matrix [32–34]. The average diameter of the nanoparticles is 20 nm. For the acquisition of the series a Fischione model 2040 dual-tilt tomography holder was used and the microscope was operated at 200 kV. The probe semiconvergence angle was 16 mrad, corresponding to a depth of focus of approximately 70 nm. Each projection image in the tilt series contained 428×428 pixels. The size for the reconstruction was therefore chosen at $428 \times 428 \times 428$ voxels.

The second dataset, depicted in Fig. 9(b), is acquired from a spiked Au nanostar particle [35,36]. A Fischione model 2040 dual-tilt tomography holder was used. The first series were acquired over a tilt range from $-70°$ to $+70°$ and the second one from $-66°$ to $+72°$. The tilt increment was $2°$ in both cases. The accelerating voltage was 200 kV. Each projection image in the tilt series contained 256×256 pixels. The size for the reconstruction was chosen at $256 \times 256 \times 256$ voxels.

In a preprocessing step of both datasets, all projection images (of both tilt series) were aligned onto each other [25].

4.1. Reconstructions

We describe the power of dual-axis reconstructions by performing SIRT reconstructions with only a single-tilt series and with an additional second tilt series, for which we used the geometrical setup as explained in Section 3.2. For the nanoassembly, the reconstructions were the result with 150 SIRT iterations with both a minimum and maximum value constraint set (as in Section 3.2). In Fig. 10, these reconstructions are shown. It is clear that the missing wedge artifacts are indeed greatly reduced around the dense particles and that any subsequent analysis on the segmented dense particles will be more accurate.

For the nanostar dataset, reconstructions were computed using 500 SIRT iterations. The cross sections and surface renderings of these reconstructions are shown in Fig. 11. When inspecting the regions denoted by the dashed circles, one can clearly see that a single tilt series reconstruction does not provide accurately reconstructed *spikes* everywhere due to missing wedge artifacts. Moreover, we observe that the areas that are reconstructed poorly in the first tilt series, are generally much more accurate in a reconstruction based on the second tilt series, and vice versa [28]. By using the full dual axis project data, however, we are able to create reconstructions accurate throughout the entire volume.

The Au nanoassembly dataset is an ideal candidate for reconstruction using the PDART method. In Fig. 12, PDART reconstructions are shown of both the single and the dual-axis tilt series. Again, the missing wedge artifact is clearly much reduced by dual-axis tomography, even more so than in SIRT reconstruction of Fig. 10(b). Furthermore, with the PDART algorithm, the dense particles can be easily extracted from the reconstruction, while still providing a decent reconstruction in the background.

Next, we investigate the effect of the mixed resolution reconstruction described in Section 3.4. In the Au nanoassembly dataset, we selected a single dense particle, in a $64 \times 64 \times 64$ box as our region of interest. As described in Fig. 5, we downsampled the background a factor $a=4$ and $a=8$, effectively reducing the number of voxels in the reconstruction equation from 78,402,752 to 1,483,091 (1.9%) and to 414,760 (0.53%). Fig. 13 shows these mixed resolution SIRT reconstructions. We also included root-mean-square-errors of the reconstructions in this ROI compared to full resolution reconstructions (i.e., Fig. 10). As these rmse values are relatively low, we can say that downsampling the background has little effect on the accuracy inside the ROI. In Section 4.2, we shall investigate the resulting benefit in reconstruction time.

Finally, we apply the Chambolle–Pock TV minimization reconstruction method as implemented in an external MATLAB script, but which uses the ASTRA Toolbox for its projection and backprojection operations. In Fig. 14, reconstructions with $\lambda=10$ are shown. Due to the l_1-norm being minimized during this reconstruction process, the small gray level variations that appear in the background of all previous reconstructions are not visible here.

4.2. Timings

In Table 1, we investigate the computation times of all reconstructions of the Au nanoassembly dataset shown in Section 4. All reconstruction was created on a Xeon E5-2630 system running at 2.30 GHz, supporting 256 GB of memory. All projection operations were accelerated on an NVIDIA Tesla K20X unit.

The first thing to note is that dual-axis reconstructions have a higher computational cost. More projection data means a larger projection matrix $\boldsymbol{W}$ (1), and thus a larger system to solve. Secondly, note that the more advanced reconstruction methods (PDART and TVmin) are slower than the straightforward SIRT algorithm, even if SIRT requires substantially more iterations before obtaining sufficiently accurate results. Finally, note that the mixed resolution SIRT reconstructions indeed result in a clear performance benefit. Compared to standard SIRT reconstructions, mixed resolution reconstructions with $a=4$ resulted in a speedup of 2.27 for single-axis and 3.02 for dual-axis. For $a=8$ this was even slightly larger (2.48 and 3.23).

5. Discussion and conclusions

In this paper, we have demonstrated that the ASTRA Toolbox can be used effectively for developing and implementing advanced algorithms for electron tomography, and for running these algorithms on real-world datasets. The ASTRA Toolbox offers the possibility to precisely specify the geometrical context of the experiment, allowing full flexibility in acquisition schemes. Here we demonstrated the possibility of using a dual-axis tilt scheme in the reconstruction. By combining the geometry definition with sophisticated algorithm concepts (the masking in PDART, a multi-resolution scheme to achieve fast reconstruction), advanced algorithms can be formed using just a short script in either the MATLAB language, or using Python, which is also supported.

Our experimental results demonstrate the relative simplicity by which sophisticated algorithms can be formulated that are directly applicable to an experimental dataset. These experiments are by no means exhaustive, but serve to illustrate the key features of the ASTRA Toolbox relevant in electron tomography, as well as the running times that were obtained using a typical GPU-equipped workstation, and the output that was observed on an experimental HAADF-STEM dataset.

A particular use-case that warrants attention in electron tomography is the use of the geometrical flexibility in advanced alignment schemes. Current alignment methods based on cross-correlation of projection images only provide simple 2D shift and rotation corrections for the projection images, even though the actual geometrical distortions may be more complicated (e.g., involving a tilt-axis that is not exactly in the same plane as the detector). Although marker-based approaches can be used to recover the parameters of such 3D geometry distortions, using these parameters optimally in a reconstruction algorithm requires a fully flexible forward and backprojection implementation, which is typically not available in existing packages. The ASTRA Toolbox forms a highly suitable platform for both the development of advanced alignment algorithms and for the use of the retrieved geometrical parameters in advanced reconstruction methods.

Other operations supported by the ASTRA Toolbox, such as the masking operations for both projection data and for particular sets of image voxels, facilitate the straightforward implementation of a wide range of advanced reconstruction algorithms, including DART and TV Minimization schemes. As all basic operations are capable of achieving high computational performance on modern GPU hardware, these algorithms can subsequently be applied to experimental datasets with relatively low running times.

Despite these advantages for algorithm development and application, the ASTRA Toolbox is by no means a substitute for existing software packages that are currently used in electron tomography. For such use, it currently lacks several key features:

(i) an intuitive user interface that provides easy and graphical operation of the key tomography operations;
(ii) the ability to deal with various file formats common in electron microscopy and tomography;
(iii) implementations of various pre- and postprocessing operations commonly applied in electron tomography (e.g., alignment and denoising).

For these reasons, we foresee that in particular three groups of users will benefit from the ASTRA Toolbox. The first group consists of computationally oriented researchers active in electron microscopy laboratories. Performing cutting edge experiments using advanced microscopy techniques often yields imaging data that is not suitable for processing by standard algorithms. To deal with such data, reconstruction algorithms must be customized for the particular dataset at hand, which may involve writing program code. Such users will benefit greatly from the ASTRA Toolbox, as a high-level platform that still offers a high degree of customization. The second group entails developers of electron tomography software packages. As the ASTRA Toolbox is open source software with a free license for non-commercial use (GPLv3), it can be used as a building block for elaborate software development in the field of electron tomography. Finally, the high-level interfaces offered by the ASTRA Toolbox can bridge the gap that currently exists between researchers in numerical mathematics and imaging on one hand, and experimental users on the other hand. In particular the use of Spot operators, as demonstrated in Section 3.5, allows us to express advanced algorithms in a linear algebra notation (common to mathematics researchers) and to use these algorithms on experimental data. Until now, making this step has typically been difficult to the limitations of standard operations available in high-level numerics packages such as MATLAB.

At present, many operations supported by the ASTRA Toolbox are limited to datasets that fit completely in the memory space of a GPU (at the time of writing available up to 12 GB), which imposes a limitation in handling very large datasets. Current research and implementation efforts are focused on extending the functionality to datasets of much larger sizes, which occur not only in electron tomography, but also in a wide range of other tomography applications (e.g., based on X-ray images).

Acknowledgments

The authors acknowledge financial support from the iMinds ICON MetroCT project, the IWT SBO TomFood project and from the Netherlands Organisation for Scientific Research (NWO), Project no. 639.072.005. Networking support was provided by the EXTREMA COST Action MP1207. Sara Bals acknowledges financial support from the European Research Council (ERC Starting Grant # 335078-COLOURATOMS).

References

[1] J. Gregor, T. Benson, Computational analysis and improvement of SIRT, IEEE Trans. Med. Imaging 27 (7) (2008) 918–924.
[2] J.M. Rebled, L. Yedra, S. Estrade, J. Portillo, F. Peiro, A new approach for 3D reconstruction from bright field TEM imaging: beam precession assisted electron tomography, Ultramicroscopy 111 (2011) 9–10.
[3] M. Weyland, Electron tomography of catalysts, Top. Catal. 21 (4) (2002) 175–183.
[4] S. Bals, G. Van Tendeloo, C. Kisielowski, A new approach for electron tomography: annular dark-field transmission electron microscopy, Adv. Mater. 18 (7) (2006) 892–895.
[5] S.V. Venkatakrishnan, L.F. Drummy, M. Jackson, M. De Graef, J.P. Simmons, C.A. Bouman, A model based iterative reconstruction algorithm for high angle annular dark field scanning transmission electron microscope (HAADF-STEM) tomography, IEEE Trans. Image Process. 22 (11) (2013) 4532–4544.
[6] W. Van den Broek, A. Rosenauer, J. Sijbers, D. Van Dyck, S. Van Aert, A memory efficient method for fully three-dimensional object reconstruction with HAADF STEM ultramicroscopy, Ultramicroscopy 141 (2014) 22–31.
[7] T. Boudier, J.P. Lechaire, G. Frebourg, C. Messaoudi, C. Mory, C. Colliex, F. Gaill, S. Marco, A public software for energy filtering transmission electron tomography (EFTET-J): application to the study of granular inclusions in bacteria from riftia pachyptila, J. Struct. Biol. 151 (2) (2005) 151–159.
[8] L. Roiban, L. Sorbier, C. Pichon, P. Bayle-Guillemaud, J. Werckmann, M. Drillon, O. Ersen, Three-dimensional chemistry of multiphase nanomaterials by energy-filtered transmission electron microscopy tomography, Microsc. Microanal. 18 (5) (2012) 1118–1128.
[9] E. Nakazawa, M. Ogasawara, T. Yotsuji, T. Hashimoto, 3D reconstruction system of H-7650 TEM for tomography and its application to biological specimen, Technical Report 1, Hitachi E.M. News, 2006.
[10] J.R. Kremer, D.N. Mastronarde, J.R. McIntosh, Computer visualization of three-dimensional image data using IMOD, J. Struct. Biol. 116 (1996) 71–76.
[11] D.N. Mastronarde, Dual-axis tomography: an approach with alignment methods that preserve resolution, J. Struct. Biol. 120 (1997) 343–352.
[12] H. Winkler, 3D reconstruction and processing of volumetric data in cryo-electron tomography, J. Struct. Biol. 157 (1) (2007) 126–137.
[13] Q.S. Zheng, M.B. Braunfeld, J.W. Sedat, D.A. Agard, An improved strategy for automated electron microscopic tomography, J. Struct. Biol. 147 (2004) 91–101.
[14] S.Q. Zheng, B. Keszthelyi, E. Branlund, J.M. Lyle, M.B. Braunfeld, J.W. Sedat, D. Agard, Ucsf tomography: an integrated software suite for real-time electron microscopic tomographic data collection and reconstruction, J. Struct. Biol. 157 (1) (2007) 138–147.
[15] C. Messaoudii, T. Boudier, C.O. Sanchez Sorzano, S. Marco, TomoJ: tomography software for three-dimensional reconstruction in transmission electron microscopy, BMC Bioinform. 8 (2007) 288.
[16] S. Nickell, F. Förster, A. Linaroudis, W.D. Net, F. Beck, R. Hegerl, W. Baumeister, J.M. Plitzko, TOM software toolbox: acquisition and analysis for electron tomography, J. Struct. Biol. 149 (3) (2005) 227–234.
[17] S. Phan, A. Lawrence, Tomography of large format electron microscope tilt series: image alignment and volume reconstruction, Congr. Image Signal Process. 2 (2008) 176–182.
[18] W.J. Palenstijn, W.J. Batenburg, J. Sijbers, Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs), J. Struct.

Biol. 176 (2) (2011) 250–253.
[19] W.J. Palenstijn, W.J. Batenburg, J. Sijbers, The ASTRA tomography toolbox, in: 13th International Conference on Computational and Mathematical Methods in Science and Engineering, vol. 4, 2013, pp. 1139–1145.
[20] ⟨http://sourceforge.net/projects/astra-toolbox/⟩.
[21] L. Plantagie, W. van Aarle, K.J. Batenburg, J. Sijbers, Filtered backprojection using algebraic filters; application to biomedical micro-ct data, in: International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, 2015.
[22] W. van Aarle, W. Ludwig, A. King, D. Penumadu, An accurate projection model for diffraction image formation and inversion using a polychromatic cone beam, J. Appl. Crystallogr. 48 (2015) 334–343.
[23] T. Roelandts, K.J. Batenburg, E. Biermans, C. Kübel, S. Bals, J. Sijbers, Accurate segmentation of dense nanoparticles by partially discrete electron tomography, Ultramicroscopy 114 (2012) 96–105.
[24] E.Y. Sidky, X. Pan, Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization, Phys. Med. Biol. 53 (2008) 4777–4807.
[25] J. De Beenhouwer, J. Sijbers, Markerless 3d alignment of an electron tomogram, in: Proceedings of the 7th International Electron Tomography Conference, Cancun, Mexico, 2014.
[26] C.C. Paige, M.A. Saunders, Lsqr: an algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Softw. 8 (1982) 47–489.
[27] P.C. Hansen, Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion, SIAM, 1998.
[28] I. Arslan, J.R. Tong, P.A. Midgley, Reducing the missing wedge: high-resolution dual axis tomography of inorganic materials, Ultramicroscopy 106 (2006) 994–1000 , Proceedings of the International Workshop on Enhanced Data Generated by Electrons.
[29] W. van Aarle, K.J. Batenburg, G. Van Gompel, E. Van de Casteele, J. Sijbers, Super-resolution for computed tomography based on discrete tomography, IEEE Trans. Image Process. 23 (2014) 1181–1193.
[30] B. Goris, W. Van den Broek, K.J. Batenburg, H. Heidari, S. Bals, Electron tomography based on a total variation minimization reconstruction technique, Ultramicroscopy 113 (0) (2012) 120–130.
[31] E.Y. Sidky, J.H. Jørgensen, X. Pan, Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle–Pock algorithm, Phys. Med. Biol. 57 (10) (2012) 3065.
[32] A. Sanchez-Iglesias, M. Grzelczak, T. Altantzis, B. Goris, J. Perez-Juste, S. Bals, G. Van Tendeloo, S.H. Donaldson Jr., B.F. Chmelka, J.N. Israelachvili, L.M. Liz-Marzan, Hydrophobic interactions modulate self-assembly of nanoparticles, ACS Nano 6 (2012) 11059–11065.
[33] T. Altantzis, B. Goris, A. Sanchez-Iglesias, M. Grzelczak, L.M. Liz-Marzan, S. Bals, Quantitative structure determination of large three-dimensional nanoparticle assemblies, Part. Part. Syst. Charac. 30 (2013) 84–88.
[34] J.E. Galvan-Moya, T. Altantzis, K. Nelissen, F.M. Peeters, M. Grzelczak, L.M. Liz-Marzan, S. Bals, G. Van Tendeloo, Self-organization of highly symmetric nanoassemblies: a matter of competition, ACS Nano 8 (2014) 3869–3875.
[35] P.S. Kumar, I. Pastoriza-Santos, B. Rodriguez-Gonzalez, F.J. Garcia de Abajo, L.M. Liz-Marzan, High-yield synthesis and optical response of gold nanostars, Nanotechnology 19 (2008) 015606.
[36] S. Barbosa, A. Agrawal, L. Rodriguez-Lorenzo, I. Pastoriza-Santos, R.A. Alvarez-Puebla, A. Kornowski, H. Weller, L.M. Liz-Marzan, Tuning size and sensing properties in colloidal gold nanostars, Langmuir 26 (2010) 14943–14950.