Fast and flexible X-ray tomography using the ASTRA toolbox

Wim van Aarle,[1] Willem Jan Palenstijn,[2] Jeroen Cant,[1,3] Eline Janssens,[1] Folkert Bleichrodt,[2] Andrei Dabravolski,[1] Jan De Beenhouwer,[1] K. Joost Batenburg,[2] and Jan Sijbers[1,*]

[1] *iMinds-Visionlab, University of Antwerp, Antwerp, Belgium*
[2] *Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands*
[3] *Agfa HealthCare NV, Mortsel, Belgium*
[*] *jan.sijbers@uantwerpen.be*

Abstract: Object reconstruction from a series of projection images, such as in computed tomography (CT), is a popular tool in many different application fields. Existing commercial software typically provides sufficiently accurate and convenient-to-use reconstruction tools to the end-user. However, in applications where a non-standard acquisition protocol is used, or where advanced reconstruction methods are required, the standard software tools often are incapable of computing accurate reconstruction images. This article introduces the ASTRA Toolbox. Aimed at researchers across multiple tomographic application fields, the ASTRA Toolbox provides a highly efficient and highly flexible open source set of tools for tomographic projection and reconstruction. The main features of the ASTRA Toolbox are discussed and several use cases are presented.

OCIS codes: (110.6955) Tomographic imaging; (100.6950) Tomographic image processing; (110.1758) Computational imaging.

References and links

1. E. Perilli, V. Le, B. Ma, P. Salmon, and K. Reynolds, "Detecting early bone changes using in vivo micro-CT in ovariectomized, zoledronic acid-treated, and sham-operated rats," Osteoporos Int. **21**, 1371–1382 (2010).
2. L. B. Christensen, S. G. H. Erbou, M. Vester-Christensen, M. F. Hansen, M. Darré, M. Hviid, and E. V. Olsen, "Optimized Workflow and Validation of Carcass CT-scanning," in *56th International Congress of Meat Science and Technology,* (2010).
3. E. Janssens, J. De Beenhouwer, M. Van Dael, P. Verboven, B. Nicolai, and J. Sijbers, "Neural Network Based X-Ray Tomography for Fast Inspection of Apples on a Conveyor Belt," in *Proceedings of IEEE International Conference on Image Processing,* (IEEE, 2015), pp. 917–921.
4. W. van Aarle, W. J. Palenstijn, J. De Beenhouwer, T. Altantzis, S. Bals, K. J. Batenburg, and J. Sijbers, "The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography," Ultramicroscopy **157**, 35–47 (2015).
5. J. Gregor and T. Benson, "Computational Analysis and Improvement of SIRT," IEEE Trans. Medical Imaging **27**, 918–924 (2008).
6. A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," J. Imaging Sciences **2**, 183–202 (2009).
7. K. J. Batenburg and J. Sijbers, "DART: a practical reconstruction algorithm for discrete tomography," IEEE Trans. Image Processing **20**, 2542–2553 (2011).
8. A. Sheppard, S. Latham, J. Middleton, A. Kingston, G. Myers, T. Varslot, A. Fogden, T. Sawkins, R. Cruikshank, M. Saadatfar, N. Francois, C. Arns, and T. Senden, "Techniques in helical scanning, dynamic imaging and image segmentation for improved quantitative analysis with X-ray micro-CT," Nucl. Instr. Meth. Phys. Res. B **324**, 49–56 (2014).
9. M. Maisl, F. Porsch, and C. Schorr, "Computed Laminography for X-ray Inspection of Lightweight Constructions," in *2nd International Symposium on NDT in Aerospace* (2010), pp. 2–8.
10. A. Dabravolski, K. J. Batenburg, and J. Sijbers, "Adaptive zooming in X-ray computed tomography," J. Xray Sci. Technol. **22**, 77–89 (2014).
11. A. Mirone, E. Brun, E. Gouillart, P. Tafforeau, and J. Kieffer, "The PyHST2 hybrid distributed code for high speed tomographic reconstruction with iterative reconstruction and a priori knowledge capabilities," Nucl. Instr. Meth. Phys. Res. B **324**, 41–48 (2014).

12. T. Dahmen, N. Marsalek, Lukas Marniok, B. Turonova, S. Bogachev, P. Trampert, S. Nickels, and P. Slusallek, "Ettention: building blocks for iterative reconstruction algorithms," in *Proceedings of Microscopy & Microanalysis* (2015).
13. J. Fessler, "Image Reconstruction Toolbox (IRT)," `http://web.eecs.umich.edu/~fessler/code/`.
14. "The ASTRA Toolbox," `http://www.astra-toolbox.com`.
15. D. M. Pelt, D. Gürsoy, W. J. Palenstijn, J. Sijbers, F. De Carlo, and K. J. Batenburg, "Integration of TomoPy and the ASTRA toolbox for advanced processing and reconstruction of tomographic synchrotron data," J. Synchrotron Radiat. **23**, 842–849 (2016).
16. W. J. Palenstijn, K. J. Batenburg, and J. Sijbers, "Performance improvements for iterative electron tomography reconstruction using graphics processing units {(GPUs)}," J. Struct. Biol. **176**, 250–253 (2011).
17. M. A. Helvie, "Digital Mammography Imaging: Breast Tomosynthesis and Advanced Applications," Radiol Clin North Am. **48**, 917–929 (2010).
18. C. B. Reid, M. M. Betcke, D. Chana, and R. D. Speller, "The development of a pseudo-3D imaging system (tomosynthesis) for security screening of passenger baggage," Nucl. Instr. Meth. Phys. Res. A **652**, 108–111 (2011).
19. L. De Chiffre, S. Carmignato, J. Kruth, R. Schmitt, and A. Weckenmann, "Industrial applications of computed tomography," CIRP Annals - Manufacturing Technology **63**, 655–677 (2014).
20. U. Ewert, B. Redmer, C. Rädel, U. Schnars, R. Henrich, K. Bavendiek, and M. Jahn, "Mobile Computed Tomography for Inspection of Large Stationary Components in Nuclear and Aerospace Industries," Materials Transactions **53**, 308–310 (2012).
21. S. Van Bael, G. Kerckhofs, M. Moesen, G. Pyka, J. Schrooten, and J. P. Kruth, "Micro-CT-based improvement of geometrical and mechanical controllability of selective laser melted Ti6Al4V porous structures," Mater. Sci. Eng. A **528**, 7423–7431 (2011).
22. B. Goris, J. De Beenhouwer, A. De Backer, D. Zanaga, K. J. Batenburg, A. Sánchze-Iglesias, L. M. Liz-Marazàn, S. Van Aert, S. Bals, J. Sijbers, and G. Van Tendeloo, "Measuring Lattice Strain in Three Dimensions through Electron Microscopy," Nano Lett. **15**, 6996–7001 (2015).
23. E. Y. Sidky, J. H. Jørgensen, and X. Pan, "Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle Pock algorithm," Phys. Med. Biol. **57**, 3065–3091 (2012).
24. W. van Aarle, P. Ghysels, J. Sijbers, and W. Vanroose, "Memory access optimization for iterative tomography on many-core architectures," in *Proceedings of 12th Int. Meeting on Fully 3D Image Reconstruction in Radiology and Nuclear Medicine,* (2013), pp. 364–367.
25. E. van den Berg and M. P. Friedlander, "Spot: a linear-operator toolbox," `http://www.cs.ubc.ca/labs/scl/spot`.
26. F. Bleichrodt, T. van Leeuwen, W. J. Palenstijn, W. van Aarle, J. Sijbers, and K. J. Batenburg, "Easy implementation of advanced tomography algorithms using the ASTRA toolbox with Spot operators," Numer. Algorithms **71**, 673–697 (2015).
27. D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "TomoPy : a framework for the analysis of synchrotron tomographic data," J. Synchrotron Radiat. **21**, 1188–1193 (2014).
28. N. Viganò, A. Tanguy, S. Hallais, A. Dimanov, and M. Bornert, "Three-dimensional full-field X-ray orientation microscopy," Sci. Rep. **6**, 1–9 (2016).
29. S. Cools, P. Ghysels, W. van Aarle, J. Sijbers, and W. Vanroose, "A multi-level preconditioned Krylov method for the efficient solution of algebraic tomographic reconstruction problems," J. Comput. Appl. Math. **283**, 1–16 (2015).

1. Introduction

The use of X-ray Computed Tomography (CT) is not limited to its most well-know application in medical diagnostics. Indeed, its ability to look at the interior of certain objects benefits many other application fields as well. Examples include biomedical research, that often relies on micro-CT scanning setups to investigate the effect of certain substances or medicines on small animals [1]; industrial applications, where tomographic reconstruction can be an invaluable tool for quality control of (semi-)automatic processes on a conveyor belt [2, 3]; chemistry, where it can be used to study and optimize reactions and their results (e.g., of polyurethane foams); and material science with various imaging modalities such as micro-CT, synchrotron radiation facilities, and electron microscopes [4]. While all of these applications share the same basic concept (multiple projection images are measured from different directions), the specific setup of the scanning system can differ from application to application.

Typically, reconstruction software comes bundled with the purchase of the scanner (e.g. NRecon for Bruker micro-CT systems, Phoenix datos|x for GE systems, and Inspect3D for FEI electron microscopes). A few commercial products are also available that are not tightly coupled to a specific scanning system, but can come at a premium rate (e.g., VG Studio Max, Octopus

Reconstruction, Ikeda's TomoShop). The main advantages of these commercial packages are their reliability due to their large user-base and their user-friendly graphical user interfaces. For the majority of CT users these tools will therefore be well sufficient. However, the implementations provided by these packages offer only a limited flexibility, restricting some use cases:

- The choice of reconstruction algorithm is typically limited to Filtered Backprojection (FBP), Feldkamp-Davis-Kress (FDK), and in some cases the Simultaneous Iterative Reconstruction Technique (SIRT) [5]. However, other reconstruction methods (such as FISTA) may offer better results, especially in cases with low statistics (highly noisy) projection data [6]. Moreover, a priori information of the scanned object cannot be easily integrated to improve reconstruction quality. Such prior knowledge is especially useful when dealing with limited data (e.g., a low number of projections, a limited angular range, truncated projection data, etc.) [7].

- In most CT systems the X-ray source and detector follow a predefined circular trajectory around the object in a plane perpendicular to the axis of rotation. In many applications however, the X-ray source and detector array have to follow a non-conventional trajectory, which means their projection data can not be easily reconstructed using conventional software. For example, helical trajectories are nowadays not only used in medical systems, but also in μ-CT systems for rock analysis [8]. Also, in a typical tomosynthesis setup, the detector panel is positioned underneath a patient and the X-ray source follows a linear trajectory. Similarly, in laminography, the source and detector are respectively fixed above and underneath a thin object, which is then tilted and rotated [9]. The trajectory is thus circular, but not perpendicular to the axis of rotation. Furthermore, in quality control of industrial processes, projections can only be acquired from a few arbitrary perspectives, and even when projections can be taken from all directions on a circular trajectory, it can be beneficial to change the magnification during the scan of objects that are elongated in one direction [10].

- The computational efficiency of the implemented algorithms may be limited, resulting in long reconstruction times, especially for large datasets. While the computational power of reconstruction frameworks keeps increasing, so does the size and resolution of X-ray detector panels.

For users of non-conventional CT systems the limitations dictated by commercial software packages can thus be a severe hindrance while prototyping new concepts and ideas.

In Table 1, a short overview is given of different open source software packages for tomographic reconstruction that have recently been updated: pyHST2 [11], Ettention [12], and IRT [13]. While each of these packages provide excellent implementations of the most common tomographic routines, it can be noted that each package is focused on a specific target application. To the authors knowledge, there is currently no software package available that is more suited for the development of advanced algorithms that can be used in multiple application fields, using non-conventional geometries and various constraints.

The *ASTRA Toolbox* (All Scales Tomographic Reconstruction Antwerp) introduced in this article is a software platform to address the need for a flexible, efficient, and easy to use development platform for tomographic algorithms. Since 2010, it has been developed by the Vision Lab, University of Antwerp, Belgium, and since 2014 jointly by Vision Lab and the Centrum Wiskunde & Informatica (CWI) in Amsterdam, The Netherlands. It is freely available as open source software under a GPLv3 license [14]. To achieve optimal efficiency, most of its code is written in C++, with its core computations offloaded to a GPU card using the CUDA language. Users with no access to GPU workstations can resort to CPU code, accelerated using OpenMP. The toolbox is accessible through a MATLAB and a Python interface, providing a fast and powerful platform for algorithm and application prototyping.

Table 1. Overview of different open source toolboxes for tomographic reconstruction.

package	**pyHST2** [11]	**Ettention** [12]	**IRT** [13]	**ASTRA**
Target applications	synchrotron (ESRF)	electron tomography	PET, SPECT, X-ray CT	all non-medical CT
Algorithms	analytical, iterative dictionary learning, FISTA, ...	analytical, iterative through plugins	analytical, iterative	analytical, iterative
Interface	Python	command line, GUI through IMOD	MATLAB	MATLAB, Python
Hardware acceleration	MPI, CUDA	OpenCL	OpenCL, CUDA	OpenMP, CUDA

The ASTRA Toolbox is mainly intended for use in research (e.g., application prototyping and algorithm development), and less so for day to day usage by end-users. However, due to its open nature, the ASTRA Toolbox can be coupled with existing practical software packages. One example is its inclusion into TomoPy [15], which combines the user-friendliness and efficient pre-and postprocessing methods of TomoPy with the flexibility and optimized reconstruction methods of the ASTRA toolbox into a single framework. Also, for applications in the field of electron tomography, the ASTRA Toolbox is available through FEI's Inspect3D software.

The main strength of the ASTRA Toolbox is its inherent flexibility, which comes in three forms:

- With its *broad support*, various non-conventional acquisition geometries are supported.
- With its *modular design*, each small building block of the toolbox (e.g., the projection operations) can be used independently such that they can be combined in other, existing, codebases. Also, the ASTRA Toolbox can be used in multiple application fields and implementations of advanced reconstruction methods that were made for a specific application using the basic ASTRA building blocks can be easily applied in other application fields as well.
- With its *scalability*, reconstruction methods and their parameters can be quickly evaluated on small-size problems with a local computation platform. Afterwards, the resulting reconstruction strategy can then, with relatively little effort, be deployed on large-scale computation clusters for large, high-resolution datasets.

This article provides an overview of the ASTRA Toolbox, its design, and some of its use-cases. Firstly, in Section (2), the key components of the toolbox are introduced. Then, Section (3) demonstrates how they can be used in a conventional cone beam setup. The subsequent two sections then delve deeper into some key advanced features of the toolbox that allow it to be used flexibly: Section (4) concerns the specification of non-conventional scanning setups, and Section (5) discusses how the ASTRA Toolbox can be easily integrated into external codebases. Finally, Section (6) concludes this work.

2. Framework concepts

There are seven different concepts and modules in the ASTRA Toolbox (Fig. 1). Volume and projection data objects (Section (2.1)) are linked to their corresponding geometry information (Section (2.2)), both of which are linked together in a projection algorithm building block (Section (2.3)), which is subsequently used together with the data in the reconstruction algorithm object (Section (2.4)).

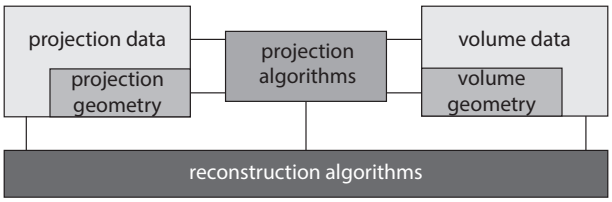
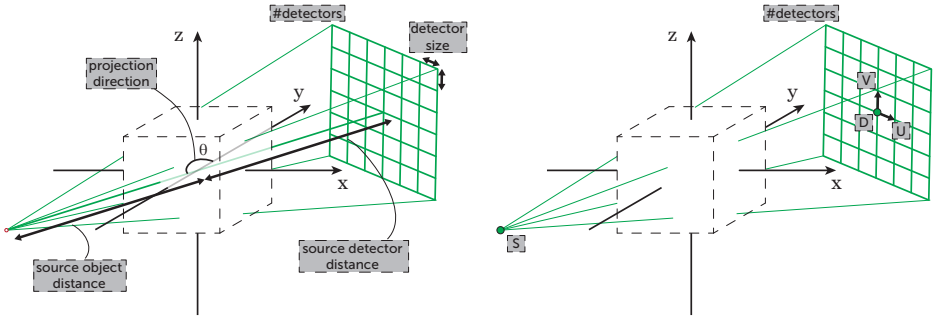


Fig. 1. Schematic overview of all ASTRA Toolbox concepts.

2.1. Projection and volume data

The data modules of the ASTRA Toolbox store and manage all datasets, both for volume data and for projection data. In order to perform actions on this data (e.g., a projection or a reconstruction) the toolbox first needs to have access to it, which is not the case if the data is stored only in the MATLAB or Python environment. Thus, memory first has to be allocated and data has to be copied from the interface layer to the actual toolbox. This operation then returns a unique identifier that can subsequently be used to refer to that particular set of data, much like how file input/output works in many programming languages.

2.2. Geometry



(a) default (b) vector-based

Fig. 2. Two approaches to the specification of a cone-beam projection geometry.

The experimental setup of the scanning system is specified by linking the data objects to a corresponding *volume geometry* or *projection geometry*. The volume geometry describes the pixel or voxel grid on which the reconstructed object is represented. A voxel is typically isotropic but can be specified to anisotropic as well. This is useful in cases where the effective resolution in a reconstruction is not the same in each dimension (e.g., tomosynthesis and laminography).

The projection geometry defines the trajectory of the X-ray source and detector relative to the volume geometry. This involves the number and size of the detectors pixels and the specification of the trajectory of the X-ray source and detector as they rotate around the object. Standard projection geometries that are available are parallel beam and fan beam for two-dimensional data, and parallel beam and cone beam for three-dimensional data.

Specifying a projection geometry in the ASTRA Toolbox can be done in two ways. In its default form, the source and detector follow a circular trajectory around the origin of the volume coordinate system and the detector plane lies perpendicular (and perfectly aligned) to the line through the source and the origin. Hence, for a circular cone-beam geometry, there are five sets of parameters to be specified: the number of detector pixels, the detector pixel size, the distance

from the source to the centre of rotation, the distance from the centre of rotation to the centre of the detector and a list of projection directions (Fig. 2(a)). With this approach, the most widely used projection setups can be specified, but non-conventional, non-circular trajectories can not. Moreover, it is not possible to model common small detector misalignments, such as a detector tilt or detector shift (when the centre of the detector does not line up with line through the source and the centre of rotation), without resorting to interpolation in the projection domain, a process which will introduce inaccuracies in the reconstructed images and should thus at all times be avoided.

Instead of specifying all these parameters in general with a fixed object source distance and detector size and instead of assuming that the X-ray source nicely rotates around the centre of rotation, the ASTRA Toolbox also allows users to specify each projection separately. This is called a *vector-based geometry* (Fig. 2(b)). As an example for a cone-beam projection model, each projection image is then defined by four 3D vectors in the volume coordinate system:

- The location of the source, $\boldsymbol{S} = (s_x, s_y, s_z)$.
- The centre of the detector panel, $\boldsymbol{D} = (d_x, d_y, d_z)$.
- The principal axes of the detector plane, $\boldsymbol{U} = (u_x, u_y, u_z)$ and $\boldsymbol{V} = (v_x, v_y, v_z)$. The length of these vectors corresponds with the size of each detector pixel.

From a user perspective, specifying a projection geometry like this is more difficult as the computation of the source and detector trajectory has to be carefully specified. It does, however, allow much more advanced setups. Moreover, detector alignment can be modeled in the reconstruction process without requiring interpolation in the projection domain. For example, a detector tilt can be modelled by rotating $\boldsymbol{U}$ and $\boldsymbol{V}$ accordingly around the line $\boldsymbol{SD}$ in each projection; and a detector shift can be modelled by translating $\boldsymbol{D}$ in the detector plane (specified by $\boldsymbol{U}$ and $\boldsymbol{V}$).

2.3. Projection operations

The most important building blocks of any reconstruction technique are the projection operations. Consider tomography in its algebraic form, the *forward projection* operator then simulates projection images from a 2D or 3D volume by computing a sparse matrix vector multiplication (SpMV):

$$\boldsymbol{p} := \boldsymbol{W}\boldsymbol{v}, \tag{1}$$

in which $\boldsymbol{p}$ is a vector representing 2D or 3D projection data, $\boldsymbol{v}$ is a vector representing 2D or 3D volume data, and $\boldsymbol{W}$ is the projection matrix and is defined by both the volume geometry and the projection geometry. The *backprojection* operation "smears" the projection data through a reconstruction volume by computing:

$$\boldsymbol{v} := \boldsymbol{W}^T\boldsymbol{p}. \tag{2}$$

The projection building blocks in the ASTRA Toolbox is where the values w_{ij} of $\boldsymbol{W}$ are calculated and used to compute projection and backprojection images. As these are the operations that typically make up most of the computation time and power of a tomographic algorithm, much effort was put into making these implementations as computationally efficient as possible. To achieve this, the ASTRA Toolbox uses the computational power provided by GPU cards through the NVIDIA CUDA language [16]. In case the user has no GPU cards available, OpenMP accelerated CPU implementations of the 2D versions of these building blocks are also provided.

2.4. Reconstruction algorithms

All tomographic reconstruction algorithms are built on top of the basic projection and back-projection building blocks. These building blocks are internally configured by the algorithm

itself and thus need no intervention of the user. Reconstruction methods that are available in the ASTRA Toolbox include the analytical reconstruction methods FBP for parallel and fan beam and FDK for cone beam datasets, several algebraic reconstruction methods such as SIRT, SART, and the Krylov subspace method CGLS. All of these are available both for 2D and 3D datasets, and also allow for some optional customization such as reconstruction limited to pixels in a certain mask, or enforcing minimum/maximum constraints during iterations. Configuration of an algorithm can be done in the MATLAB or Python interface by linking it to the correct data identifiers and by setting some algorithm specific options. It is important to note that the forward- and backprojection building blocks themselves can also be directly accessed from within the external interfaces. 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.

3. Conventional cone beam example

This section demonstrates the interface of the ASTRA Toolbox for a user wishing to perform a simple SIRT reconstruction of a set of conventional cone beam projection images. The code examples that are provided make use of the MATLAB interface to the ASTRA Toolbox. The Python interface uses the same design and interface principles and differs only in its syntax.

It is assumed that projection data is already stored as a 3D array in the MATLAB environment. As each CT machine utilizes different file formats to store projection images and geometry information, 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.

1. Firstly, the projection and volume geometries are defined. Here, a $600 \times 600 \times 300$ isotropic voxel grid is considered, with the size of each voxel being $1 \times 1 \times 1$mm. The centre of this volume is located at its central slice, at the axis of rotation. The projection geometry is specified with the parameters depicted in Fig. 2(a). Each projection is recorded as a 512×512 image with pixel size 0.7×0.7mm. The source-to-object, and object-to-detector distances are 1000mm and 1500mm, respectively. In total, 360 projection images are defined over a full angular range $[0, 2\pi)$.

```
vol_geom = astra_create_vol_geom(600,600,300);
proj_geom = astra_create_proj_geom('cone', 0.7, 0.7, 512, 512, ...
                                   linspace(0,2*pi,360), 1000, 1500);
```

2. Projection data is copied into the toolbox memory and space is allocated store the reconstruction in. The result of these operations is two data identifiers which are used in subsequent steps of the workflow. Each data object is linked to its corresponding geometry object.

```
vol_id = astra_mex_data3d('create', '-vol', vol_geom, 0);
proj_id = astra_mex_data3d('create', '-proj3d', proj_geom, p); % 'p' contains the
                                                                 projection data
```

3. With the data accessible to the toolbox, a reconstruction algorithm object can be configured. Consider here a CUDA accelerated SIRT implementation for 3D data problems. This algorithm internally configures the projection building blocks that it will use. For this specific case, a non-negativity constraint is also applied. This is a simple form of prior knowledge that can lead to improved reconstruction quality. The result of this configuration is an identifier referencing to the algorithm object.

```
cfg = astra_struct('SIRT3D_CUDA');
cfg.ProjectionDataId = proj_id;
cfg.ReconstructionDataId = vol_id;
```

```
cfg.option.MinConstraint = 0;
alg_id = astra_mex_algorithm('create', cfg);
```

4. The algorithm identifier is used to perform 100 iterations.

```
astra_mex_algorithm('iterate', alg_id, 100);
```

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

```
reconstruction = astra_mex_data3d('get', vol_id);
```

4. Non-conventional geometries

This section explores a few distinct applications of CT with a non-conventional projection geometry. The specification of each of these can be done in the ASTRA Toolbox using the vector based approach as described in Section (2.2) and Fig. 2(b). Specifically, with l the total number of projection images, an $l \times 12$ matrix is constructed in which each row represents the geometry of a single projection. The first three columns represent the position of the X-ray source ($\boldsymbol{S}$), the fourth to sixth column represent the position of the centre of the detector array ($\boldsymbol{D}$), and columns seven to nine and ten to twelve represent both unit detector vectors ($\boldsymbol{U}$ and $\boldsymbol{V}$). This matrix and the number of detectors are then supplied to the toolbox when building a cone beam projection geometry (Table 2).

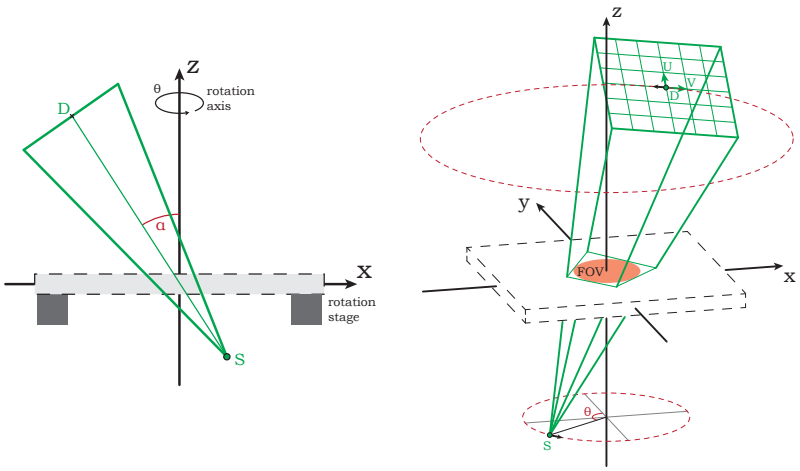
Table 2. MATLAB code demonstrating how a conventional cone beam setup can be specified by a vector-based geometry object. The variable *theta* represents a $l \times 1$ vector of projection directions; *SOD* and *ODD* represent the source-to-object and object-to-detector distance respectively; *detWidth* and *detHeight* represent the size of each detector pixel; and *detCol* and *detRows* represent the number of detectors in each projection image.

```
vectors = zeros(l,12);
vectors(:,1:3) = [sin(theta) -cos(theta) 0] * SOD; % S
vectors(:,4:6) = [-sin(theta) cos(theta) 0] * ODD; % D
vectors(:,7:9) = [cos(theta) sin(theta) 0] * detWidth; % U
vectors(:,10:12) = repmat([0 0 detHeight], [l 1]); % V
proj_geom = astra_create_proj_geom('cone_vec', detCols, detRows, vectors);
```

4.1. Rotary computed laminography

In conventional cone-beam CT, the object is imaged in a circular trajectory. However, for large flat objects, such as circuit boards and paintings, this is not possible as they might not physically fit longitudinally between the source and detector, or because the total X-ray attenuation in that direction might be too high. In those cases *rotary computed laminography* can offer a solution by tilting the object and rotation stage such that the rotation axis is no longer perpendicular to the plane defined by the point source and the central row of the detector (Fig. 3) [9]. This is equivalent to a setup with a fixed object, where the source and detector follow a circular trajectory parallel to the $z = 0$ plane. The radius of these circles depends on the angle α by which the object is tilted, and on the object-to-source or object-to-detector distance. The choice of α also represents a trade-off between a small *field of view (FOV)* (small α) and long intersections of the X-ray beam with the object and thus possibly noisy projection statistics (large α). For $\alpha = 90°$, this setup is equivalent to a conventional cone-beam setup.

Representing this type of projection geometry in the ASTRA Toolbox is relatively straightforward. Define Δt as the size of each detector pixel, Δs as the distance from the source to the centre

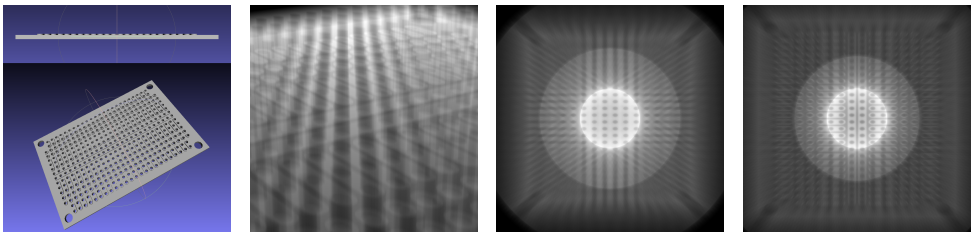


(a) (b)

Fig. 3. In rotary computed laminography, the object is tilted such that its axis of rotation is not perpendicular to the plane defined by the X-ray source and the central row of the detector plane. This is equivalent to a setup with a fixed object where the source and detector follow a circular trajectory parallel to the $z = 0$ plane. (a) Side-view. Note how the rotation stage must have an opening for the X-ray beam to pass through. (b) Three-dimensional sketch of the projection geometry.

of the coordinate system, and Δd as the distance from the centre of the detector array to the centre of the coordinate system. Let $i \in \{1, \ldots, l\}$ denote the projection index and let θ_i denote the viewing perspective corresponding to each projection. The full projection geometry is then:

$$
\begin{aligned}
\boldsymbol{S}_i &= (\Delta s \sin\alpha \sin\theta_i, -\Delta s \sin\alpha \cos\theta_i, -\Delta s \cos\alpha)\,, \\
\boldsymbol{D}_i &= (-\Delta d \sin\alpha \sin\theta_i, \Delta d \sin\alpha \cos\theta_i, -\Delta d \cos\alpha)\,, \\
\boldsymbol{U}_i &= (\Delta t \cos\theta_i, \Delta t \sin\theta_i, 0)\,, \\
\boldsymbol{V}_i &= (\Delta t \cos\alpha \sin\theta_i, -\Delta t \cos\alpha \cos\theta_i, \Delta t \sin\alpha)\,.
\end{aligned}
$$



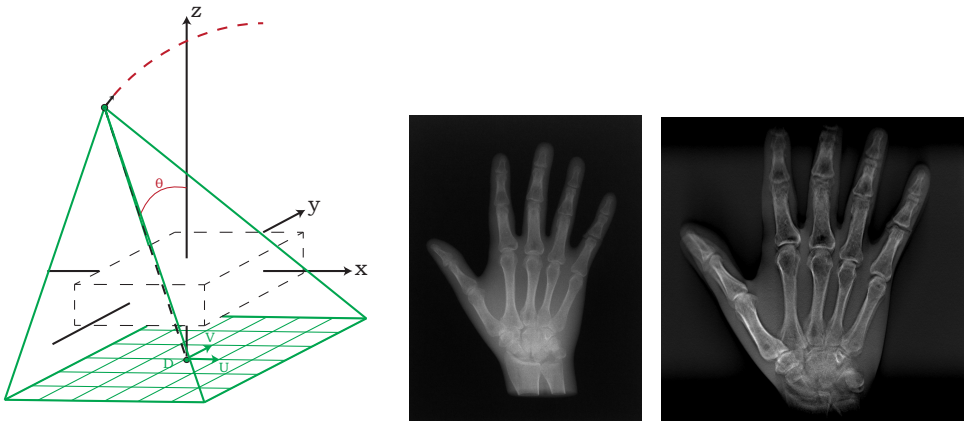
(a) CAD model (b) projection image (c) middle slice (d) top slice

Fig. 4. Simulation study of a laminographic reconstruction of a large flat object. The projection data (b) was simulated using the NVIDIA Optix raytracing toolbox.

In Fig. 4, an example is shown of a laminographic reconstruction of a large, yet flat, object. The CAD model depicted in Fig. 4(a) represents a 1000 × 1000mm circuit board with an array of holes with protruding edges drilled into it. In total, 270 800 × 800 projections (Fig. 4(b)) were

simulated of this CAD model using the NVIDIA Optix raytracing framework. The laminographic tilt was set to $\alpha = 50°$, the source-to-object distance to $\Delta s = 750mm$, and a object-to-detector distance to $\Delta s = 1500mm$. A realistic level of Poisson noise was applied to the projection images. A 300 iteration CGLS reconstruction was subsequently performed using the ASTRA Toolbox, of which two slices are shown in Figs. 4(c) and 4(d). It is clear that only a small portion of the object falls into the field of view.

4.2. Tomosynthesis



(a) tomosynthesis geometetry (b) projection image (c) reconstructed slice

Fig. 5. In tomosynthesis, a stationary detector is placed underneath the object. The X-ray source moves on a circular trajectory above the object, creating a set of projections from a limited angle.

Tomosynthesis is an imaging technique mainly used in medical radiography when only a few limited-angle projection images can be taken and when the minimization on X-ray dose is of crucial importance. Examples include breast cancer screening, where tomosynthesis has a higher accuracy compared to conventional mammography [17]; and portable, in-bed, radiography for patients in intensive care units that can not be positioned in a CT or MRI scanner. Tomosynthesis can also be applied for non-medical purposes, such as baggage screening in airports [18].

In tomosynthesis, projections are acquired with an X-ray source which moves on a linear or circular trajectory above the object, and a stationary flat panel detector. An example is illustrated in Fig. 5(a). In this geometry, only a limited angle can be covered by the acquisitions resulting in a depth resolution that is lower than the in-plane resolution. Therefore, the reconstructed volume is typically formed by anisotropic voxels, with a voxel size which is several times higher in the in z-direction compared to the x- and y-direction.

Let $i \in \{1, \ldots, l\}$ denote the projection index and let θ_i denote the acquisition angle corresponding to each projection. The detector plane is in a fixed position, centred underneath the object centre at distance Δd. The full projection geometry is then defined by:

$$\begin{aligned} \boldsymbol{S}_i &= (\Delta s \sin\theta_i, 0, \Delta s \cos\theta_i) \\ \boldsymbol{D}_i &= (0, 0, -\Delta d) \\ \boldsymbol{U}_i &= (1, 0, 0) \\ \boldsymbol{V}_i &= (0, 1, 0) \end{aligned}$$

Consider the example shown in Fig. 5(b). A total of 45 tomosynthesis projection images were

acquired of a human hand. The projection angles θ_i were equally distributed over an angular range of $[-20°, 20°]$. The object centre was placed at a distance of $\Delta d = 65$mm above the detector panel and the object-to-detector distance was $\Delta s = 1039$mm. The detector counted 2208×2668 pixels of size 0.160μm. Images were acquired at 48kV and 0.32mAs. These projection images were reconstructed (Fig. 5(c)) using 100 SIRT iterations on a volume containing $552 \times 667 \times 30$ anisotropic voxels of size $(0.640, 0.640, 4.0)$mm.

4.3. Conveyor belt tomography

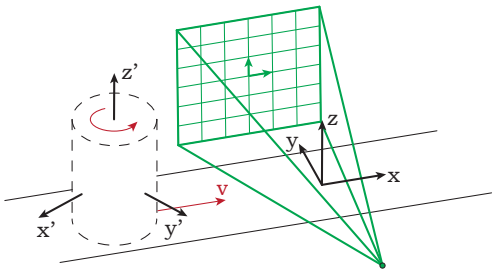


Fig. 6. Conveyor belt geometry of an object moving on a conveyor belt with a fixed source and flat-panel detector.

In industrial applications, accurate quality control is often crucial in order to maximize profit and customer satisfaction. *Non-destructive testing* using computed tomography can be a valuable tool to achieve that [19]. In manufacturing, CT can be used to inspect the quality of welding seams [20], 3D printed objects [21], printed circuit boards, etc. In the food processing industry, using CT, foodstuffs that do not meet certain requirements (e.g., apples with early rotting on the inside) can be discarded before shipping [3], and the cutting of meat (e.g., pig carcasses) can be optimized when it is known where the meat, fat and bones are located [2].

For fast inspection of objects in a mass production setting (a production line), the use of a conventional CT is not convenient. Placing every object separately in a CT scanner is both expensive and very time consuming. In order not to interfere with the efficiency of the production line, inline tomography can be used. Here, an object moves on a conveyor belt where a fixed X-ray source is positioned on one side, and a fixed flat-panel detector is placed opposite to it. While the object translates on the conveyor belt with a speed v, it passes through the field of view of the source and detector, and multiple cone beam projection images can be recorded. To allow projections from the full angular range, also consider a rotation of the object around its central axis with rotation speed ω, perpendicular to the conveyor belt (Fig. 6). In the fixed coordinate system $< x, y, z >$, this geometry is defined by:

$$\boldsymbol{S} = (0, -\Delta s, 0)\,, \quad \boldsymbol{D} = (0, \Delta d, 0)\,, \quad \boldsymbol{U} = (\Delta u, 0, 0)\,, \quad \boldsymbol{V} = (0, 0, \Delta u)\,.$$

Instead of translating and rotating the object while keeping the source/detector system fixed, the same setup can also be described from the point of view of a fixed object. In this case the source and detector translate and rotate around the object. The origin of the fixed system is located at the centre on the crossing of the path of the object and the line that connects the source and the centre of the detector. For every time point i, i.e. every position of the object on the conveyor belt, let t_i denote the time passed since the first projection, and define R_i as the rotation matrix

$$\boldsymbol{R}_i = \begin{pmatrix} \cos(\omega t_i) & -\sin(\omega t_i) & 0 \\ \sin(\omega t_i) & \cos(\omega t_i) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The virtual projection geometry in function of the fixed object coordinate system $< x', y', z' >$ can then be expressed as follows:

$$\begin{aligned} \boldsymbol{S}_i' &= \boldsymbol{R}_i\boldsymbol{S} - (A_0 + vt_i)\boldsymbol{R}_i\boldsymbol{U}, \\ \boldsymbol{D}_i' &= \boldsymbol{R}_i\boldsymbol{D} - (A_0 + vt_i)\boldsymbol{R}_i\boldsymbol{U}, \\ \boldsymbol{U}_i' &= \boldsymbol{R}_i\boldsymbol{U}, \\ \boldsymbol{V}_i' &= \boldsymbol{R}_i\boldsymbol{V}, \end{aligned}$$

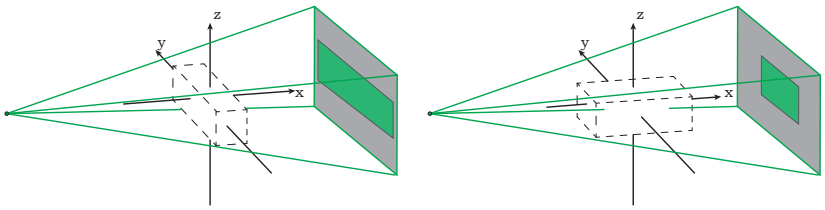
where A_0 is the distance between the origin of the $< x, y, z >$ and the $< x', y', z' >$ coordinate systems at the first projection.

(a) projection image (b) reconstruction with$\omega = 0°/\text{ms}$ (c) reconstruction with $\omega = -0.36°/\text{ms}$

Fig. 7. Reconstruction of a toy in an inline scanning geometry.

In Fig. 7, an example is shown of an inline reconstruction of a toy. The inline projection data is derived from 493 projections of a conventional circular CT scan (Fig. 7(a)). After conversion, 100 524 $\times$ 3000 projections were obtained at a 10ms interval, with source-to-object distance Δs = 121mm, object-to-detector distance Δd = 40mm, detector pixel size $\Delta u = 35.36\mu$m, a conveyor belt velocity of $v = 100\mu$m/ms, an object rotation speed of $\omega = -0.36°/\text{ms}$, and A_0 = -50mm. Fig. 7(b) shows a reconstruction of the central slice without the rotation of the object on the conveyor belt (i.e. $\omega = 0°/\text{ms}$) and Fig. 7(c) shows the same reconstruction but with rotation $\omega = -0.36°/\text{ms}$. It is clear that the extra rotation is required to obtain an accurate reconstruction.

4.4. Adaptive zooming



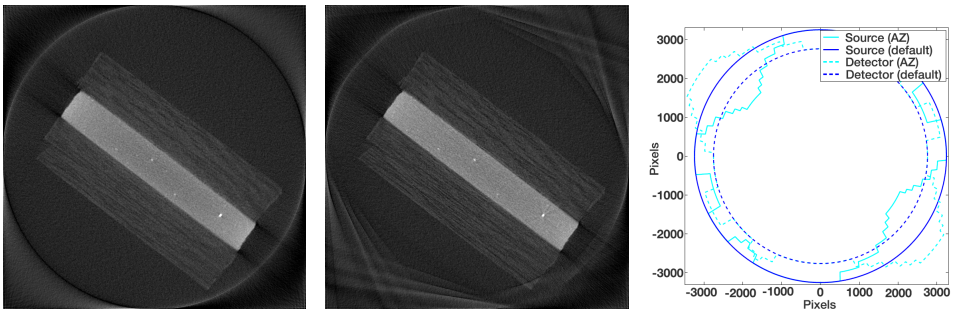
(a) (b)

Fig. 8. For elongated objects, a fixed source-to-detector distance leads to suboptimal use of the detector array in some viewing directions.

In a cone beam geometry, the source-to-detector distance is sometimes referred to by the *zooming factor*, since the closer the source is to the object, the larger it appears on the detector. Ideally, it is chosen such that a projection of the object just fills the detector array. That way, the projection image has the highest achievable resolution without inducing object truncation and region-of-interest artifacts. Consider, however, an elongated object, such as in Fig. 8. In conventional setups, the source-to-object distance should, due to physical constraints, be chosen with respect to the orientation in which the longest axis of the object lines up with the source-to-detector line. This guarantees the best resolution in that direction (Fig. 8(a)), but leads to suboptimal detector usage in other projections (Fig. 8(b)).

With an *adaptive zooming* approach, the idea is to move the position of the object closer to or farther away from the source as the scan is being taken. The most optimal way of doing so is by precomputing the trajectory from the convex hull of the scanned object [10]. This approach allows to acquisition of more information about the object, thus increasing the reconstruction quality.

To mimic a tomographic system with variable source and detector position, an experiment was conducted on a desktop micro-CT system SkyScan-1172 (Bruker-MicroCT, Belgium). A piece of a pencil with a diameter of 7mm and a length of 15mm was used as an elongated object. Seven full-angle datasets were obtained, each containing 600 projections of 880×666, with the source-to-object distances ranging from 80.77 to 117.01mm. The source-to-detector distance was fixed at 216.392mm.



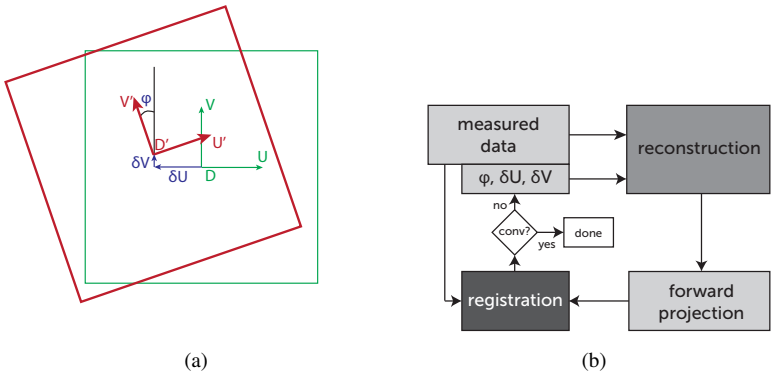
(a) conventional geometry (b) adaptive zooming (c) adaptive trajectory

Fig. 9. Reconstructions of an elongated object (a piece of pencil) with a conventional and an adaptive zooming projection geometry. [10]

In Fig. 9(a), a 700 iteration SIRT reconstruction is shown from the largest source-to-object distance. Based on this reconstruction, a convex hull of the object was obtained. For each projection angle the closest possible source position was calculated according to the technique described in [10], and a projection was chosen from the dataset obtained from the smallest distance bigger than or equal to the distance from the calculated source position to the centre of rotation. The resulting trajectories are presented in Fig. 9(c), where is can be seen that the zooming factor is increased for the angles in which the object has a narrow shadow. The corresponding reconstruction is shown in Fig. 9(b), where the adaptive zooming approach results in a slightly improved contrast.

4.5. Automatic projection alignment

Another advantage of considering the projection geometry in its vector form is its inherent flexibility when it comes to correcting for projection misalignment. Such misalignments may come in many forms. For example, the centre of rotation might not project exactly to the centre of



(a) (b)

Fig. 10. (a) A shifted and tilted detector array, seen from the point of view of the X-ray source. (b) Schematic overview of an iterative procedure to automatically estimate detector shift and tilt.

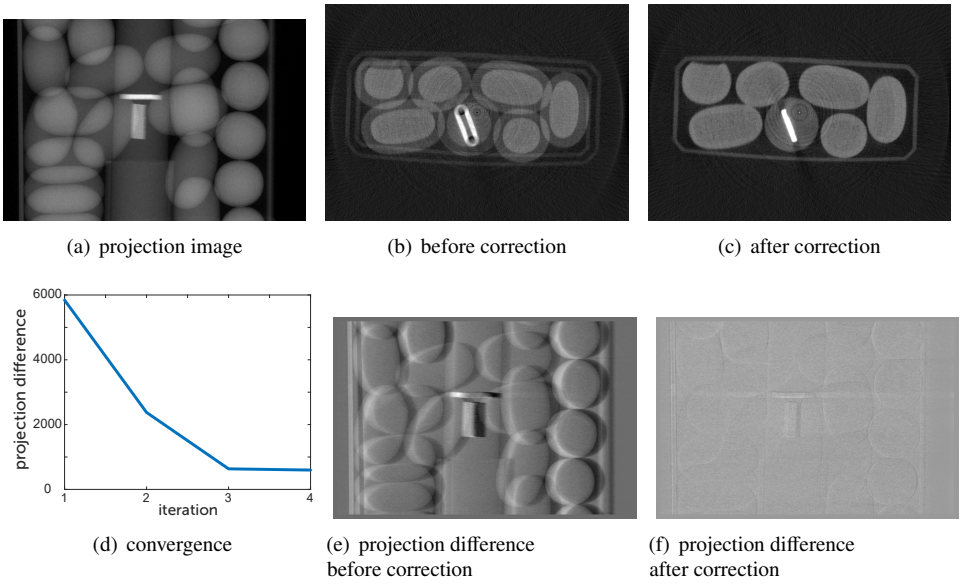
the detector array and the detector plane may not be exactly perpendicular to the line connecting the source and the centre of rotation. Moreover, the rotation stage might have small inaccuracies, the focal spot size might not be constant, and the scanned object may move during the scan. All of these misalignments can be modelled simply by altering the vectors of the projection geometry. However, to do so, one first needs to know exactly which misalignments have occurred during the scan, and by how much. Recently, it has been shown that it is possible to automatically find these misalignment parameters by optimizing the vectors in the projection geometry such that the projection difference between the measured data and simulated data of the reconstructed images is minimal [22].

Here, we consider a simplified version of that work, focused on only a subset of misalignment factors: the detector offset δU and δV, and detector tilt ϕ (Fig. 10(a)). These misalignments generally result in double edges, such as in Fig. 11(b). When the offset and tilt factors are known, one can first translate and rotate the projection images, and subsequently continue with a reconstruction with a conventional geometry. However, instead of changing the data to fit the geometry, which requires interpolation in the projection domain, it is better to change the geometry to fit the data. Let $\boldsymbol{R}_{i,\phi}$ denote the rotation matrix around the vector $\boldsymbol{D}_i - \boldsymbol{S}_i$ with angle ϕ. The corrected projection geometry can then be expressed by:

$$\begin{aligned} \boldsymbol{S}'_i &= \boldsymbol{S}_i, \\ \boldsymbol{D}'_i &= \boldsymbol{D}_i + \delta U \boldsymbol{U}_i + \delta V \boldsymbol{V}_i, \\ \boldsymbol{U}'_i &= \boldsymbol{R}_{i,\phi} \boldsymbol{U}_i, \\ \boldsymbol{V}'_i &= \boldsymbol{R}_{i,\phi} \boldsymbol{V}_i. \end{aligned}$$

Sometimes, the offset and tilt factors are measured by the scanning system and can be found in a log file corresponding to the data. Yet in other cases, this information is lacking or is inaccurate. Manual correction is tedious and should thus be avoided. In Fig. 10(b), an algorithm is schematically presented that automatically estimates the unknown alignment parameters. Based on the measured projections, a first reconstruction is computed with a conventional projection geometry (i.e. with $\phi = \delta U = \delta V = 0$). From this reconstructed volume, a single forward projection image is created, and a registration procedure is used to find the translation and rotation parameters that optimally match the measured to the simulated projection data. Because the original reconstruction was created using a misaligned projection geometry, it is likely to contain some misalignment artifacts, which might also have influenced the registration procedure.

The whole process is therefore repeated with the updated, and more accurate, projection geometry until the alignment parameters converge. This typically happens after only a few iterations.



(a) projection image (b) before correction (c) after correction

(d) convergence (e) projection difference before correction (f) projection difference after correction

Fig. 11. Automatic detector offset and tilt estimation of a dataset containing a box of candy with a highly attenuating resolution phantom.

To demonstrate this method, consider Fig. 11. A plastic box containing candy and a highly attenuating QRM resolution phantom was scanned in the TomoLab microCT at the SYRMEP beamline at Elettra, Trieste, Italy. In total 1444 projections of 890×1336 were acquired, of which only 288 are considered here. The source-to-object distance was 6000mm, the object-to-detector distance was 1000mm, and the detector size was 0.375 μm. A single projection image is shown in Fig. 11(a). The uncorrected reconstruction shown in Fig. 11(b) clearly shows severe misalignment artifacts. After 4 iterations of the automatic alignment method presented above, these artifacts have disappeared (Fig. 11(c)). Figs. 11(e) and 11(f) show the corresponding projection difference.

5. Algorithm prototyping

Suppose a user has a custom MATLAB library with an implementation of some special advanced reconstruction algorithm. One example might be some form of *total variation minimization (TVmin)*, for example using Chambolle-Pock [23], which currently does not have a native implementation inside the ASTRA framework. Custom libraries are typically built for general use, and their forward- and backprojection operations are likely performed as *sparse matrix vector multiplications (SpMV)*. In order to use these libraries, the user is thus responsible for the construction of the projection matrix. While the ASTRA Toolbox provides the functionality to turn any 2D geometrical setup into a sparse projection matrix (see Table 3), this can only be advised for very small problems. For realistic data sizes, the projection matrix is simply too large to fit into the main memory space. Moreover, as the performance in tomographic reconstructions is latency-bound rather than computation-bound [24], it is best to always recompute the projection matrix on the fly, i.e. during each forward- or backprojection. The ASTRA Toolbox has very efficient implementations for these projection operators, exploiting the power of the GPU hardware. Ideally therefore, the projection operations of the ASTRA Toolbox are used inside the advanced logic of the custom reconstruction library.

Table 3. MATLAB code demonstrating how to extract a **projection matrix** for a certain projection and volume geometry from the ASTRA toolbox, and how to use this matrix inside an external function performing a SIRT reconstruction. Due to memory constraints, this approach is feasable only for small problems.

```
function v = SIRT(W, p, v0, iters)
   R = 1 ./ W*ones(size(v0));
   C = 1 ./ W'*ones(size(p));
   v = v0;
   for k = 1:iters
      v = v + C.*(W'*(R.*(p-W*v)));
   end
end

proj_id = astra_create_projector('linear', proj_geom, vol_geom);
matrix_id = astra_mex_projector('matrix', proj_id);
W = astra_mex_matrix('get', matrix_id);

v = SIRT(W, p, v0, iters);
```

Table 4. MATLAB code demonstrating a **naive** approach to include ASTRA projector operators in a SIRT function.

```
function v = SIRT_naive(proj_geom, vol_geom, p, v0,iters)
   [~, tmp] = astra_fp(ones(size(v0)), proj_geom, vol_geom);
   R = 1 ./ tmp;
   [~, tmp] = astra_bp(ones(size(p)), proj_geom, vol_geom);
   C = 1 ./ tmp;
   v = v0;
   for k = 1:iters
      [~, fp] =  astra_fp(v, proj_geom, vol_geom);
      [~, upd] =  astra_bp(R.*(p-fp), proj_geom, vol_geom);
      v = v + C .* upd;
   end
end

v = SIRT_naive(vol_geom, proj_geom, p, v0, iters);
```

Table 5. MATLAB code demonstrating how the **opTomo** object is configured and simply passed on to the same SIRT-function as in Table 3.

```
W = opTomo('cuda', proj_geom, vol_geom);
v = SIRT(W, p, v0, iters);
```

The naive approach to this would be to open up the external library and replace all the SpMV occurrences with function calls to the appropriate elements of the ASTRA Toolbox (Table 4). While possible, this is not advisable as the user might not be familiar with the internal code of the custom library and maintaining its code can be quite difficult.

A better approach is to use a recent MATLAB library called the Spot-Toolbox [25]. This allows one to overwrite operations like matrix-multiplications with custom logic. The ASTRA Toolbox for MATLAB comes combined with such a spot-operator called *opTomo* [26]. This operator can be configured with the projection and volume geometry of the setup, after which MATLAB will regard it in many ways as a normal matrix. It can thus be passed to the external library as a normal function parameter. However, each time MATLAB detects a multiplication of this matrix and a vector or a multiplication of the transposed matrix and a vector, respectively the ASTRA forward projection or backprojection code is called (Table 5).

As an example, consider the *Simultaneous Iterative Reconstruction Technique (SIRT)*. This

algebraic reconstruction method regards the reconstruction problem as the solving of the system

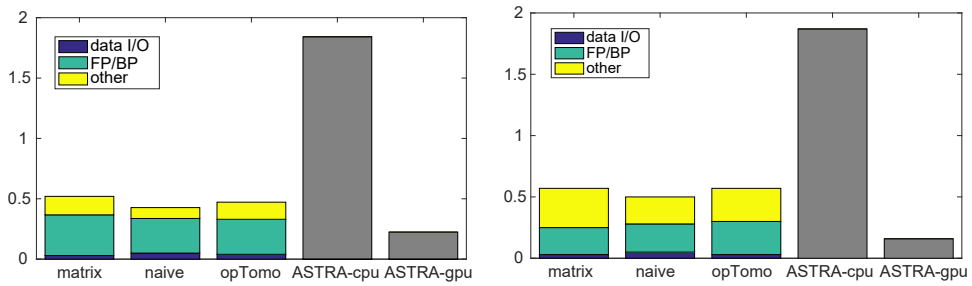
$$\boldsymbol{W}\boldsymbol{v} = \boldsymbol{p}, \tag{3}$$

in which $\boldsymbol{p}$ is a vector containing the projection data, $\boldsymbol{v}$ is a vector representing the unknown reconstructed volume, and $\boldsymbol{W}$ is the projection matrix mapping the volume geometry onto the projection geometry. As Eq. (1) cannot be solved directly, SIRT uses the following iteration scheme:

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)} + \boldsymbol{C}\boldsymbol{W}^T\boldsymbol{R}(\boldsymbol{p} - \boldsymbol{W}\boldsymbol{v}^{(k)}), \tag{4}$$

where $\boldsymbol{C}$ is a diagonal matrix containing the inverse column sums of $\boldsymbol{W}$ (i.e. $c_{jj} = 1/\sum_i w_{ij}$), and $\boldsymbol{R}$ is a diagonal matrix containing the inverse row sums of $\boldsymbol{W}$ (i.e. $r_{ii} = 1/\sum_j w_{ij}$).

In Fig. 3, a MATLAB function is shown that performs a SIRT reconstruction given a projection matrix ($\boldsymbol{W}$), projection data ($\boldsymbol{p}$), an initial estimation ($\boldsymbol{v}^{(0)}$), and a number of iterations. In practice, this script would not be useful as SIRT also has a native implementation inside the ASTRA Toolbox, which is much faster. However, this does provide the flexibility to quickly prototype new ideas. To use the SIRT function, a projection matrix is extracted from the ASTRA Toolbox and passed to the function as a parameter. This method requires a lot a memory, and does not exploit the efficient implementation of the ASTRA Toolbox. With a naive linking of the ASTRA Toolbox and the script in Table 4, the projection operations run on the GPU using the ASTRA Toolbox implementation, but the code turns from 6 elegant lines to 10 lines that are a lot less readable. Also note that the function parameter list is slightly different, which might break other pieces of code. With the usage of the Spot tools, in Table 5, the best of both worlds are combined.

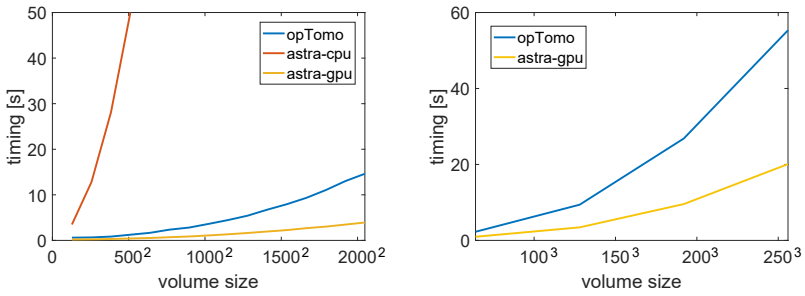


(a) SIRT (b) CGLS

Fig. 12. Timings for 100 iterations of various implementations of the (a) SIRT and (b) CGLS reconstruction algorithm on a 100×100 volume. The first three bars correspond to Table 3, 4, and 5, respectively. The final two bars refer to native ASTRA implementations of the SIRT and CGLS algorithm.

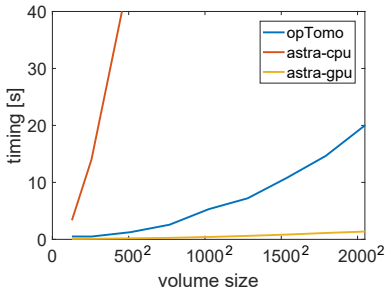
In terms of performance, Fig. 12(a) shows the timings of these different SIRT implementations for 100 iterations on a 128×128 volume with 180 projections, as performed on a system containing an Intel Core i7-6700HQ CPU running at 2.60GHz, an NVIDIA GeForce GTX 960M GPU, and 32GB of system memory. Of the three implementations presented in this section, the original script (Fig. 3) is the slowest as it does not make use of the efficiently implemented ASTRA Toolbox projection operations. Both the naive (Table 4) and the opTomo (Table 5) code are substantially faster, with the opTomo code being only very slightly the weaker of the two. Also included in Fig. 12 are two native ASTRA implementations of SIRT. The one that fully runs on the CPU, turns out to be substantially slower than any of these MATLAB implementations. This is due to MATLAB also using GPU computations for performing SpMV's. The ASTRA GPU based implementation is slightly more than a factor two faster than the fastest MATLAB

code. This is mainly due to the constant copying back and forth of the data that is going on in these MATLAB versions. In Fig. 12(b), the same experiment is repeated with a 100 iteration CGLS reconstruction algorithm.

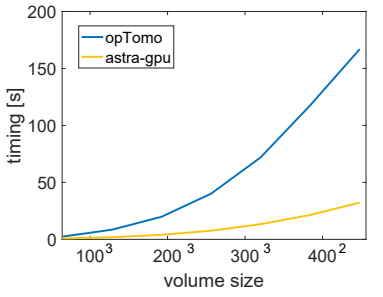


(a) System A, 2D

(b) System A, 3D



(c) System B, 2D



(d) System B, 3D

Fig. 13. Reconstruction times of 100 SIRT iterations as a function of the volume size (both 2D and 3D), on two distinct systems.

In Fig. 13, reconstruction times of 100 SIRT iterations are plotted as a function of the volume size (both 2D and 3D). In each, a total of 180 parallel-beam projection angles were considered. This experiment was performed on two different systems: System A, as described in the previous paragraph; and System B, a 12-core Intel Xeon E5-2630 computation server running at 2.30GHz and fitted with 64GB of memory and a powerful Tesla K20 GPU card.

6. Discussion and conclusions

In this article, the ASTRA Toolbox has been introduced and several use-cases were presented showing off its main features. These include its free, open source nature [14], its ability to describe virtually any 2D and 3D projection setup using vector geometries, and its flexibility regarding the easiness with which it can be included in other frameworks. Combined, they make the ASTRA Toolbox very suitable for fast prototyping of new applications, new geometry setups, and new reconstruction methods. Furthermore, due to its efficiently implemented building blocks, these prototypes can straightforwardly be upscaled to realistic data sizes and usages.

However, it should be noted that the ASTRA Toolbox is by no means the first and only tool to consider when dealing with tomographic reconstruction. Many users might prefer commercial software packages as they provide an easy-to-use graphical user interface, which the ASTRA Toolbox does not. Also, because of its flexibility regarding application fields, scanning devices and protocols, the ASTRA Toolbox can not provide out-of-the box support for various file

formats. In order to use the toolbox, the user thus requires knowledge of these file formats, and the skill to parse and process them in the MATLAB or Python layer. Moreover, the algorithms bundled in the ASTRA Toolbox are limited to reconstruction methods, and do not include typical preprocessing (e.g., flat field correction, phase retrieval) and postprocessing (e.g., segmentation, morphological operations, mesh generation) algorithms. For practical use, the toolbox must thus be linked with other tools such as TomoPy [27]. These disadvantages limit the target audience of the ASTRA Toolbox mainly to researchers and users with a expertise in computer science.

While this may seem a huge issue, these research communities are in fact relatively large, and more of them are appearing in the literature. The list of applications that can benefit from the ASTRA Toolbox is extensive, and includes (i) material sciences, through electron tomography [4], diffraction contrast tomography with synchrotron radiation [28], microCT systems, laminography systems [9], etc.; (ii) biomedical sciences, through microCT [1], tomosynthesis setups, etc.; (iii) industrial purposes, through conveyor belt systems [3], microCT systems, etc.; and (iv) security purposes [18].

In summary, the ASTRA toolbox is an excellent platform to bridge the large gap between application scientists and researchers in the field of numerical mathematics and linear solvers [29]. That way, new and advanced numerical solvers can be tested on realistic data, benefiting both communities.

Funding

iMinds (ICON MetroCT); Agency for Innovation by Science and Technology in Flanders (IWT) (SBO120033); Dutch Organization for Scientific Research (NWO) (639.072.005); European Cooperation in Science and Technology (COST) (EXTREME Action MP1207).

Acknowledgments

The authors which to express their gratitude towards Lucia Mancini and Diego Dreossi at the Elettra Synchrotron, Trieste, Italy, for providing the data used in Section (4.5).