Contents lists available at ScienceDirect

Ultramicroscopy

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

The Ettention software package

Tim Dahmen [a,c,*], Lukas Marsalek [b,c], Nico Marniok [c], Beata Turoňová [c,d], Sviatoslav Bogachev [c], Patrick Trampert [a], Stefan Nickels [a], Philipp Slusallek [a,c]

[a] German Research Center for Artificial Intelligence GmbH (DFKI), 66123 Saarbrücken, Germany
[b] Eyen SE, Na Nivách 1043/16, 141 00 Praha 4, Czech Republic
[c] Saarland University, 66123 Saarbrücken, Germany
[d] IMPRS-CS, Max-Planck Institute for Informatics, Campus E 1.4, 66123 Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 27 January 2015
Received in revised form
9 October 2015
Accepted 11 October 2015

Keywords:
Electron tomography
Tomographic reconstruction
Software architecture
GPU
OpenCL
Block iterative methods
High performance computing

ABSTRACT

We present a novel software package for the problem "reconstruction from projections" in electron microscopy. The Ettention framework consists of a set of modular building-blocks for tomographic reconstruction algorithms. The well-known block iterative reconstruction method based on Kaczmarz algorithm is implemented using these building-blocks, including adaptations specific to electron tomography. Ettention simultaneously features (1) a modular, object-oriented software design, (2) optimized access to high-performance computing (HPC) platforms such as graphic processing units (GPU) or many-core architectures like Xeon Phi, and (3) accessibility to microscopy end-users via integration in the IMOD package and eTomo user interface. We also provide developers with a clean and well-structured application programming interface (API) that allows for extending the software easily and thus makes it an ideal platform for algorithmic research while hiding most of the technical details of high-performance computing.

1. Introduction

1.1. Image reconstruction from projections

Image reconstruction from projections (tomographic reconstruction) is a fundamental technique for obtaining a three-dimensional (3D) representation of a physical quantity from its indirect observations, the two-dimensional (2D) projections. It has applications in 3D microscopy, including conventional transmission electron microscopy (CTEM) [1], scanning-transmission electron microscopy (STEM) [2], and X-Ray microscopy [3]. Applications are also numerous in other fields, for example industrial non-destructive testing [4], medical imaging [5], or geological sciences [6].

Hardware advances in projection acquisition and automated data collection routines, as well as the need for better resolution and isotropy of the reconstruction put increasing demands on quality, flexibility, and speed of execution of tomographic reconstruction algorithms. Iterative reconstruction techniques have become a popular tool to address these demands due to their ability to deal with conditions frequently encountered in electron tomography (ET), such as arbitrary projection geometries, noisy data due to low-dose imaging, and incomplete projection sets. The straightforward incorporation of a-priori information and regularization techniques is an additional favorable property.

However, iterative reconstruction methods impose high computational demands that require parallel implementations to make them practically applicable. Specialized programming of either the parallel single instruction multiple data (SIMD) units of CPUs [7] or the graphics processing units (GPUs) [8] is required to create such high-performance software. Both approaches require technical, low-level access to hardware and are therefore difficult to use and time-consuming to implement.

GPU-based acceleration of iterative techniques has first been investigated in the area of X-Ray computed tomography (CT), using fast texture mapping hardware and hardware-accelerated projections [9,10]. A CUDA-based implementation is demonstrated in [11], where forward projections are realized through ray marching with fixed sampling step and a distance-weighted back projection. This approach

* Corresponding author at: Saarland University, 66123 Saarbrücken, Germany.
E-mail address: Tim.Dahmen@dfki.de (T. Dahmen).

http://dx.doi.org/10.1016/j.ultramic.2015.10.012
0304-3991/© 2015 Elsevier B.V. All rights reserved.

forms a foundation for most GPU-based implementations of iterative techniques and makes use of the high efficiency of ray marching on GPUs, which has been demonstrated numerous times, e.g. [12,13].

1.2. Existing open source software for reconstruction in electron tomography

IMOD [14] is a comprehensive and well-established package for electron tomography processing and reconstruction, offering the algorithms weighted back projection (WBP) and simultaneous iterative reconstruction technique (SIRT). It provides mature CPU and cluster parallelization options, which enable the framework to run generic parallel processes. Highly optimized support for GPU processing using CUDA is integrated as well. IMOD is designed as a set of command-line programs, which can optionally be connected through a Java-based user interface (eTomo) into pre-defined workflows. It is possible to extend IMOD with additional reconstruction techniques. Unfortunately, the re-use of existing code is not straightforward as individual modules do not explicitly expose an application programming interface (API).

The ASTRA toolbox [15] uses MATLAB, Python or C++ to provide GPU based algorithms for 2D and 3D tomography. It supports fan and parallel beam geometry for 2D, as well as parallel and cone beam geometry for 3D. GPU implementations are available for SIRT, discrete algebraic reconstruction technique (DART) [16], Conjugate Gradient for Least Square [17], and Feldkamp–Davis–Kress [18] algorithms. The ASTRA toolbox is focused on algorithmic research. Similarly to Ettention, ASTRA provides direct access to GPU-accelerated forward projection and back projection operations. However, for cases that do not allow slices to be reconstructed independently, the software assumes that the entire reconstruction volume fits in device memory. This leads to a limitation to a relatively low reconstruction resolutions. However, apart from the original approach to use the library by writing code, it has recently been integrated in the FEI Inspect3D Software as well, indicating a shift towards direct usability by microscopy users.

The software package TomoJ [19] is a plugin for ImageJ, a widespread image processing tool implemented in Java. TomoJ provides functionality for preprocessing, alignment, and tomographic reconstruction of ET tilt series. The main advantage of this software is the integration into ImageJ with its powerful and well-structured API.

A strongly integrated approach to ET is taken by the software EM3D [20]. The software provides standard support for alignment and reconstruction of ET tilt series. Additionally, it features advanced functions for segmentation, iso-surface extraction, and rendering of tomograms. The Tomo3D package [21] takes a different approach for accelerating popular reconstruction techniques like WBP and SIRT by using the SIMD features of x86 multi-core CPUs. It achieves comparable performance and quality to GPU reconstructions for some experimental conditions. PROTOMO [22] is a software package designed specifically for ET. It focuses on a marker-free alignment and reconstruction by variable-weight weighted back projection. The software package OpenMBIR introduces a unique iterative reconstruction algorithm with a forward model, which includes a compensation for Bragg scattering [23]. UCSF Tomography [24] is a software package for tilt-series collections, which originally made use of sample movement prediction avoiding the need for additional focusing and tracking. Recently, it has been extended with real-time alignment and reconstruction of ET tilt series aiming at providing immediate feedback during data collection rather than highest-possible quality reconstruction. The real-time automatic mode of operation is achieved through employing marker-free alignment and WBP reconstruction.

Several packages with focus on related averaging techniques such as sub-tomogram averaging and single particle analysis (SPA) also provide basic tomogram reconstruction capabilities. Examples include EMAN2 [25], SPIDER [26], RELION [27], Bsoft [28], or PyTom [29].

1.3. Existing commercial packages

Apart from open-source and research-based software packages there are only a few commercial tools. IMAGIC [30] is a general purpose, interactive image analysis software package written in FORTRAN 77. It is mainly applied in the field of high-resolution biological electron microscopy. It also contains advanced techniques for the analysis of images using single particle techniques, including pattern recognition. Its main 3D reconstruction algorithm is based on the exact filter back projection algorithm.

Inspect3D [31] is an integrated software package for acquisition, alignment, reconstruction, and visualization of electron tomography data. Its key feature with respect to this work is the highly optimized GPU SIRT technique. It targets primarily EM end-users without exposing any of the GPU code blocks.

A comprehensive closed-source suite focused on commercial GPU-based electron tomography reconstruction is Digisens DigiECT [32]. It has a wide support for various iterative reconstruction algorithms like SART, SIRT, or OS-SART and offers multi GPU support and high-speed reconstructions.

2. Summary

As can be seen, a number of software packages for electron tomography already exists. They can be classified into two groups depending on the targeted users. Software focusing on applicability by end-users requires support for high-resolution images and volumes, high reconstruction performance, and good usability, preferentially with a graphical user interface and integration with existing workflows. IMOD is the most important representative of this group. Other software packages aim mainly at providing researchers with a platform to implement algorithmic innovations. Those software packages require a modular design, good debugging functionality, means to easily incorporate a-priori information and regularization techniques, and a well-structured API, which allows the convenient re-use of existing code. The ASTRA toolbox is the most noteworthy representative of this group. However, many projects require both algorithmic innovation as well as immediate applicability to high-resolution experimental data. Additionally, it is necessary to provide suitable hardware-independent and domain-specific abstractions over the various underlying many-core platforms, such that the technical details are hidden, enabling non-expert developers to use the software easily.

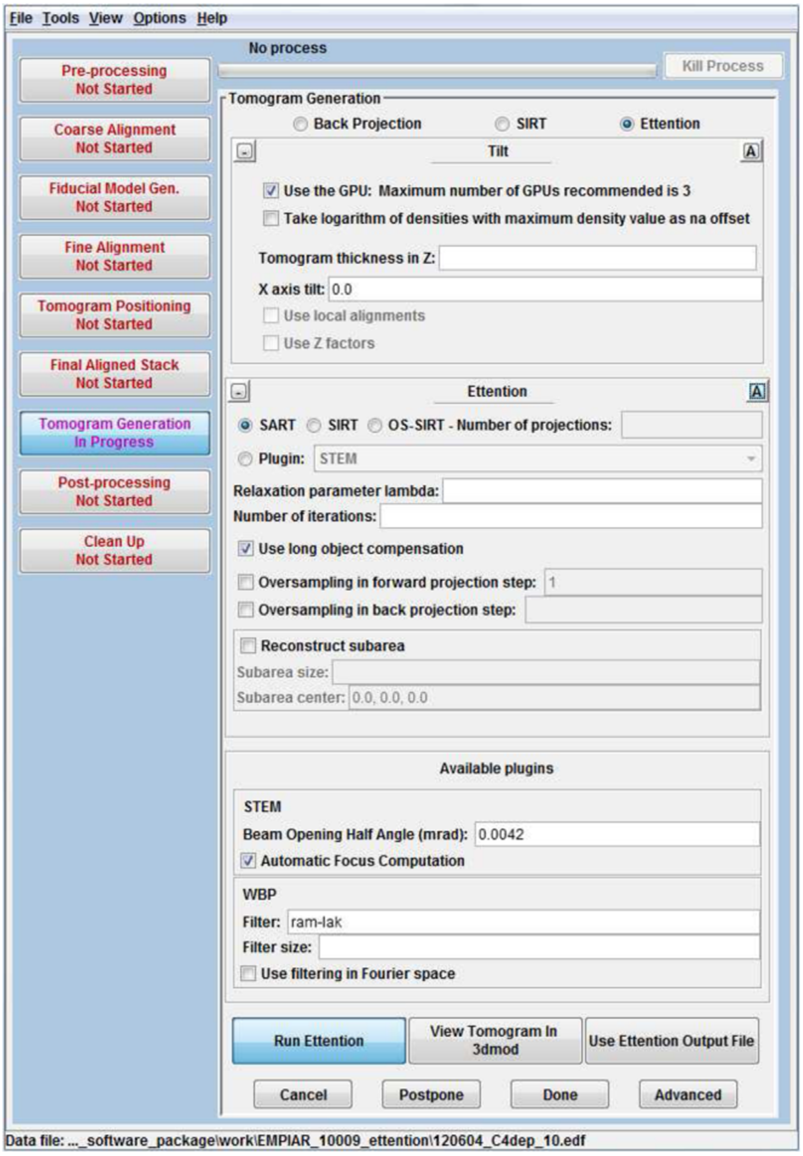
2.1. Key contributions

In this work, we present the software package Ettention, which addresses the issues mentioned above by creating a set of high-performance computing (HPC) building blocks for quickly assembling situation-specific advanced tomographic reconstruction algorithms. These building blocks are self-contained as far as memory management and parallelism are concerned, which makes them easily applicable to high-resolution experimental data. We describe and implement several different algorithms using the proposed set of building blocks and show that it is possible to assemble various reconstruction algorithms with little effort while maintaining a good computational performance. We also show how the assembled algorithms can be readily integrated in existing workflows, specifically inside the IMOD package and eTomo user interface.

3. Using Ettention

The simplest way to start using Ettention is via the eTomo Graphical User Interface (GUI) in the IMOD package [14]. ETomo is a user interface written in Java, which provides easy access to the individual algorithmic steps in IMOD, implemented as separate executables. The interface covers the entire workflow for electron tomography, including projection preprocessing, CTF correction, and alignment as well as tomogram region-of-interest selection, reconstruction, postprocessing, and filtering. Ettention plugs in to seamlessly replace the tomographic reconstruction step in this pipeline.

a


File Tools View Options Help
No process
Kill Process
Pre-processing Not Started
Coarse Alignment Not Started
Fiducial Model Gen. Not Started
Fine Alignment Not Started
Tomogram Positioning Not Started
Final Aligned Stack Not Started
Tomogram Generation In Progress
Post-processing Not Started
Clean Up Not Started
Run Ettention
View Tomogram In 3dmod
Use Ettention Output File
Cancel
Postpone
Done
Advanced

Tomogram Generation

Back Projection SIRT [✓ Ettention]

Tilt

[✓ Use the GPU: Maximum number of GPUs recommended is 3]

[ ] Take logarithm of densities with maximum density value as na offset

Tomogram thickness in Z:

X axis tilt: 0.0

[ ] Use local alignments

[ ] Use Z factors

Ettention

[✓ SART] SIRT OS-SIRT - Number of projections:

Plugin: STEM

Relaxation parameter lambda:

Number of iterations:

[✓ Use long object compensation]

[ ] Oversampling in forward projection step: 1

[ ] Oversampling in back projection step:

[ ] Reconstruct subarea

Subarea size:

Subarea center: 0.0, 0.0, 0.0

Available plugins

STEM

Beam Opening Half Angle (mrad): 0.0042

[✓ Automatic Focus Computation]

WBP

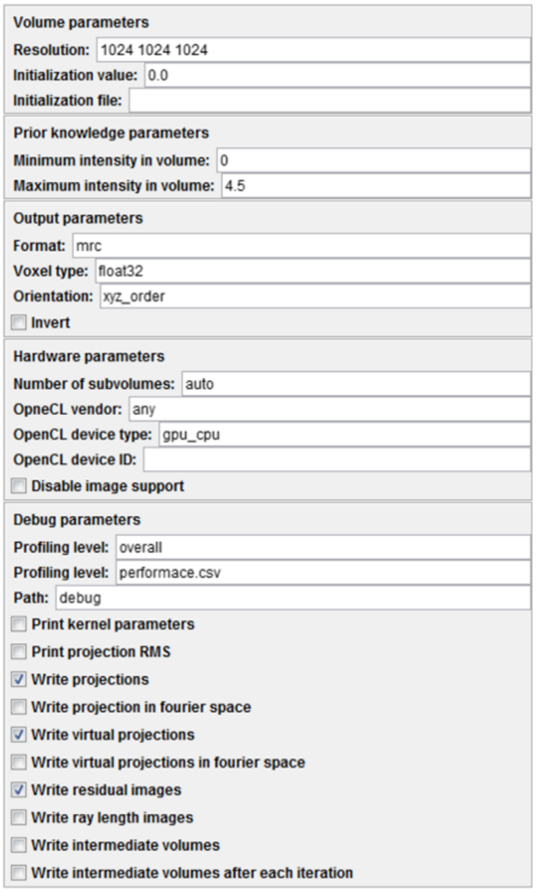
Filter: ram-lak

Filter size:

[ ] Use filtering in Fourier space

Data file: ..._software_package\work\EMPIAR_10009_ettention\120604_C4dep_10.edf

b



Volume parameters

Resolution: 1024 1024 1024

Initialization value: 0.0

Initialization file:

Prior knowledge parameters

Minimum intensity in volume: 0

Maximum intensity in volume: 4.5

Output parameters

Format: mrc

Voxel type: float32

Orientation: xyz_order

[ ] Invert

Hardware parameters

Number of subvolumes: auto

OpneCL vendor: any

OpenCL device type: gpu_cpu

OpenCL device ID:

[ ] Disable image support

Debug parameters

Profiling level: overall

Profiling level: performace.csv

Path: debug

[ ] Print kernel parameters

[ ] Print projection RMS

[✓ Write projections]

[ ] Write projection in fourier space

[✓ Write virtual projections]

[ ] Write virtual projections in fourier space

[✓ Write residual images]

[ ] Write ray length images

[ ] Write intermediate volumes

[ ] Write intermediate volumes after each iteration

Fig. 1. The integration of Ettention into eTomo GUI allows accessing the Ettention functionality from this widespread tool. (a) The basic view allows convenient access to the most common parameters as well as new, user-assembled algorithms via the plugin mechanism. (b) The advanced view allows to access additional parameters, such as functions for debugging and profiling or detailed output format configurations.

The components necessary to incorporate Ettention into IMOD package and eTomo GUI ship with the Ettention distribution and consist of two parts: a modified version of eTomo and a generic IMOD adapter for Ettention. Adding Ettention into an existing IMOD/eTomo installation requires two steps: installing Ettention and copying the modified eTomo.jar file into the IMOD folder.

Once installed, additional GUI elements are displayed in the eTomo User Interface (Fig. 1), which allow to access frequently used functions. That includes reconstruction algorithm (Ettention SART, Ettention SIRT or Ordered Subset SIRT), relaxation parameter, number of iterations, and oversampling options for forward and back projection.

In order to access only occasionally used functions, the GUI can be set to advanced mode. This exposes a large number of parameters such as hardware device selection (GPU), functions for debugging and profiling, options to load initial volumes, and more detailed choices for the output format.

As will be explained later, new user-assembled algorithms can be added to Ettention via a plugin mechanism. These new algorithms may also be made available in IMOD/eTomo with little effort. Every plugin consists of a shared library (.dll or.so) and an user interface extension file (.xml). The dll/so file is copied into the Ettention plugin folder, the xml file into the eTomo folder. The plugin can then be activated using a radio button, as show in (Fig. 1). By default, the following plugins ship with the Ettention distribution: STEM tomography using Combined Tilt- and Focal Series [33,34], Weighted Back Projection, Weighted SIRT, and tomography with Geometric Prior Knowledge.

As an alternative to using IMOD/eTomo, Ettention can also be operated from the command line on Windows or Linux. The command "reconstructConsoleApp –help" lists the available parameters. Most importantly, a configuration file can be specified using the parameter "–configfile filename.xml", which allows storing setups in xml files for repeated execution.

4. Extending Ettention

Ettention can be extended using a plugin concept. There are several types of plugins, which address different aspects of the software. The following plugin types exist:

- **Reconstruction Algorithm** allows to define a new algorithm, which takes a set of projections as input and generates a volume as output. The core of the new algorithms being created by users is the Reconstruction Algorithm plugin. Two instances of the creation of such plugins are discussed in detail later. By default, Ettention supports a Block-Iterative Reconstruction with configurable block size, SART and SIRT.
- **Forward Projection** takes a volume and a projection direction and generate a projection image. By default, Ettention supports parallel and perspective projection and a parallel projection with limited depth of field.
- **Back Projection** takes a volume, a residual image and a projection direction as input and correct the volume. By default, Ettention supports parallel and perspective back projection and two back projections for STEM tomography with limited depth of field.
- **Projection Set Iterator** is a method to iterate over input projections, which allows to implement for example sequential or random access schemes. By default, Ettention supports sequential access, random access and the Saxton scheme.
- **Image File Format** provides support for adding new projection file formats. By default, Ettention supports MRC, multislice TIFF, and image stacks.
- **Volume File Format** allows to add a new tomogram output file format. By default, Ettention supports MRC.

Some applications require to provide more than one of these types at the same time, for example a new reconstruction algorithm that depends on a custom forward and back projection.

5. The building blocks

In order to address the quick algorithm prototyping as well as high-performance and applicability to large real-world data, Ettention introduces the concept of "building blocks" for tomographic reconstruction algorithms. These building blocks hide the technical details of HPC programming, encapsulate memory management and parallelism, and provide domain-specific C++ objects. The building blocks include various forward projections, back projections, image manipulations, and input/output operations. On the following pages, we will describe the most important building blocks.

5.1. Forward projections

Forward projection operations are provided for parallel projection (e.g. TEM tomography) and for perspective projection. A version assuming parallel projection with limited depth of field is provided for use with aberration corrected STEM. The forward projection iterates over the pixels of a projection and for each pixel it uses one of the grid traversal routines to accumulate its value.

Similar to the super-sampling approach discussed in [35], the forward projections also support optional subpixel sampling of the projections. In this mode a 2D grid is placed over the area covered by each pixel, and an individual ray is traced through the center of each grid cell. Results are then weighted and averaged to compute the value of an individual pixel in the virtual projection.

5.2. Back projections

Building blocks for real-space back projection operations correct a volume by projecting a voxel back onto an image plane and adjusting the voxel intensity based on the pixel values covered by the voxel's projection using a customizable weighting function. The operation is implemented in a voxel-by-voxel fashion such that no synchronization issues arise. Because the projection geometry is provided as a general 4×4 matrix, the operation can be used with a large variety of microscope geometries. Preconfigured building blocks are provided for parallel and perspective back projection and for several regularization schemes, for example for use in STEM tomography [33].

All back projections support optional subvoxel sampling of a volume. Samples are placed by first computing a 2D rectangle covering the voxel footprint in a given projection. Then, a grid is placed over the rectangle; the center point of each grid cell constitutes one sample. A line is specified by the position of the sample and the projection direction. The length of the intersection between the line and the voxel is computed for each sample and assigned as a weight. The samples are then averaged according to their weights. This method achieves a better sampling than a regular 3D grid in the voxel could, because overlapping samples are avoided.

5.3. *Image processing operations for residuals and 2D projections*

Several building blocks that operate on 2D images are provided. Per image operations include the computation of residuals (implemented as per pixel difference computation), long object compensation [8], real and complex multiplication, and several statistical functions such as minimum, maximum and mean pixel intensity, and root mean square.

Another very useful basic operation is the transfer of images from real space to frequency space and back by means of a fast Fourier transform (FFT). One application of FFT is the implementation of convolution and deconvolution operations. A deconvolution is required as a component of many reconstruction algorithms, like in WBP.

5.4. *Building algorithms with Ettention*

In the following, we demonstrate how the building blocks described above are used to assemble new algorithms. A first example is the well-known Weighted Back Projection (WBP) algorithm. WBP is a direct inversion method, which corresponds to a “single iteration algorithm” in Ettention.

The Ettention framework is built around the idea of assembling building blocks that can be combined as “pipelines”. As a consequence, most algorithms consist of two steps: a setup step, where the required building blocks are instantiated, and an execution step, where the execution of the individual blocks is triggered. WBP requires two main components: a filter step, which is implemented as a convolution with the Ram-Lak filter, and a real-space back projection. More advanced filter operations such as combinations with Hamming windows can be easily implemented by exchanging the filter kernel. The setup of the pipeline is implemented in the constructor of the WBP class.

```
WBP::WBP(Framework* framework, WBP_ParameterSet* parameterSet)
{
    // open the input stack specified by a configuration file and extract projection geometry
    inputStack = framework->openInputStack(parameterSet);
    geometricSetup = new GeometricSetup(inputStack);

    // allocate an output volume, resolution is specified by configuration
    volumeOnCpu = new Volume(VoxelType::FLOAT_32, parameterSet->Get<VolumeParameterSet>());

    // allocate objects on the GPU
    volume = new GPUMappedVolume(framework->getCLStack(), volumeOnCpu);
    sourceImage = new GPUMapped<Image>(framework->getCLStack(), inputStack->getResolution());

    // allocate compute kernels and generate filter mask
    filterSize = Vec2ui(31, 31);
    ramLakGenerator = new GenerateRamLakKernel(framework->getCLStack(), filterSize);
    ramLakGenerator->launchKernel();
    convolution = new ConvolutionOperator(framework->getCLStack(),
                                          ramLakGenerator->getOutput(), sourceImage);
    backprojection = new BackProjectionOperator(framework, parameterSet, geometricSetup,
                                                volume, convolution->getOutput());
}
```

The execution is implemented in the run method. It consist of an iteration over all input projections. Each input projection is first filtered by means of a convolution and subsequently, the back projection is executed.

```
void WBP::Run()
{
    // iterate over input stack, copy to GPU, filter each image, and perform back projection
    for (int i = 0; i < inputStack->GetNumberOfProjections(); i++)
    {
        sourceImage->setObjectOnCPU(inputStack->loadProjection(i));
        convolution->launch();
        backprojection->launch();
    }
}
```

Based on the building blocks introduced with the WBP algorithm, we can now consider more interesting examples. The weighted SIRT [36] algorithm combines the ideas of using a convolution with a ramp filter in an iterative approach for tomographic reconstruction. The algorithm can be implemented in Ettention by extending the SIRTOperator class. In addition to the SIRT functionality, a convolution with a Ram-Lak filter is required. More complex filters can be created the same way.

```
WeightedSIRT::WeightedSIRT(Framework* framework,
                           AlgebraicParameterSet* parameterSet)
  : SIRTOperator(framework, parameterSet)
{
    ramLakGenerator = new GenerateRamLakKernel( framework->getClStack(), filterSize );
    ramLakGenerator->launchKernel();
    convolution = new ConvolutionOperator(framework->getClStack(),
                                          ramLakGenerator->getOutput(), NULL, false);
}
```

The convolution operator can now be used to replace the computeResidual function of the SIRT algorithm. In this implementation, the residual is first generated by computing the per pixel difference between real projection and virtual projection, as done for the SIRT algorithm. The difference is then filtered by computing the convolution with a Ram-Lak filter kernel.

```
void WeightedSIRT::ComputeResidual(void)
{
    // compute per pixel difference of real- and virtual projection
    compareKernel->SetInput(state->GetCurrentRealProjection(), state->GetVirtualProjection());
    compareKernel->launchKernel();

    // filter the residual
    convolution->setInput(compareKernel->getOutput());
    convolution->setOutput(state->GetResidual());
    convolution->launch();
}
```

After the modification of the residual value, the SIRT operator can proceed as normal.

The examples given above are achieved by recombining existing building blocks on C++ level. Some more advanced algorithms require implementing new building blocks, such as customized forward and back projections using OpenCL. For example, an algorithm that makes use of geometric prior knowledge in the form of the outer contour of an irregular reconstruction area can be implemented by adapting the forward and back projections to consider a “mask volume” provided as input. A detailed description of this example is beyond the scope of this publication and can be found along with more tutorials in the Ettention developer guide.

6. Evaluation of reconstruction quality

The reconstruction in Ettention was evaluated on a dataset of human anti–HIV-1 gp120 antibody IgG1-b12 [37]. The dataset (EMPIAR-10009) consists of a single axis tilt series of 51 images, each of which has a resolution of $2{,}048^2$ pixels. The tilt series was recorded over the tilt range of $\pm 60°$ in 1°–3° increments. We have chosen this particular dataset for comparability of different implementations, as it is publicly available on the Electron Microscopy Pilot Image Archive (EMPIAR) [38]. Tomograms were reconstructed using the IMOD

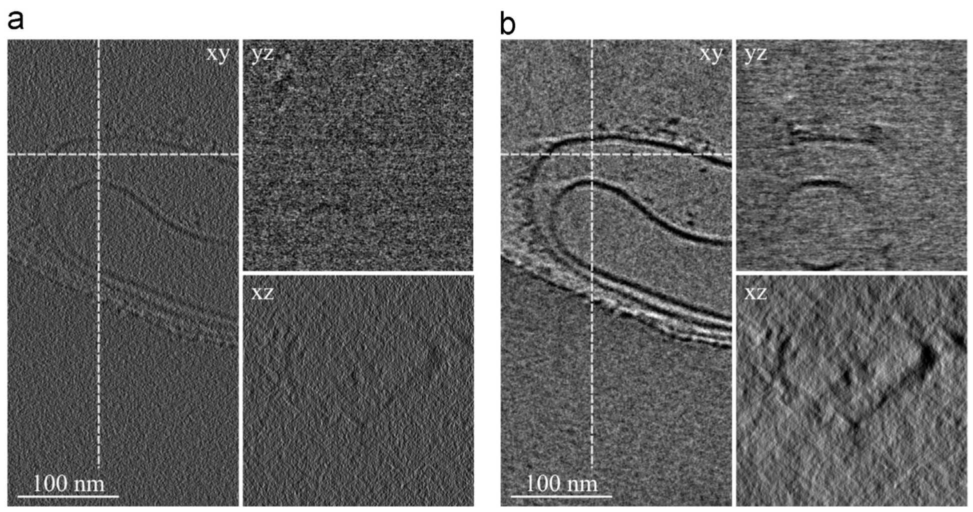


Fig. 2. Tomographic reconstructions performed with IMOD (a) WBP reconstruction and (b) SIRT with 10 iterations. The dashed lines in the xy plane indicate the relative positions of the yz and xz plane. Contrast was scaled for optimal view.

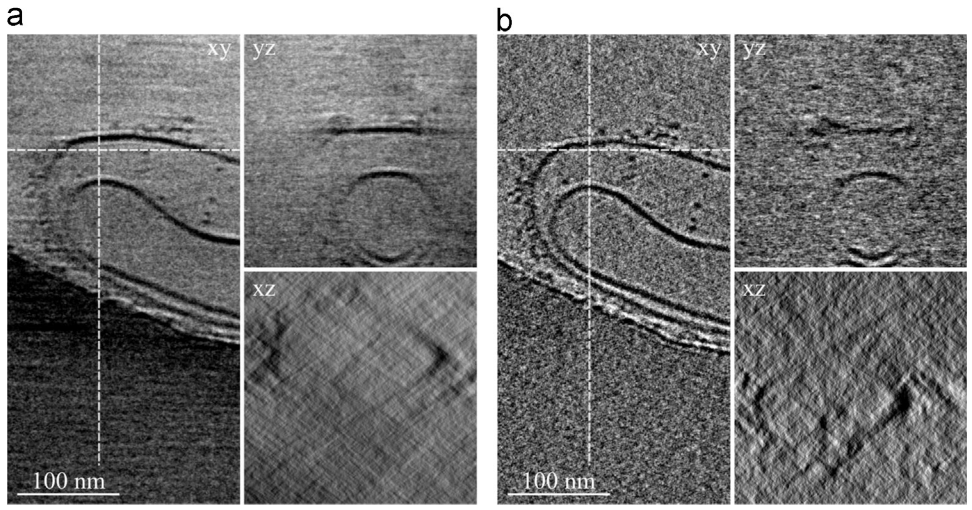


Fig. 3. Tomographic reconstructions performed with Ettention. (a) Reconstruction using 10 iterations of SIRT (b) Reconstruction using 3 iterations of SART. The dashed lines in the xy plane indicate the relative positions if the yz and xz plane. Contrast was scaled for optimal display.

Table 1
Reconstruction times with Ettention on different hardware platforms. Times are given per iteration in seconds.

	Ettention SIRT		**Ettention SART**	
	1K	**2K**	**1K**	**2K**
Tesla C2075	21 s	159 s	26 s	630 s
2 Xeon X5560	415 s	–	376 s	–
Xeon Phi 31S1P	171 s	–	246 s	–

implementations of WBP and SIRT as well as the Ettention implementations of SART and SIRT.

As a baseline, the EMPIAR-10009 dataset was reconstructed using the IMOD package. Both weighted back projection and 10 iterations of SIRT reconstructions were performed. As expected for a dataset with low signal-to-noise ratio (SNR) and limited tilt range, the SIRT reconstruction shows better contrast and reveals more details.

The same dataset was reconstructed using the Ettention software package. The reconstruction was performed using 10 iterations of SIRT and 3 iterations of SART. As can be seen by comparing Figs. 2 and 3, the SART reconstruction reveals more details than the WBP reconstruction performed using IMOD, but looks noisier than the SIRT reconstructions performed with IMOD and Ettention. Compared to the IMOD SIRT, the SIRT reconstructions with Ettention seem to yield slightly smoother results, which reveal more details, particularly in the *yz* plane. However, the Ettention reconstruction contains a low-frequency, dark region, which is not present in the IMOD reconstruction. So, Ettention generates a notably different reconstruction result than IMOD, even if both software systems use the SIRT algorithm. Reasons for this might include (1) the higher precision for the volume representation in Ettention, and (2) the fact that IMOD uses an additional weighting filter [34], which is not part of the original SIRT implementation.

7. Reconstruction performance

7.1. Test hardware

The performance measurements were run on a number of different hardware platforms: (1) A Nvidia Tesla C2075 with 6 GB of memory represents a typical GPU workstation setup, (2) a two CPU system with Xeon X5560 and 48 GB of memory represents a high-end CPU setup, and (3) an Intel Xeon Phi 31S1P with 8 GB of memory was included to represent a non-GPU HPC platform.

7.2. Total reconstruction times

All reconstructions were tested on the EMPIAR-10009 dataset. The reconstructions were executed twice, once from a projection resolution of $2{,}048^2$ pixels to an output resolution of $2{,}048^2 \times 1{,}024$ voxels (2K case) and once from a reduced projection resolution of $1{,}024^2$ pixels to a tomogram resolution of $1{,}024^2 \times 512$ (1K case). All execution times are given per iteration in seconds (Table 1).

As expected, the GPU platform shows feasible performance, particularly in the in-core (1K) case while the CPU platform is outperformed due to inferior floating point compute power and memory bandwidth. The Xeon Phi performs better than the Xeons, but does not reach the performance of the Tesla card. In the out-of-core (2K) case, the SIRT implementation outperforms SART because the volume has to be transferred via the PCI bus less often as a result of the larger size of the algorithm update block. However, when comparing the execution times of SART and SIRT, it has to be kept in mind that the SART algorithm typically shows a higher rate of convergence. Longer runtime per iteration is usually compensated by using fewer iterations. Execution times for the 2 K cases on the Xeon and Xeon Phi architectures are unfeasibly high.

8. Conclusions

With Ettention, we provide a novel framework consisting of building blocks, which can be quickly assembled into application-specific tomographic reconstruction algorithms. Compared to existing software packages for tomographic reconstruction in electron microscopy, Ettention fills the gap between IMOD and the ASTRA toolbox. For algorithmic research without the need for immediate application to high-resolution data, we recommend using the ASTRA toolbox because of the powerful MATLAB language binding. The limitation to the in-core case, i.e. to small reconstruction volumes, is less important in this case. For microscopy experimentalists, who require well-established reconstruction algorithms, we recommend using IMOD because of its highly-optimized reconstruction performance on high-resolution data. For projects that require both, algorithmic innovation and immediate application to high-resolution experimental data, we believe that the Ettention software package should be considered because it delivers a reasonable (though not optimal) performance even on high-resolution data and exposes a rich and well-structured API that allows for fast and efficient implementation of algorithmic innovations.

The Ettention frameworks hides the technically challenging aspects of GPU programming by providing an abstraction layer above memory management and parallelism and thus allows for formulating reconstruction algorithms in a problem domain language. It is implemented as a C++ library, which internally uses OpenCL to access a variety of HPC platforms. The framework is intended for integration in existing software such as IMOD and for extension by means of a plugin concept. Hereby, the plugin developer is given a rich API that exposes essentially every aspect of the framework. The feasibility of the approach was demonstrated by applying the framework to a number of research projects and by implementing a variety of tomographic reconstruction algorithms using the framework.

In order to facilitate this platform approach, Ettention is released under an LGPL license and can be downloaded from www.ettention.org for free. Plugins can have arbitrary licensing models, i.e. it is allowed for third parties to write closed source Ettention plugins under commercial licenses.

Acknowledgments

Lukas Marsalek, Nico Marniok and Tim Dahmen contributed equally to this work. Research has been partially funded within the European Research Project NOTOX (FP7-267038) and the DFG Grant IMCL (AOBJ: 600875). The authors thank the DFKI GmbH and Eyen SE for additional funding and for providing the necessary infrastructure. We thank David Mastronarde and Sue Held for their support in integrating Ettention with IMOD.

References

[1] Z. Li, G.J. Jensen, Electron cryotomography: a new view into microbial ultrastructure, Curr. Opin. Microbiol. 12 (3) (2009) 333–340.
[2] C. Kübel, A. Voigt, R. Schoenmakers, M. Otten, D. Su, T.-C. Lee, A. Carlsson, J. Bradley, Recent advances in electron tomography: TEM and HAADF-STEM tomography for materials science and semiconductor applications, Microsc. Microanal. 11 (5) (2005) 378–400.
[3] W. Meyer-Ilse, D. Hamamoto, A. Nair, S.A. Lelièvre, G. Denbeaux, L. Johnson, A.L. Pearson, D. Yager, M.A. Legros, C.A. Larabell, High resolution protein localization using soft X-ray microscopy, J. Microsc. 201 (Pt 3) (2001) 395–403.
[4] R.K. Sun, W. Brown, S.B. Leach, An Overview of Industrial X-ray Computed Tomography, 2012.
[5] D. Van de Sompel, S.M. Brady, J. Boone, Task-based performance analysis of FBP, SART and ML for digital breast tomosynthesis using signal CNR and Channelised Hotelling Observers, Med. Image Anal. 15 (1) (2011) 53–70.
[6] B. Romanowicz, Seismic Tomography of the Earth's Mantle, Annu. Rev. Earth Planet. Sci. 19 (1) (1991) 77–99.
[7] J.I. Agulleiro, E.M. Garzon, I. Garcia, J.J. Fernandez, Vectorization with SIMD extensions speeds up reconstruction in electron tomography, J. Struct. Biol. 170 (3) (2010) 570–575.
[8] W. Xu, F. Xu, M. Jones, B. Keszthelyi, J. Sedat, D. Agard, K. Mueller, High-performance iterative electron tomography reconstruction with long-object compensation using graphics processing units (GPUs), J. Struct. Biol. 171 (2) (2010) 142–153.
[9] F. Xu, K. Mueller, Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware, Nucl. Sci. IEEE Trans. 52 (3) (2005) 654–663.
[10] W.J. Palenstijn, K.J. Batenburg, J. Sijbers, Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs), J. Struct. Biol. 176 (2) (2011) 250–253.
[11] B. Keck, H. Hofmann, H. Scherl, M. Kowarschik, J. Hornegger, GPU-accelerated SART reconstruction using the CUDA programming environment, Proc. SPIE Med. Imaging Phys. Med. Imaging, 7258, (2009) 72582B (72582B–72582B–9).
[12] T. Klein, M. Strengert, S. Stegmaier, T. Ertl, Exploiting frame-to-frame coherence for accelerating high-quality volume raycasting on graphics hardware, in: Proceedings of in VIS 05 on IEEE Visualization, 2005, pp. 223–230.
[13] L. Marsalek, A. Hauber, P. Slusallek, High-speed volume ray casting with CUDA, in: IEEE Symposium on Interactive Ray Tracing, 2008.
[14] J.R. Kremer, D.N. Mastronarde, J.R. McIntosh, Computer visualization of three-dimensional image data using IMOD, J. Struct. Biol. 116 (1) (1996) 71–76.
[15] W. Van Aarle, W.J. Palenstijn, et al., The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography, Ultramicroscopy 157 (2015) 35–47.
[16] K. Batenburg, J. Sijbers, DART: a fast heuristic algebraic reconstruction algorithm for discrete tomography, Image Processing 2007, ICIP 2007, vol. 4, no. 2, 2007, p. IV 133.
[17] C. Popa, A hybrid Kaczmarz–Conjugate Gradient algorithm for image reconstruction, Math. Comput. Simul. 80 (12) (2010) 2272–2285.
[18] K. Wiesent, K. Barth, N. Navab, P. Durlak, T. Brunner, O. Schuetz, W. Seissler, Enhanced 3-D-reconstruction algorithm for C-arm systems suitable for interventional procedures, IEEE Trans. Med. Imaging 19 (5) (2000) 391–403.
[19] 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.
[20] D.B. Ress, M.L. Harlow, R.M. Marshall, U.J. McMahan, Methods for generating high-resolution structural models from electron microscope tomography data, Structure 12 (10) (2004) 1763–1774 , Oct..
[21] J.I. Agulleiro, J.J. Fernandez, Fast tomographic reconstruction on multicore computers, Bioinformatics 27 (4) (2011) 582–583.
[22] H. Winkler, 3D reconstruction and processing of volumetric data in cryo-electron tomography, J. Struct. Biol. 157 (1) (2007) 126–137.
[23] S.V. Venkatakrishnan, L.F. Drummy, M. De Graef, J.P. Simmons, C.A. Bouman. Model Based Iterative Reconstruction Bright Field electron Tomography, vol. 8657, 2013, p. 86570A.
[24] S.Q. Zheng, B. Keszthelyi, E. Branlund, J.M. Lyle, M.B. Braunfeld, J.W. Sedat, D. a Agard, UCSF tomography: an integrated software suite for real-time electron microscopic tomographic data collection, alignment, and reconstruction, J. Struct. Biol. 157 (1) (2007) 138–147.
[25] G. Tang, L. Peng, P.R. Baldwin, D.S. Mann, W. Jiang, I. Rees, S.J. Ludtke, EMAN2: an extensible image processing suite for electron microscopy, J. Struct. Biol. 157 (1) (2007) 38–46.
[26] T.R. Shaikh, H. Gao, W.T. Baxter, F.J. Asturias, N. Boisset, A. Leith, J. Frank, SPIDER image processing for single-particle reconstruction of biological macromolecules from electron micrographs, Nat. Protoc. 3 (12) (2008) 1941–1974.
[27] S.H.W. Scheres, RELION: implementation of a Bayesian approach to cryo-EM structure determination, J. Struct. Biol. 180 (3) (2012) 519–530.

[28] J.T. Huiskonen, J. Hepojoki, P. Laurinmäki, A. Vaheri, H. Lankinen, S.J. Butcher, K. Grünewald, Electron cryotomography of Tula hantavirus suggests a unique assembly paradigm for enveloped viruses, J. Virol. 84 (10) (2010) 4889–4897.
[29] T. Hrabe, Y. Chen, S. Pfeffer, L.K. Cuellar, A.-V. Mangold, F. Förster, PyTom: a python-based toolbox for localization of macromolecules in cryo-electron tomograms and subtomogram analysis, J. Struct. Biol. 178 (2) (2012) 177–188.
[30] M. van Heel, W. Keegstra, IMAGIC: A fast, flexible and friendly image analysis software system, Ultramicroscopy 7 (2) (1981) 113–129.
[31] R.H.M. Schoenmakers, R.A. Perquin, T.F. Fliervoet, W. Voorhout, H. Schirmacher, New Software for High Resolution , High Throughput Electron Tomography, (2005) 5–6.
[32] "Digisens." [Online]. Available: ⟨http://www.digisens3d.com⟩ (Accessed: 18.06.15).
[33] T. Dahmen, J.-P. Baudoin, A.R. Lupini, C. Kübel, P. Slusallek, N. de Jonge, Combined scanning transmission electron microscopy tilt- and focal series, Microsc. Microanal. 20 (2) (2014) 548–560.
[34] T. Dahmen, H. Kohr, N. de Jonge, P. Slusallek, Matched Backprojection Operator for Combined Scanning Transmission Electron Microscopy Tilt- and Focal Series, Microsc. Microanal. (2015) 1–14.
[35] M. Kunz, A.S. Frangakis, Super-sampling SART with ordered subsets, J. Struct. Biol. 188 (2) (2014) 107–115.
[36] D. Wolf, A. Lubk, H. Lichte, Weighted simultaneous iterative reconstruction technique for single-axis tomography, Ultramicroscopy 136 (2014) 15–25.
[37] C.A. Diebolder, F.J. Beurskens, R.N. de Jong, R.I. Koning, K. Strumane, M.A. Lindorfer, M. Voorhorst, D. Ugurlar, S. Rosati, A.J.R. Heck, J.G.J. van de Winkel, I.A. Wilson, A. J. Koster, R.P. Taylor, E.O. Saphire, D.R. Burton, J. Schuurman, P. Gros, P.W.H.I. Parren, Complement is activated by IgG hexamers assembled at the cell surface, Science 343 (6176) (2014) 1260–1263.
[38] A. Patwardhan, J.-M. Carazo, B. Carragher, R. Henderson, J.B. Heymann, E. Hill, G.J. Jensen, I. Lagerstedt, C.L. Lawson, S.J. Ludtke, D. Mastronarde, W.J. Moore, A. Roseman, P. Rosenthal, C.-O.S. Sorzano, E. Sanz-García, S.H.W. Scheres, S. Subramaniam, J. Westbrook, M. Winn, J.R. Swedlow, G.J. Kleywegt, Data management challenges in three-dimensional EM, Nat. Struct. Mol. Biol. 19 (12) (2012) 1203–1207.