Distributing reconstruction algorithms using the ASTRA Toolbox

Willem Jan Palenstijn

CWI, Amsterdam

26 jan 2016 – Workshop on Large-scale Tomography, Szeged

Part 1: Overview of ASTRA

ASTRA Toolbox in a nutshell

Fast and flexible building blocks for 2D/3D tomographic reconstruction, aimed at algorithm developers and researchers.

Matlab/Python toolbox for easy implementation of algorithms.

NVIDIA GPU support for high performance

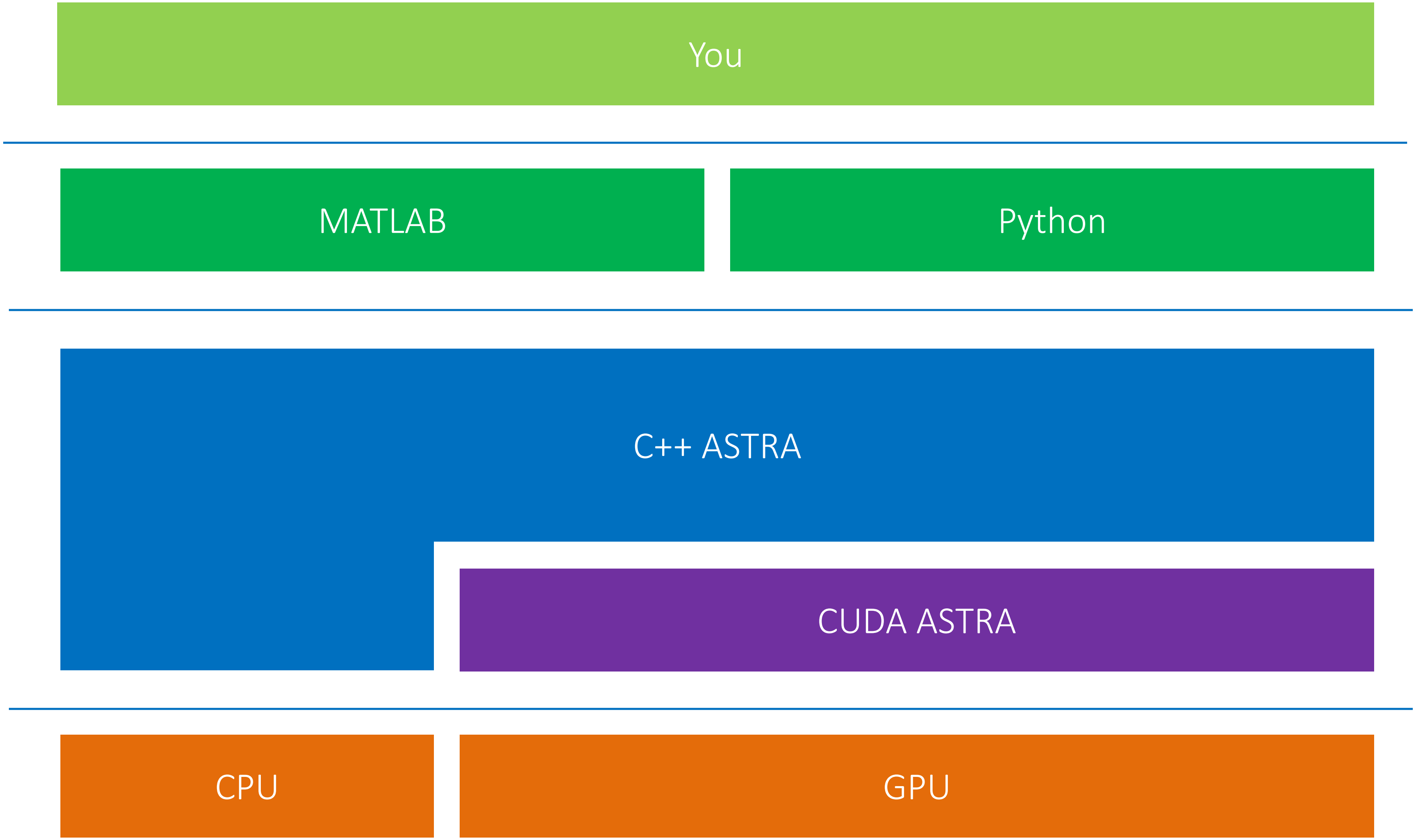
Runs on Windows and Linux.

Free and open source.

History

- 2007: Toolbox started at U. Antwerp (Vision Lab)
  - Initial goal: reduced implementation work for internal PhD projects
- 2012: First open source release
- 2014: Now developed jointly by CWI and Vision Lab
- Dec 2015: Release 1.7

Toolbox organization



Features

Geometries:

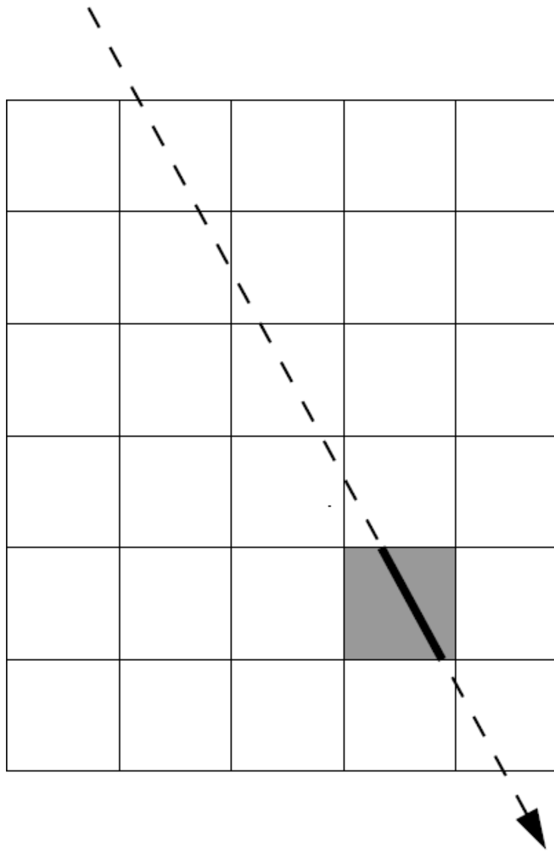
- 2D parallel and fan beam
- 3D parallel and cone beam
- All with fully flexible source/detector positioning

Reconstruction algorithms:

- FBP, FDK reconstruction
- Iterative SIRT, CGLS reconstruction

Primitives for building your own algorithms

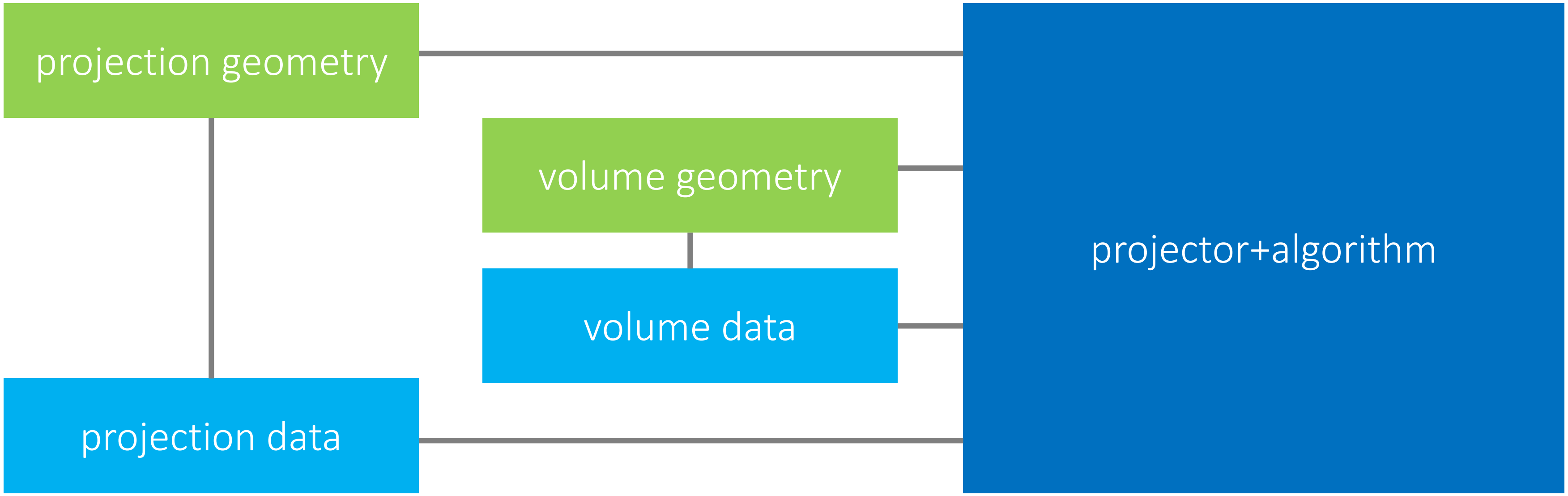
Tomography as linear equations



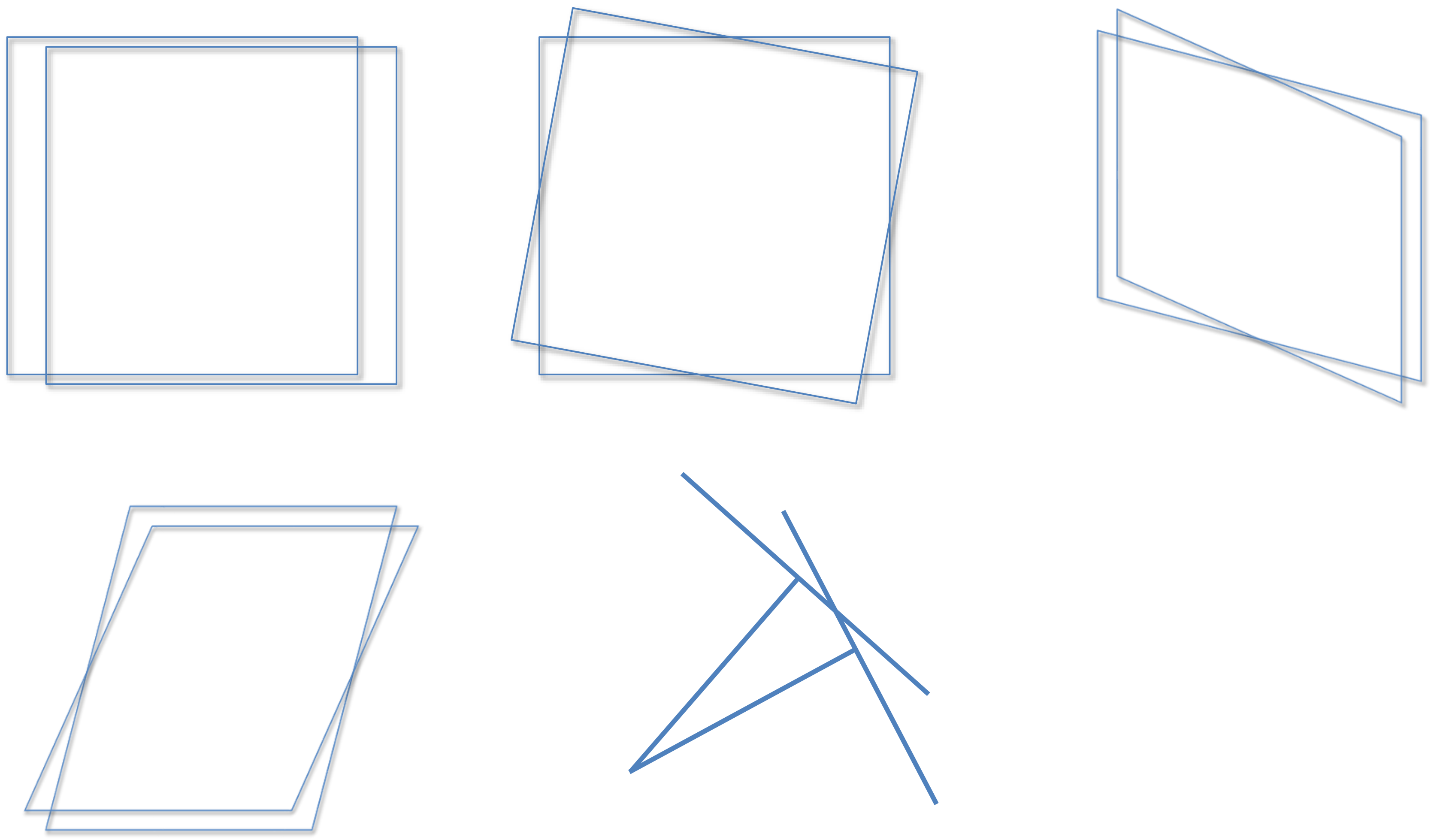
$$Wx = p$$

$$\sum_{j=1}^{n} w_{ij} x_j = p_i$$

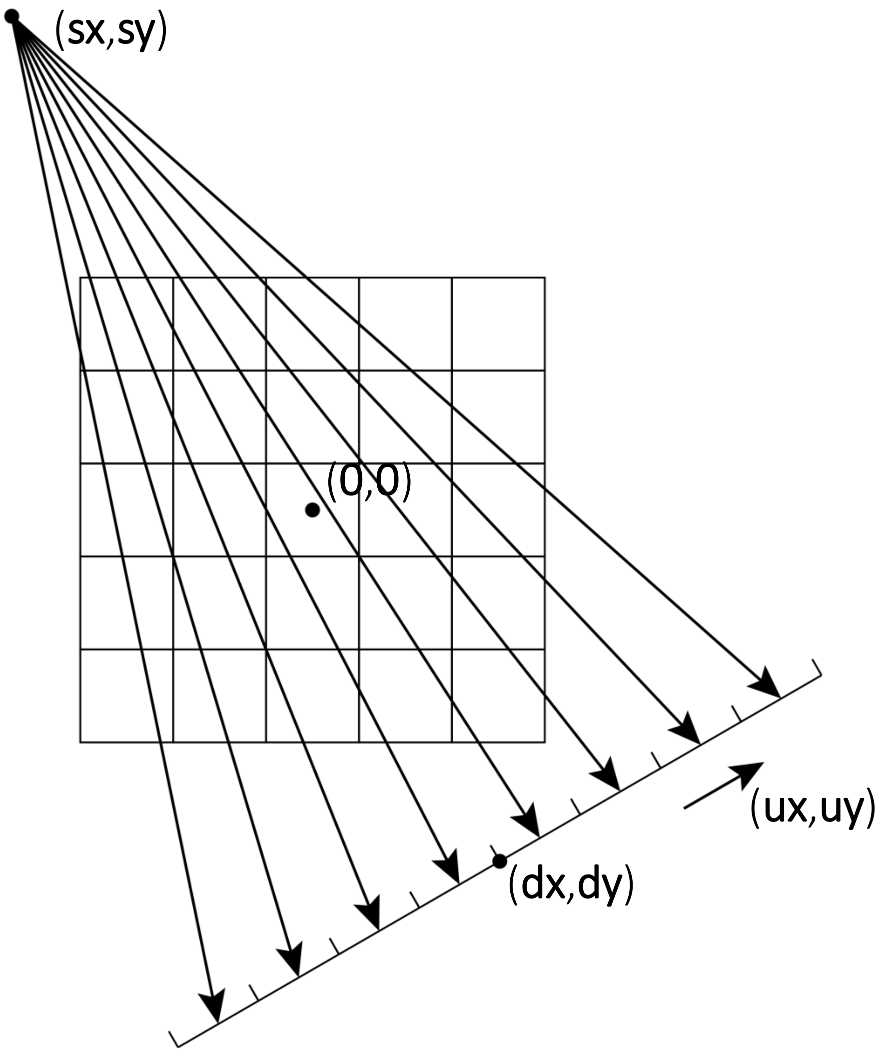
Toolbox concepts



Highlight: full 3D alignment



Flexible geometries



Basic usage: from Matlab or Python

- Either by calling ASTRA's Forward Projection (FP) and Backprojection (BP) operators manually;
- Or, directly call our reconstruction algorithms (FBP, SIRT, ...)

Reconstruction – SIRT

```
proj_geom = astra_create_proj_geom(‘parallel', 1.0, 1024, angles);
vol_geom = astra_create_vol_geom(1024, 1024);

sino_id = astra_mex_data2d('create', '-sino', proj_geom, sinogram);
rec_id = astra_mex_data2d('create', '-vol', vol_geom, 0);

cfg = astra_struct('SIRT_CUDA');
cfg.ProjectionDataId = sino_id;
cfg.ReconstructionDataId = rec_id;

alg_id = astra_mex_algorithm('create', cfg);

astra_mex_algorithm('iterate', alg_id, 100);

vol = astra_mex_data2d('get', rec_id);
```

Usage: with Matlab + Spot

Described in next talk, by Folkert Bleichrodt (CWI)

Basic usage: Python

- ASTRA calls very similar to Matlab version.
- See Python example scripts for exact syntax.

Part 2: New large-scale features

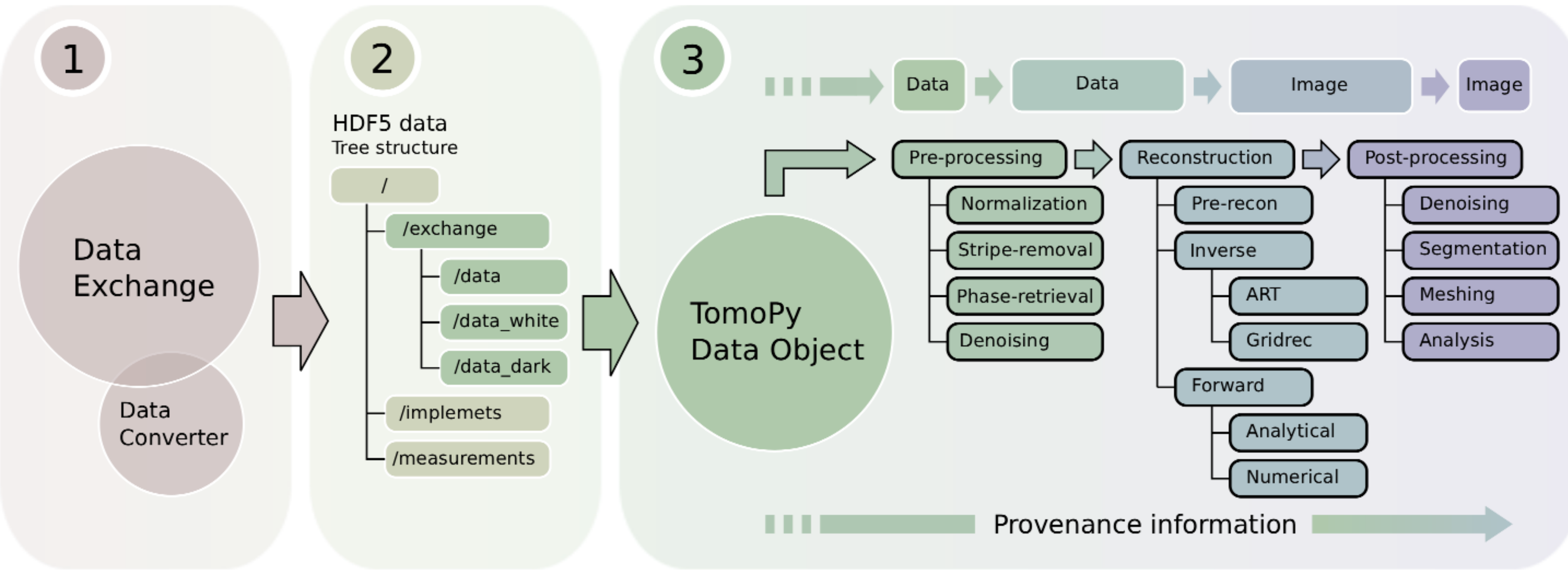
TomoPy

TomoPy provides

``a collaborative framework for the analysis of synchrotron tomographic data that has the goal to unify the effort of different facilities and beamlines performing similar tasks.''

Developed by APS at Argonne National Laboratory (USA).

TomoPy



1
Data Exchange
Data Converter
2
HDF5 data
Tree structure
/
/exchange
/data
/data_white
/data_dark
/implemets
/measurements
3
TomoPy Data Object
Data
Data
Image
Image
Pre-processing
Normalization
Stripe-removal
Phase-retrieval
Denoising
Reconstruction
Pre-recon
Inverse
ART
Gridrec
Forward
Analytical
Numerical
Post-processing
Denoising
Segmentation
Meshing
Analysis
Provenance information

TomoPy + ASTRA

With APS, Daniel Pelt (CWI) has integrated TomoPy and ASTRA.

This provides, from TomoPy:

- Data import/export
- Pre-/post-processing
  - Finding center of rotation, ring artefact correction, ...
- Parallelization of large data sets in parallel slices

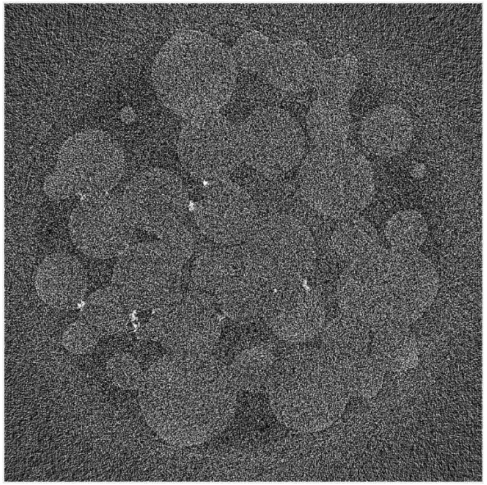
And from ASTRA:

- GPU acceleration
- Building blocks for your own reconstruction algorithms
- ...

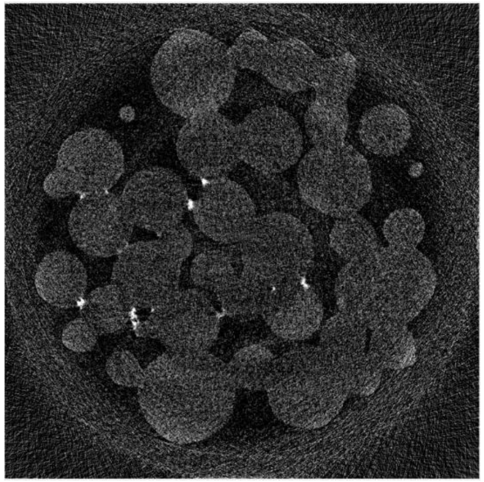
TomoPy + ASTRA

This interface is currently available for testing.

See the TomoPy documentation on how to get and install the necessary modules. (Search for TomoPy astra)



TomoPy gridrec



TomoPy preproc + ASTRA GPU SIRT

CWI

ASTRA for large data sets

Historically, a major limitation of ASTRA has been the requirement that data fits in GPU memory.

With ASTRA 1.7, we've started removing this limit.

Current status:

- 3D Forward and backprojection transparently split up data to fit
- Reconstruction algorithms don't (yet)

ASTRA for large data sets

For FP, BP this works transparently:

```
proj_geom = astra_create_proj_geom('parallel3d', 1.0, 1.0, 1024, 1024, angles);
vol_geom = astra_create_vol_geom(1024, 1024, 1024);

[x, projdata] = astra_create_sino3d_cuda(voldata, proj_geom, vol_geom);
[y, voldata] = astra_create_backprojection3d_cuda(projdata, proj_geom, vol_geom);
```

Also works transparently with Spot (next talk)

Upcoming feature (next release, "soon"):

- Multi-GPU support for 3D FP/BP

```
astra_mex('set_gpu_index', [0 1]);
```

Distributed ASTRA

Major new feature in ASTRA 1.7 (Dec 2015):

Distributed ASTRA: run ASTRA on a (GPU) cluster

- Process larger data sets faster
- A toolbox of building blocks for your (and our) own algorithms
- (Most of) the flexibility of ASTRA
- Python interface (only)

Developed by Jeroen Bédorf at CWI.

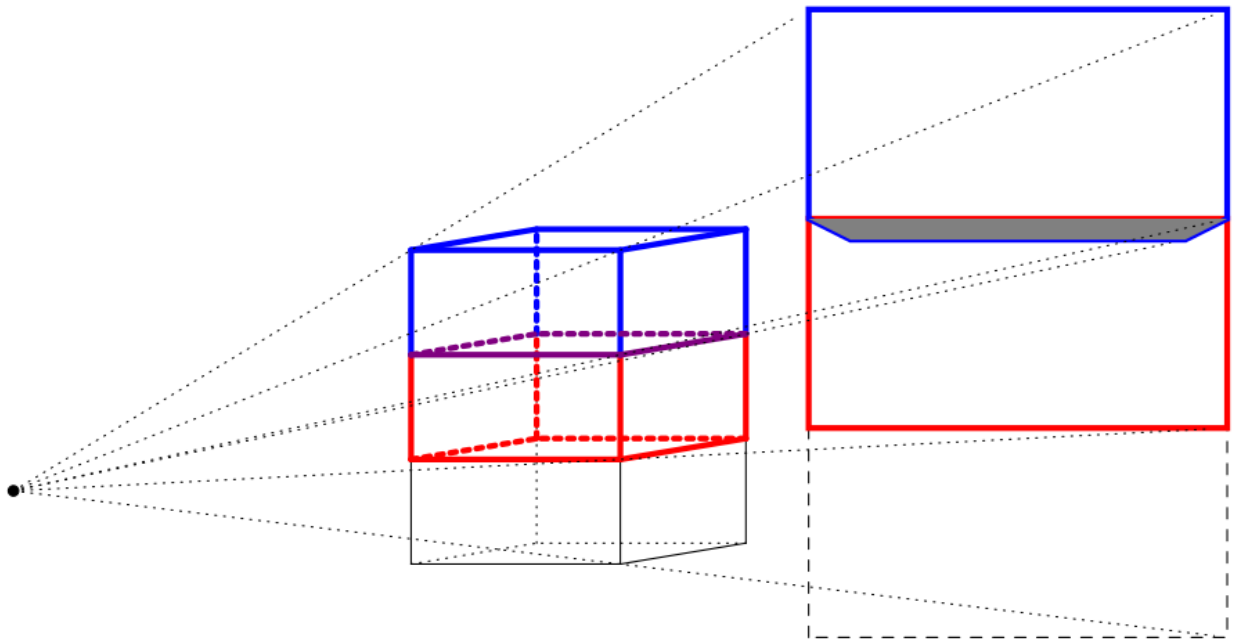
Distributed ASTRA

Our approach:

Data objects are distributed, and (some) 3D algorithms will automatically run distributed:

- SIRT
- CGLS
- Forward projection
- Backprojection

Distributed ASTRA



(Un)distributed ASTRA

```
# Configure geometry
angles = np.linspace(0, 2*np.pi, nAngles, False)
proj_geom = astra.create_proj_geom('cone', 1.0, 1.0,
                detectorRowCount, detectorColCount, angles,
                originToSource, originToDetector)
vol_geom = astra.create_vol_geom(vx, vy, vz)

# Data objects for input, output
proj_id = astra.data3d.create('-proj3d', proj_geom, P)
rec_id = astra.data3d.create('-vol', vol_geom)

# Configure algorithm
cfg = astra.astra_dict('SIRT3D_CUDA')
cfg['ReconstructionDataId'] = rec_id
cfg['ProjectionDataId'] = proj_id
alg_id = astra.algorithm.create(cfg)

# Run
astra.algorithm.run(alg_id, 100)
rec = astra.data3d.get(rec_id)
```

Distributed ASTRA

```
# Configure geometry
angles = np.linspace(0, 2*np.pi, nAngles, False)
proj_geom = astra.create_proj_geom('cone', 1.0, 1.0,
                detectorRowCount, detectorColCount, angles,
                originToSource, originToDetector)
vol_geom = astra.create_vol_geom(vx, vy, vz)

# Set up the distribution of data objects
proj_geom, vol_geom = mpi.create(proj_geom, vol_geom)

# Data objects for input, output
proj_id = astra.data3d.create('-proj3d', proj_geom, P)
rec_id = astra.data3d.create('-vol', vol_geom)

# Configure algorithm
cfg = astra.astra_dict('SIRT3D_CUDA')
cfg['ReconstructionDataId'] = rec_id
cfg['ProjectionDataId'] = proj_id
alg_id = astra.algorithm.create(cfg)
astra.algorithm.run(alg_id, 100)
rec = astra.data3d.get(rec_id)
```

Distributed ASTRA

It is also possible to run your own functions on distributed data:

```
def addScaledArray(src_id, dst_id, value):
 dataS = astra.data3d.get_shared_local(src_id)
 dataD = astra.data3d.get_shared_local(dst_id)
 dataD[:] = dataD + value*dataS2
```

```
mpi.run(addScaledArray, [tmp_id, vol_id, lambda])
```

Also functions that return output are possible, such as for example taking norms or inner products over the full volume.

Distributed ASTRA

Running your own functions on distributed data:

- Many possibilities:
  basic arithmetic, segmentation, inner products, norms, file I/O, …

- Some operations, such as edge detection, blurring, etc, look at multiple pixels at once, which can be in neighbouring volume blocks.

  We handle this by (optionally) letting the volume blocks overlap slightly.

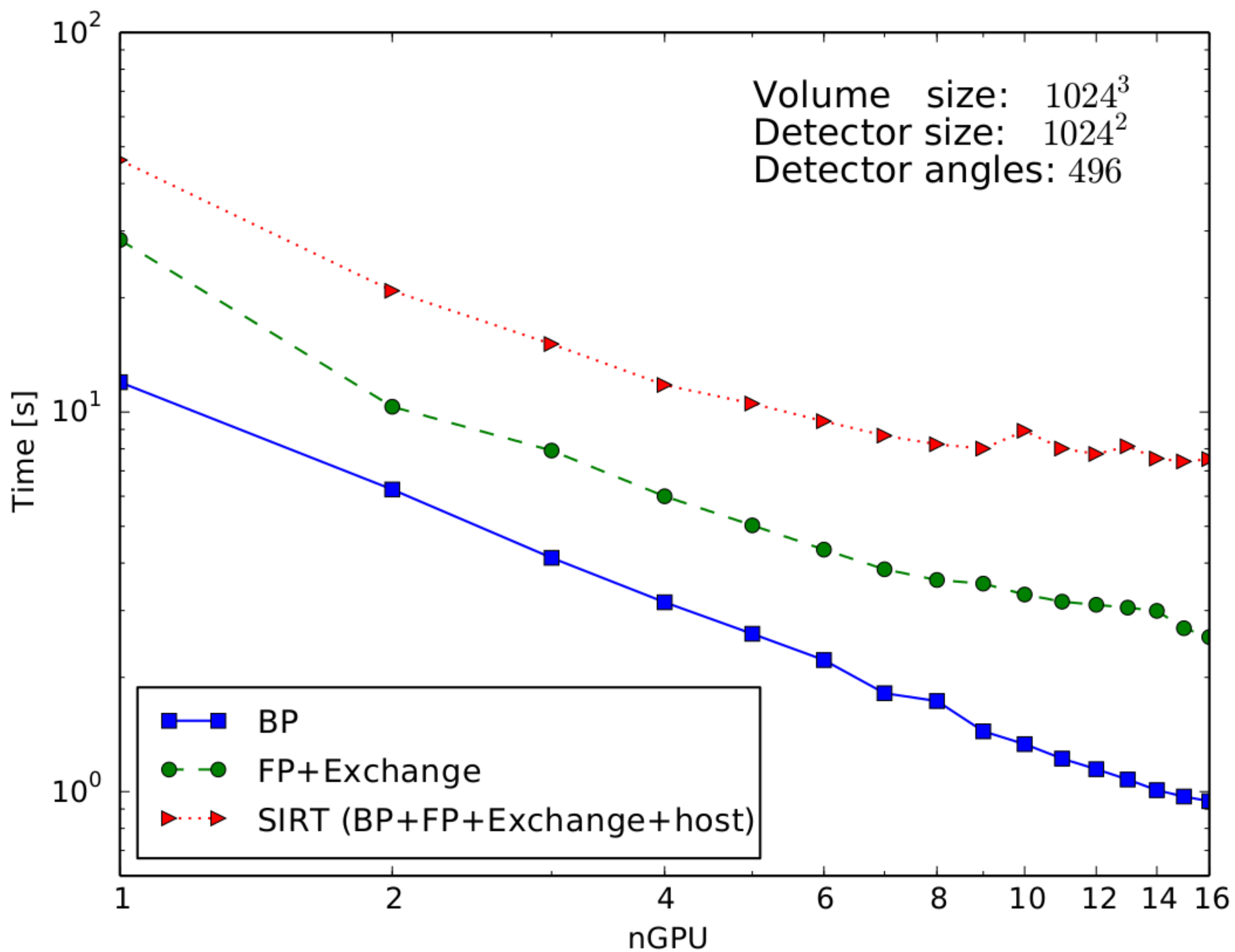
Distributed ASTRA – Limitations

Limitations:

- Python only
- Linux only

- Currently, we always only split volumes in slabs along the z-axis. For this to work, the projection geometry must (approximately) rotate around the z-axis.

Distributed benchmarks



Useful links

- Webpage, for downloads, docs:

  https://sourceforge.net/p/astra-toolbox/

- Github, for source code, issue tracker:

  https://github.com/astra-toolbox/

- Email:

  astra@uantwerpen.be

  willem.jan.palenstijn@cwi.nl

Acknowledgements

University of Antwerp

- Jan Sijbers
- Wim van Aarle

CWI

- Joost Batenburg
- Jeroen Bédorf
- Daniel Pelt
- Folkert Bleichrodt

Financial support from NWO, iMinds.

Publications

Palenstijn et al, "*Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs)*", **J. of Structural Biology**, 2011

van Aarle, Palenstijn et al, "*The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography*", **Ultramicroscopy**, 2015

Palenstijn, Bédorf et al, "*A distributed SIRT implementation for the ASTRA Toolbox*", **Proceedings of Fully3D 2015**