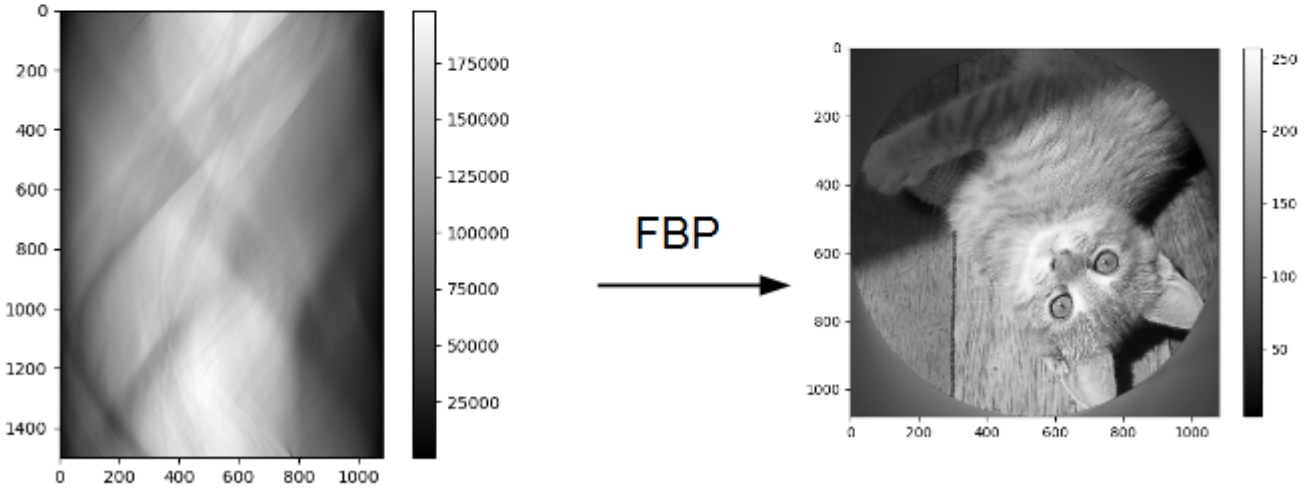
5th Silx Code Camp
September 12, 2017

THIS TALK

- Introduction
  - Novelties
- Status of silx
- Goals of the code camp
  - For users
  - For core developers
- Hands on!

Filtered Back Projection in silx

- Filtered Back-Projection (**FBP**) is the usual reconstruction method in (parallel) tomography
- silx now provides a FBP module
- The filtering can be omitted if the data is already filtered
- Works on both GPU and CPU (Mac OS is not supported)



sinogram

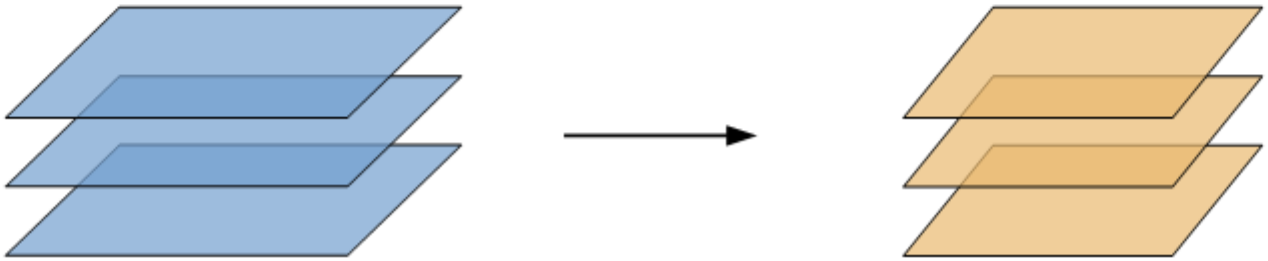
slice

Filtered Back Projection in silx

- Principle : define a geometry and use it to reconstruct one or several sinograms.
- Geometry = sinogram shape, [series of angles, slice shape, rotation center position]

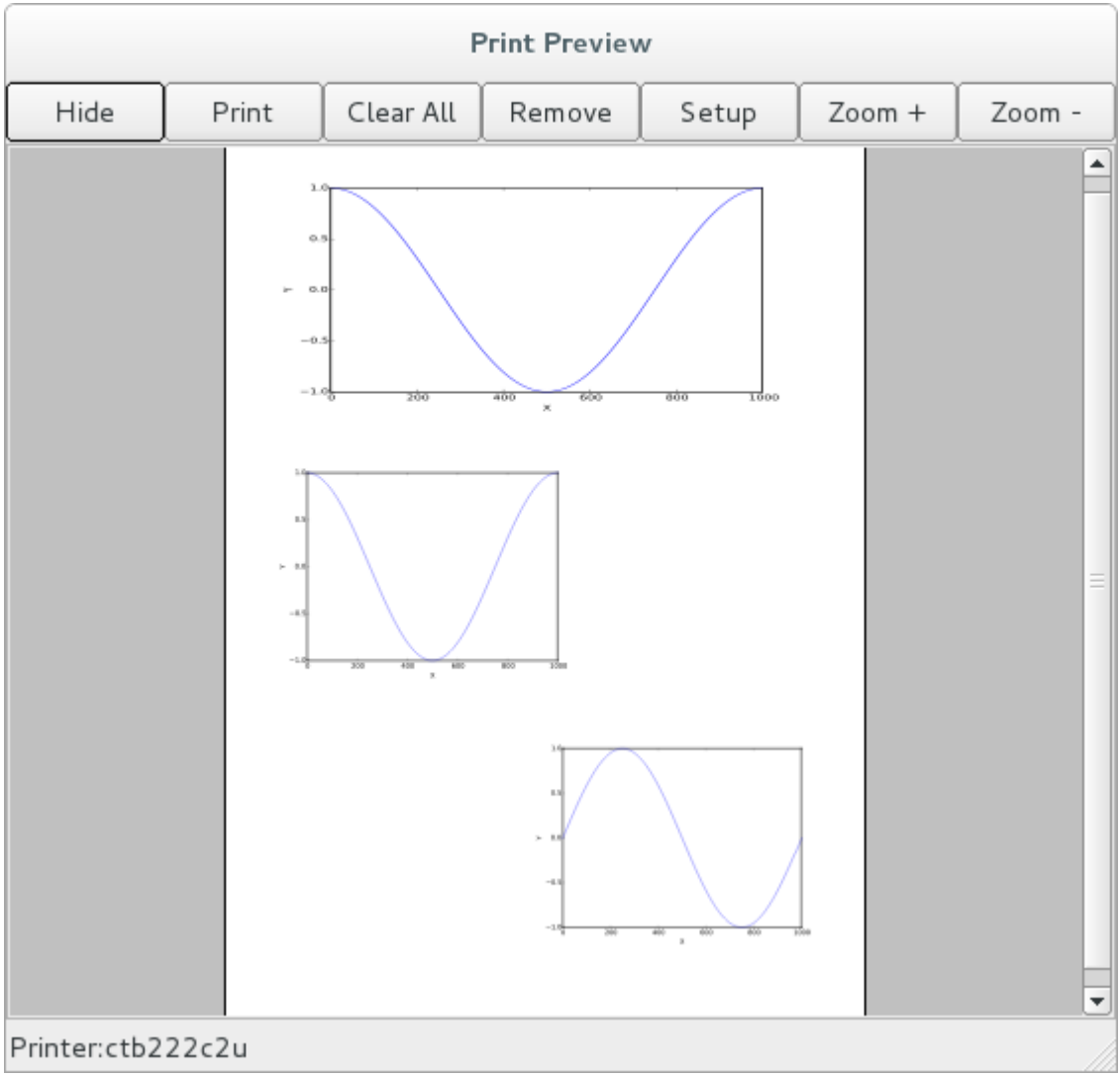
```
from  silx.opencl.backprojection import Backprojection
# Compute the tomography geometry
tomo_geometry = Backprojection(sinograms_stack.shape[1:],
                               axis_position=1337,
                               devicetype='GPU')

# Allocate the memory for volume reconstruction
num_sinos = sinograms_stack.shape[0]
reco = np.zeros((num_sinos,) + tomo_geometry.shape)
# Reconstruct
for i in range(num_sinos):
    reco[i] = tomo.fbp(sinograms_stack[i])
```



Print Preview

- Print preview dialog (with addImage, addPixmap and addSvgItem methods)
- Tool button for a plot widget *(to send the plot as an SVG item)*
- Items can be dragged and resized. (*Geometry can be configured prior to send the plot).*



silx.io.commonh5

- This new module provides a common base for *silx.io.spech5* and *silx.io.fabioh5* to provide the h5py-like API for various data formats.

- If new formats are handled by silx in the future, and they inherit the commonh5 classes, they will benefit from the existing tools:
  - *silx.io.convert*
  - *silx.io.utils* (is_dataset, is_group, is_file,…)

silx.io.convert

- Module

- Before only SPEC files could be converted (*silx.io.spectoh5*)
- New *silx.io.convert* supports Fabio images (replaces *spectoh5)*

- Application

- New command line application to convert files to HDF5

silx convert –help
silx convert filename

SpecFile Licence

- The SpecFile C library license changed from LGPL to MIT

- This makes *silx* entirely MIT

Median Filter (silx.math.medianfilter)

Previously only 'nearest' mode.

Cpp Implementation of 'reflect', 'mirror' and 'shrink' modes.

6	7	4
8	8	5
8	7	4

input

kernel size = 5

Treatment of the value '6'

6	6	6	7	4	4	4
6	6	6	7	4	4	4
6	6	6	7	4	4	4
8	8	8	8	5	5	5
8	8	8	7	4	4	4
8	8	8	7	4	4	4
8	8	8	7	4	4	4

nearest

4	7	8	7	4	7	8
5	8	8	8	5	8	8
4	7	6	7	4	7	6
5	8	8	8	5	8	8
4	7	8	7	4	7	8
5	8	8	8	5	8	8
4	7	6	7	4	7	6

mirror

8	8	8	8	5	5	8
7	6	6	7	4	4	7
7	6	6	7	4	4	7
8	8	8	8	5	5	8
7	8	8	7	4	4	7
7	8	8	7	4	4	7
8	8	8	8	5	5	8

reflect

6	7	4
8	8	5
8	7	4

shrink

```
from silx.math import medianfilter
import numpy

img = numpy.random.rand(48, 48)
medianfilter.medfilt2d(image=img, kernel_size=3, conditional=False, mode='reflect')
```

Actions Refactoring (silx.gui.plot.actions)

Replacement of ***silx.gui.plot.PlotActions*** by the ***silx.gui.plot.actions*** module.

silx.gui.plot.actions
- control.py
  - `KeepAspectRatioAction` class
  - `ResetZoomAction` class
  - `XaxisLogarithmicAction` class
  - `ZoomBackAction` class
  - ...
- fit.py
  - `FitAction` class
- histogram.py
- io.py
- medfilt.py
- PlotAction.py

API for PlotActions <=0.5 is preserved but deprecated.
Each new PlotAction should be based on the new design.

Colormap Object (silx.gui.plot.Colormap)

Colormaps are now defined as a ***Colormap*** object instead of a dictionary.

This allow modifications on colormaps objects to be managed by other classes such as ***PlotWidget*** or ***ColorBar*** (using Qt.Signal).

```
from silx.gui.plot.Colormap import Colormap

colormap = Colormap(name='temperature',
                    normalization=Colormap.LOGARITHM,
                    vmin=None,
                    vmax=None)
```

API with colormaps as a dictionary is kept but deprecated.

PlotWidget axis

- Provide a plot axis API

  axes =plot.getXAxis(), plot.getYAxis()

  - Provides getters, setters
  - Signals on limits, scale, label, direction

- Constraints on axes

  xaxis.setLimitsConstraints(minPos, maxPos)

  xaxis.setRangeConstraints(minRange, maxRange)

  - A demo is available at *examples/plotLimits.py*

- Helper to synchronize axes

```
from silx.gui.plot.utils.axis import SyncAxes
sync = SyncAxes([plot1.getXAxis(),
                 plot2.getXAxis(),
                 plot3.getXAxis()])
```

  - A demo is available at *examples/syncaxis.py*

Silx resources

A project can use silx as resource provider

```
import silx.resources

PYFAI_RESOURCE_DIR = None  # It has to be set for Debian package

silx.resources.register_resource_directory(
    "pyfai",
    pyFAI.resources,
    forced_path=PYFAI_RESOURCE_DIR)

filename = silx.resources.resource_filename("pyfai:calibrant/LaB6.C")

import silx.opencl.utils
filename = silx.opencl.utils.get_cl_file("pyfai:opencl/integrate")

import silx.gui.icons
icon = silx.gui.icons.getIcons("pyfai:icons/pyfai")
```

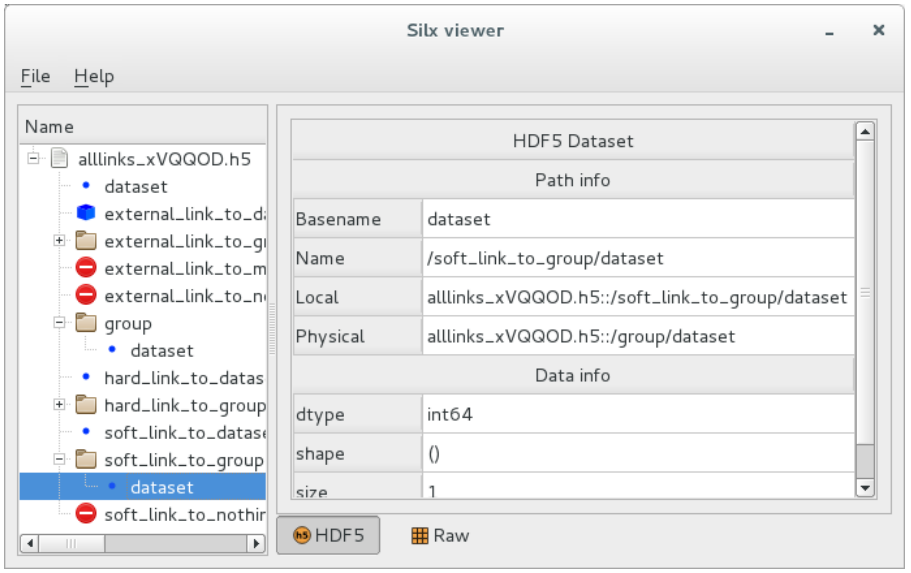
Improvements to silx view

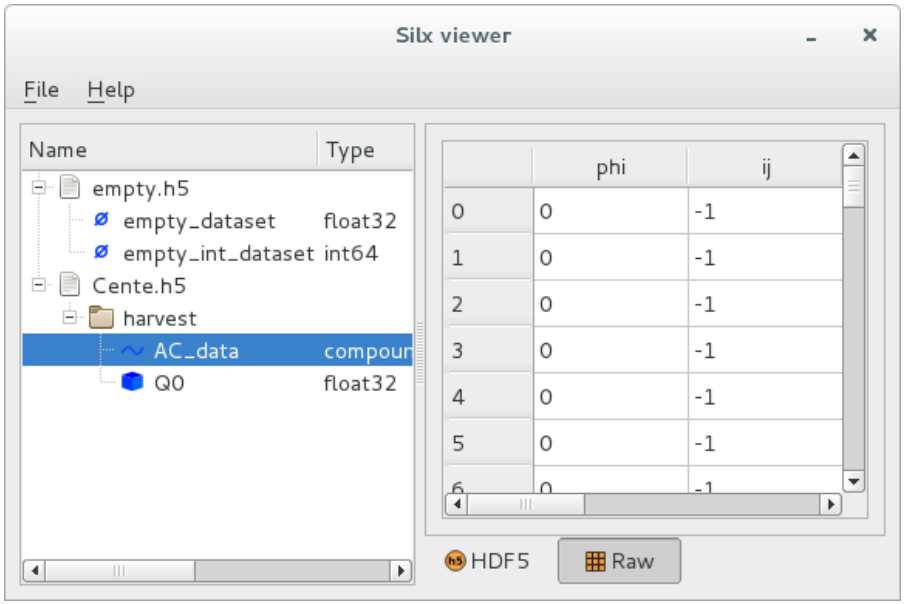
- Support empty dataset *(shape=None)*
  f["d"] = h5py.Empty(dtype='f'))

- Fix time consumption when displaying big structured arrays (aka. HDF5 database)

- Improve HDF5 property view
  - *Display accurate node path with and without link resolution*

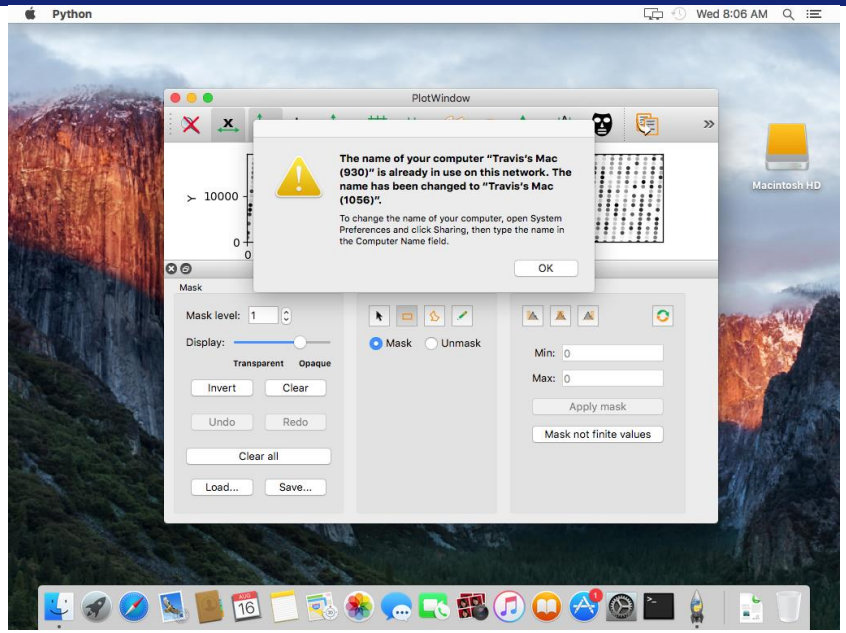
- Display debug information on the logs (*--debug*)



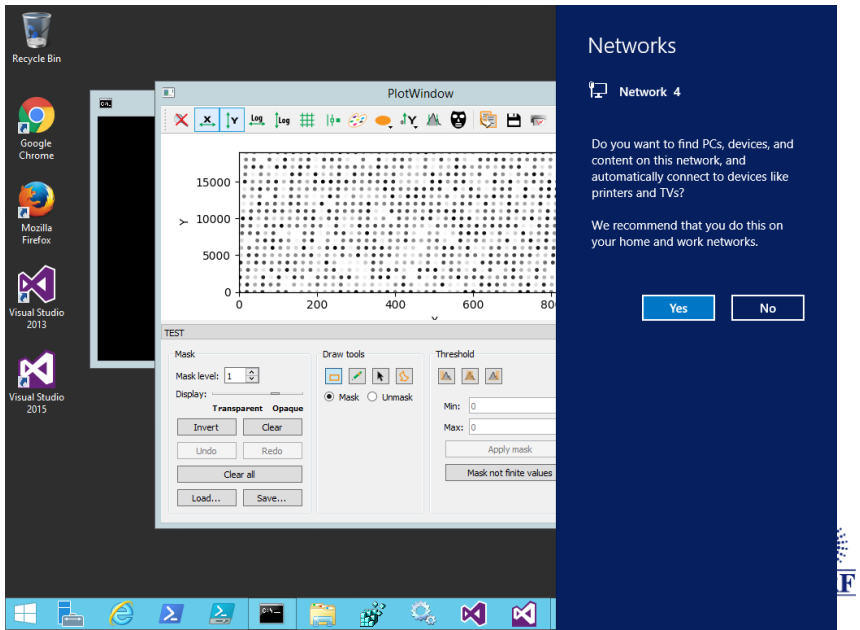


Continuous Integration Improvements

- Fix compilation flags to be able to test C/C++ assertion of cython extensions.



- Tool to close system popups.



Improvements for other projects

- Cleanup logging information

  https://docs.python.org/2/howto/logging.html#configuring-logging-for-a-library

- Deprecation messages are logged only once

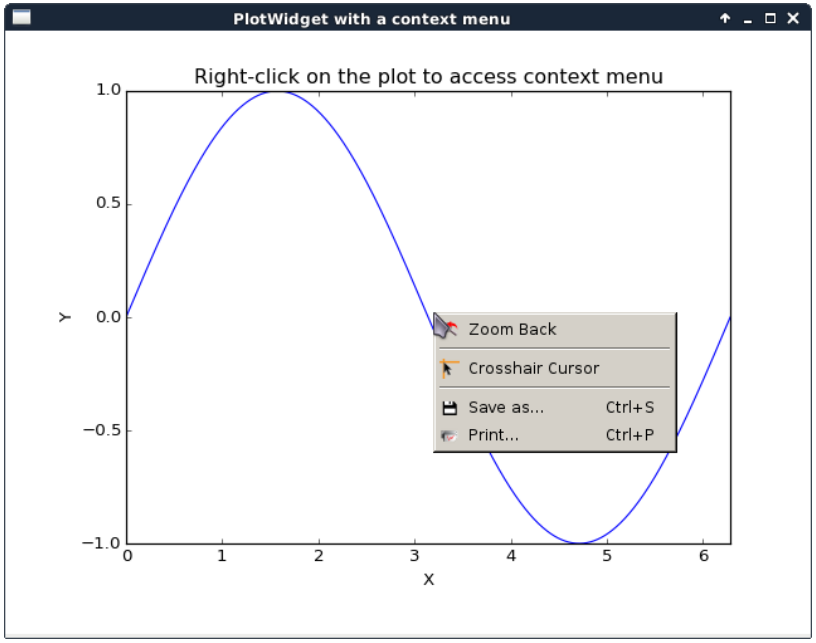
- Provide matplotlib Qt backend access

```
from silx.gui.plot.matplotlib import backend
backend.FigureCanvasQTAgg
```

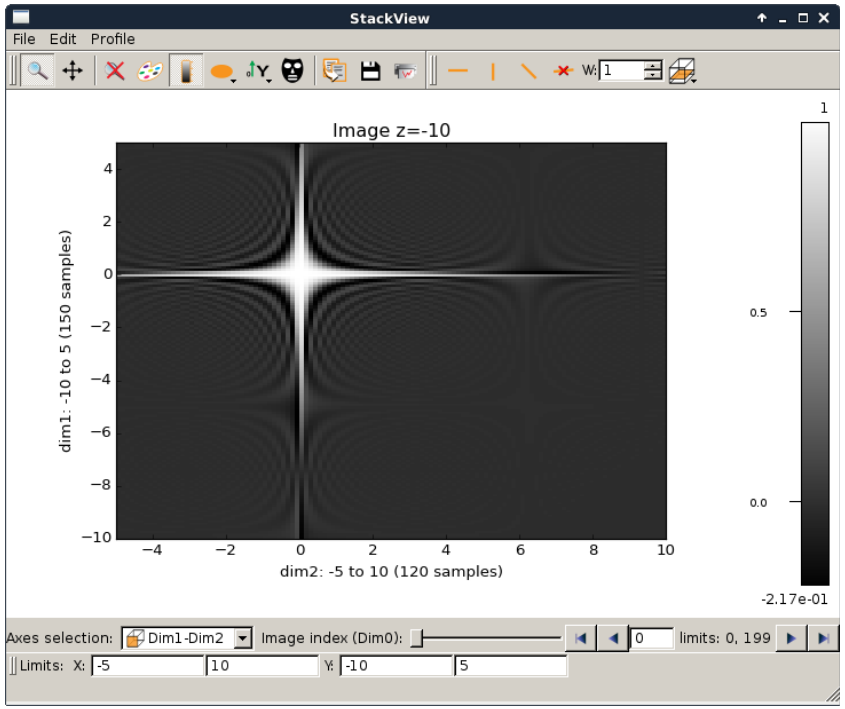
or

```
from silx.gui.plot.matplotlib import FigureCanvasQTAgg
```

- PlotWidget: Add support for context menu:
  plotContextMenu.py



- PlotWindow, Plot2D
  - *Add colorbar*



silx.gui.plot API

- Add signals on *PlotWidget* items (i.e. curves, images, markers,…) notifying updates: *sigItemChanged*

- Internals: Merged classes *Plot* and *PlotWidget*

OpenGL in *plot3d* and *plot*

- Support for Qt >= 5.4 OpenGL Widgets *(QOpenGLWidget)*

- Better support of OpenGL context issues (i.e. missing QtOpenGL, ssh GLX forwarding disabled,…) : display an error message rather than raising exceptions.

- First steps of Continuous Integration for OpenGL-based widgets

Structure of silx

- gui: Graphical User Interface widgets
  - Plot, image display, masks, HDF5 tree view, fitting
- image: Image processing tools
  - Image interpolation, registration and drawing primitives
- io: Input / Ouput
  - Support for SPEC, HDF5 and image formats
- math:
  - least squares fit with constraints, isosurface calculations, histograms, …
- opencl: Optimize the use of GPU
- third-party: External utilities
- utils: Internal utilities
- sx: Convenience module for interactive use

Plot: Object API

When getting a curve or an image from a Plot widget in silx, it used to return a list describing this item.

- Since v0.5.0 it returns an object:
  - Add support for updating items in the Plot:
    curve, image, markers...
  - Mostly backward-compatible with previous API

- Documentation:

http://www.silx.org/doc/silx/dev/modules/gui/plot/items.html

Plot: Object API

- Example: Getting image information:

```
from silx import sx
w = sx.imshow(img)
```

- Object API:

```
image = w.getActiveImage()
data = image.getData(copy=True)
scale = image.getScale()
```

- Legacy API:

```
image = w.getActiveImage()
data = image[0]
scale = image[4]['scale']
```

Plot: Object API

Example: Updating an image:

```
from silx import sx
w = sx.imshow(img)
```

- Object API:

```
image = w.getActiveImage()
image.setScale(2., 2.)
```

- Legacy API:

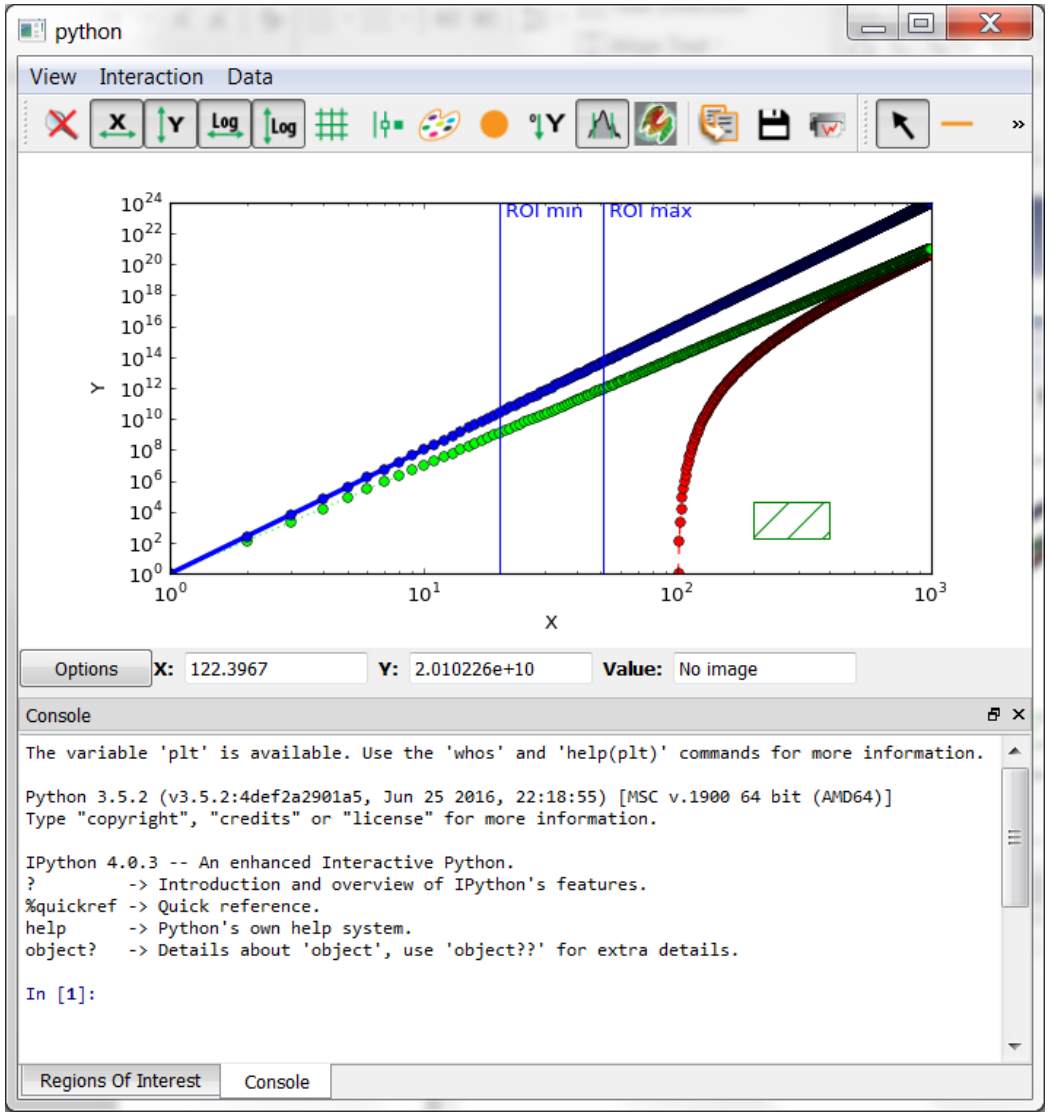
```
data, legend, info, pixmap, params = w.getActiveImage()
w.addImage(data,
           legend=legend,
           info=info,
           pixmap=pixmap,
           scale=(2., 2.))
```

silx.gui: Plot 1D

- Visualize 1D data
- Apply ROIs on them
- Control the plot via an interactive console
- Fitting capabilities
- Object oriented API

silx.gui: base widgets for scientific applications

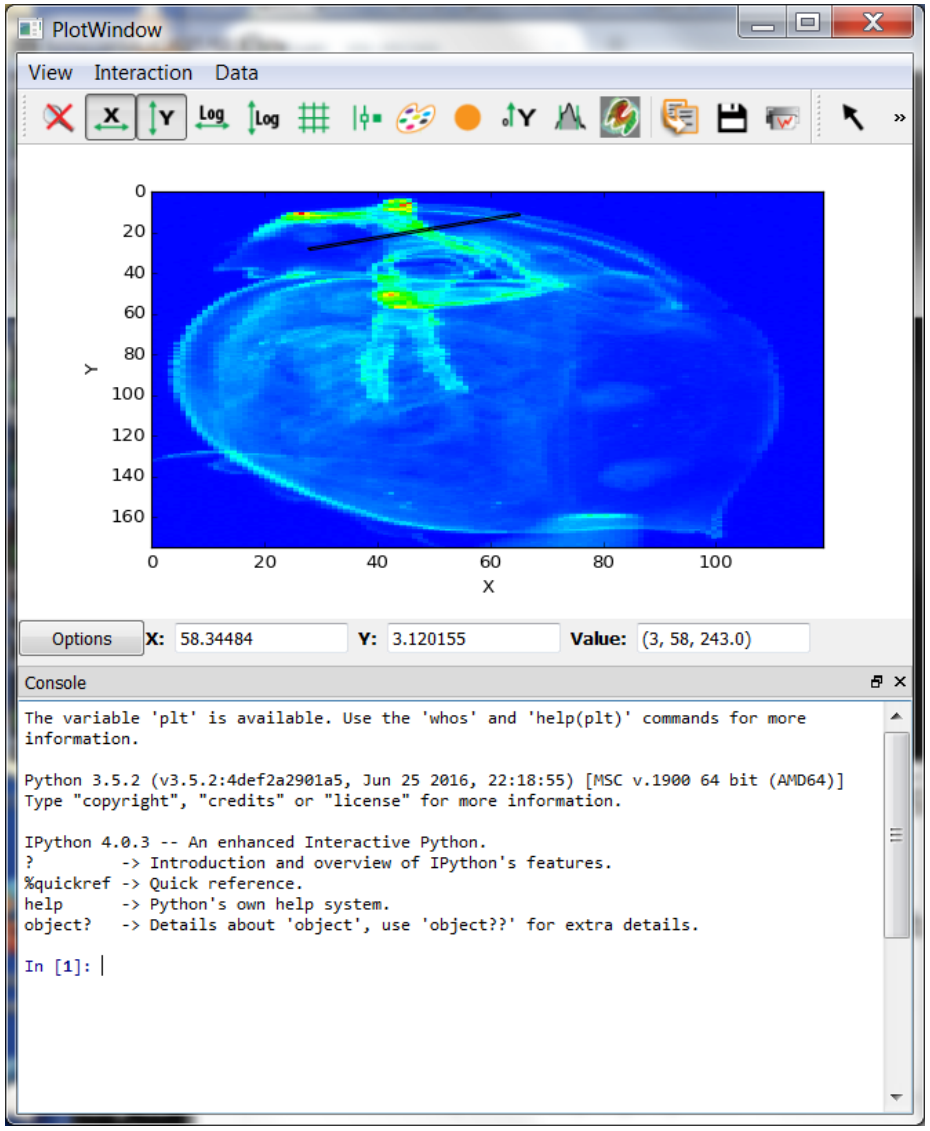
- Browsing file contents
  - Single widget for HDF5, SPEC, Images
- Plotting curves
  - with ROI, fitting
- Display of images
  - with masks, profiles
- Interactive console

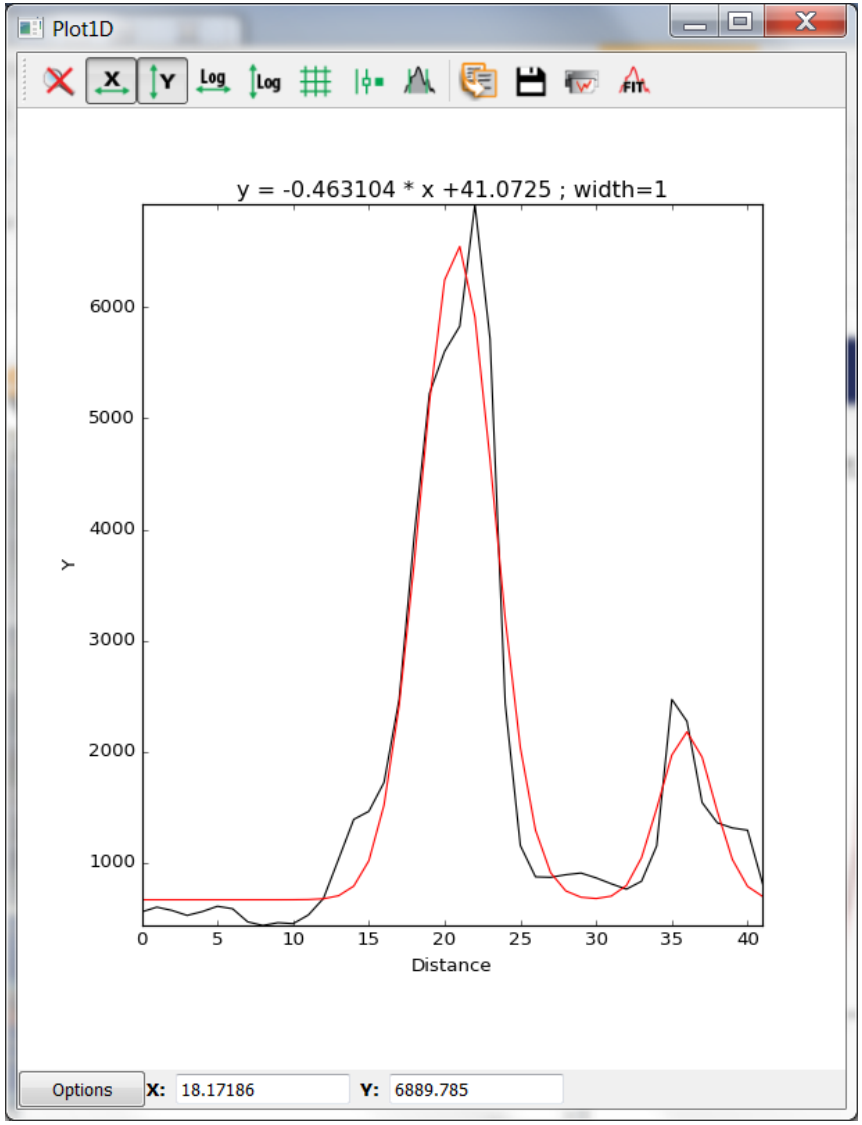


silx.gui: Plot 2D

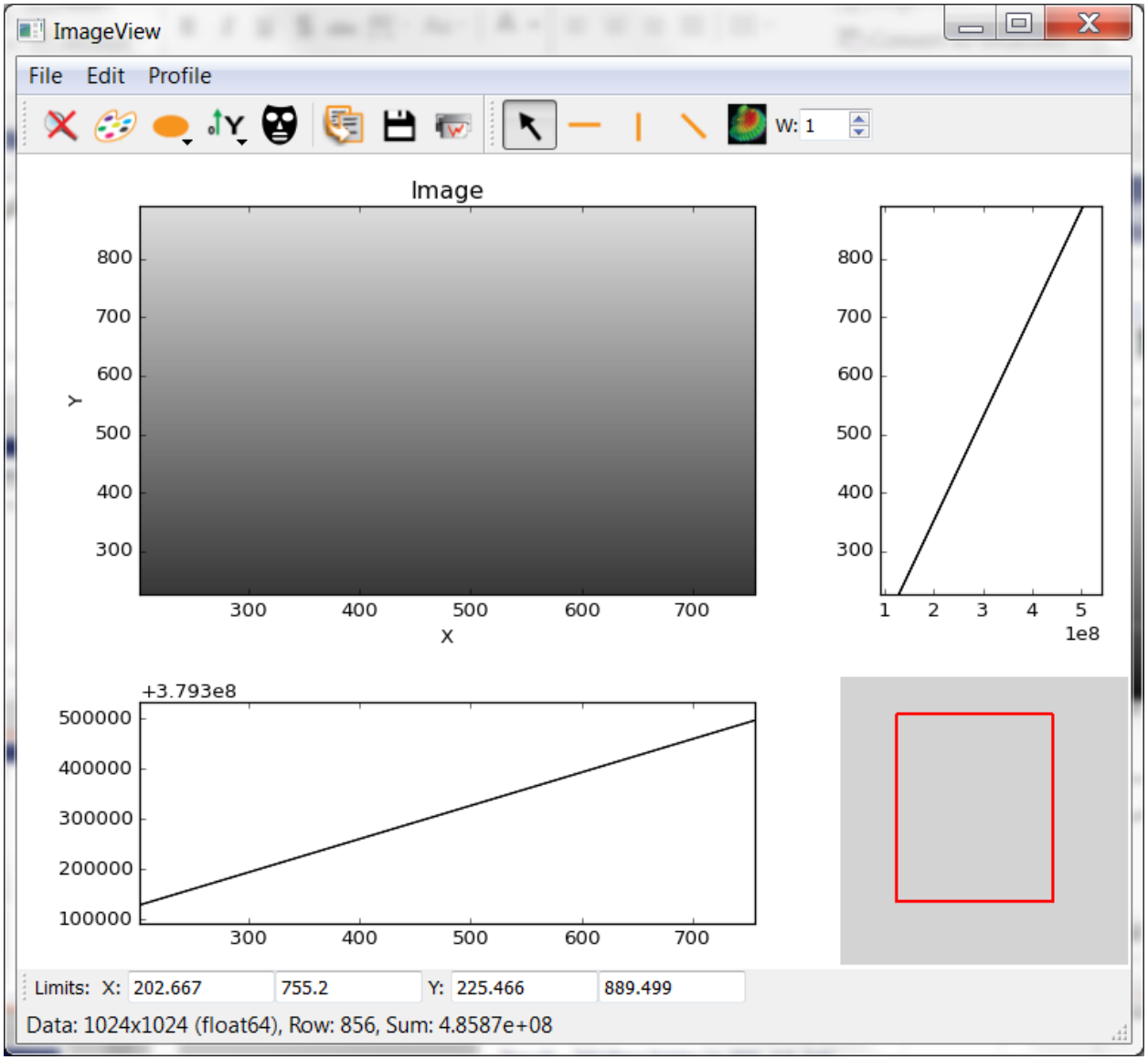
- Visualize 2D data (Images and Stacks of Images)
  - Support Median Filters, Profiles and Masks on them
- Visualize 3D data as scatter plots
  - Support Masks on them
- Apply different colormaps
- Plot an image with associated histograms
- Visualize 3D scalar fields (Isosurfaces)

Full-featured widgets





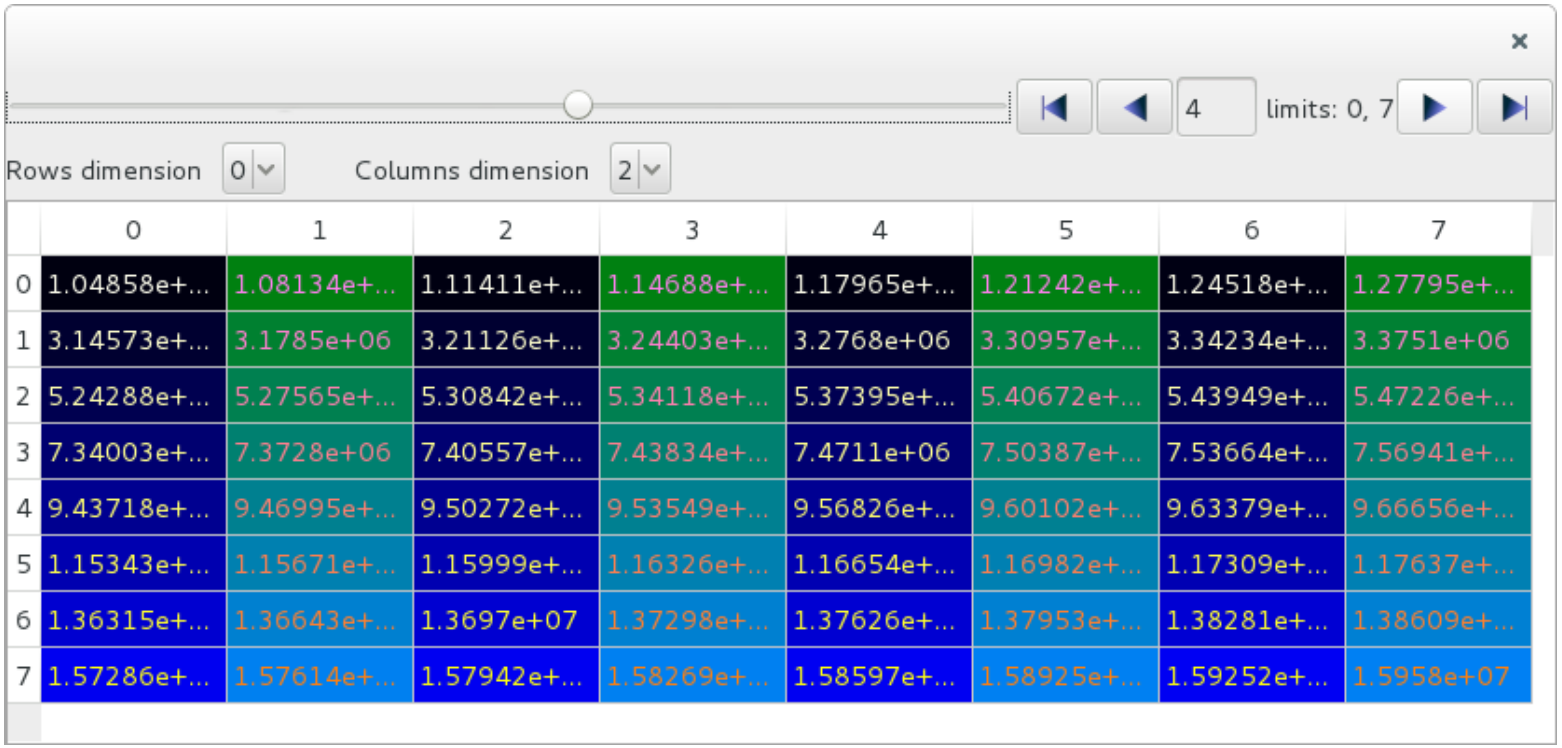
Full-featured Widgets



silx.gui.data.ArrayTableWidget

- Display arrays and datasets of any number of dimensions in a TableView

- Lazy loading for datasets: only the currently displayed 2D slice is read from HDF5 file

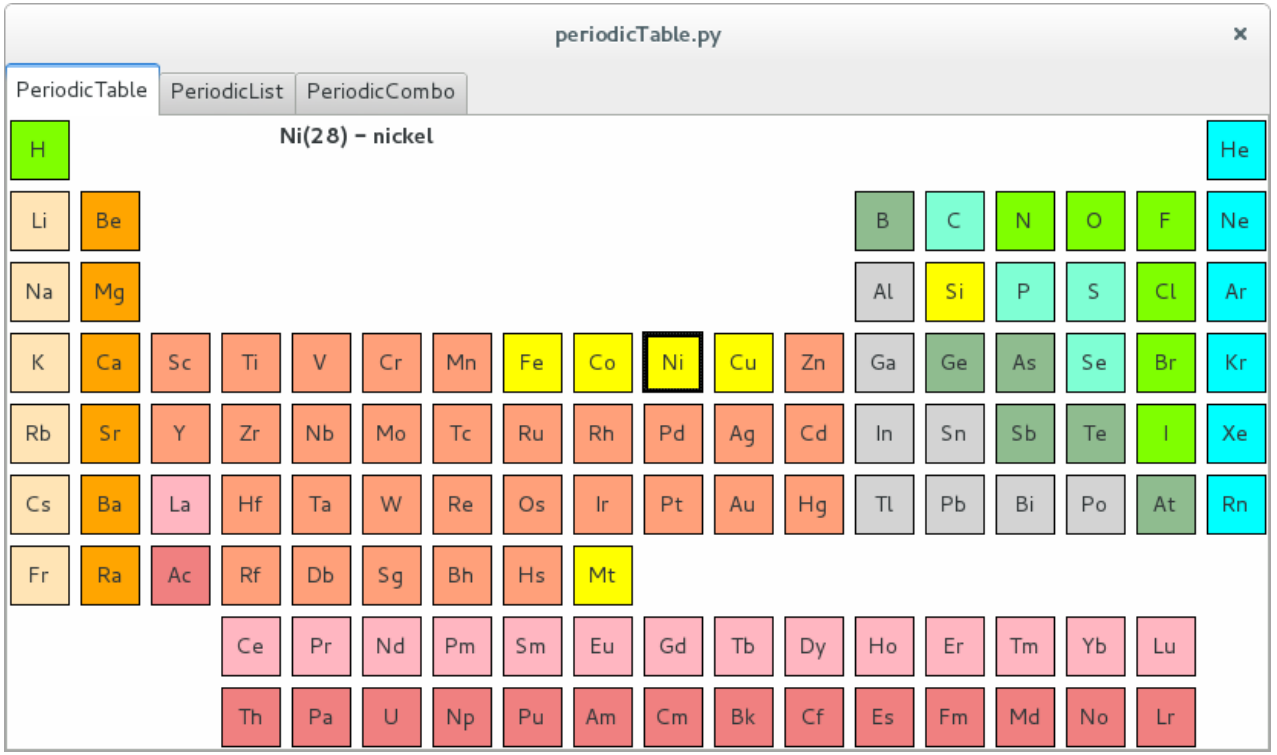


	0	1	2	3	4	5	6	7
0	1.04858e+...	1.08134e+...	1.11411e+...	1.14688e+...	1.17965e+...	1.21242e+...	1.24518e+...	1.27795e+...
1	3.14573e+...	3.1785e+06	3.21126e+...	3.24403e+...	3.2768e+06	3.30957e+...	3.34234e+...	3.3751e+06
2	5.24288e+...	5.27565e+...	5.30842e+...	5.34118e+...	5.37395e+...	5.40672e+...	5.43949e+...	5.47226e+...
3	7.34003e+...	7.3728e+06	7.40557e+...	7.43834e+...	7.4711e+06	7.50387e+...	7.53664e+...	7.56941e+...
4	9.43718e+...	9.46995e+...	9.50272e+...	9.53549e+...	9.56826e+...	9.60102e+...	9.63379e+...	9.66656e+...
5	1.15343e+...	1.15671e+...	1.15999e+...	1.16326e+...	1.16654e+...	1.16982e+...	1.17309e+...	1.17637e+...
6	1.36315e+...	1.36643e+...	1.3697e+07	1.37298e+...	1.37626e+...	1.37953e+...	1.38281e+...	1.38609e+...
7	1.57286e+...	1.57614e+...	1.57942e+...	1.58269e+...	1.58597e+...	1.58925e+...	1.59252e+...	1.5958e+07

silx.gui.widgets.PeriodicTable

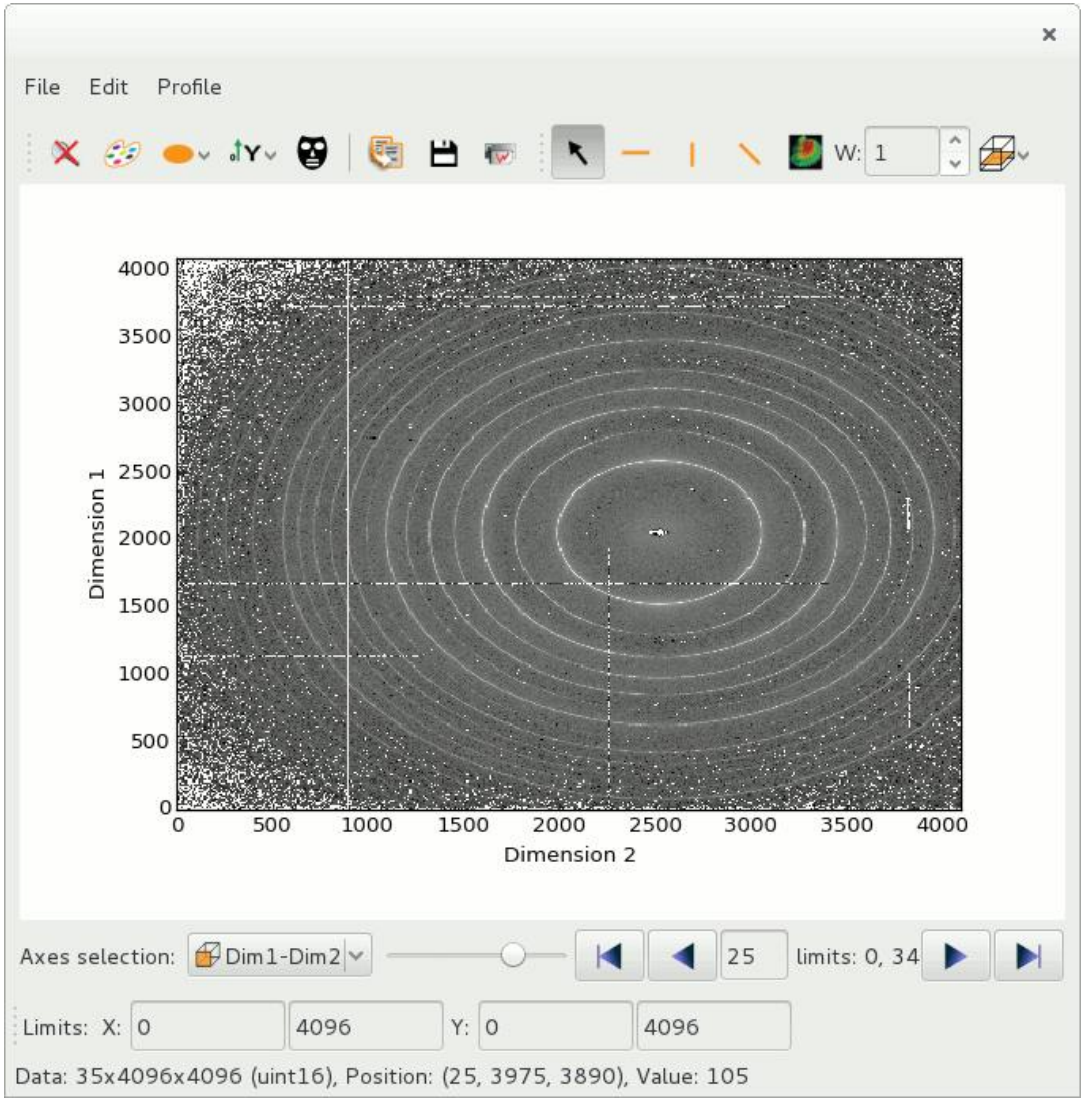
- Periodic table, list (QTreeView) and combo/dropdown list providing minimal data for elements: symbol, name, atomic number, mass

- Selectable elements, signals for element clicked and selection changed events



silx.gui.plot.StackView

- Viewing 3D arrays, 3D datasets or list of 2D arrays as a stack of images.
- Axes selection
- Profile tool to extract a 2D slice from the 3D stack
- Lazy loading for datasets (except when doing diagonal 3D profile)



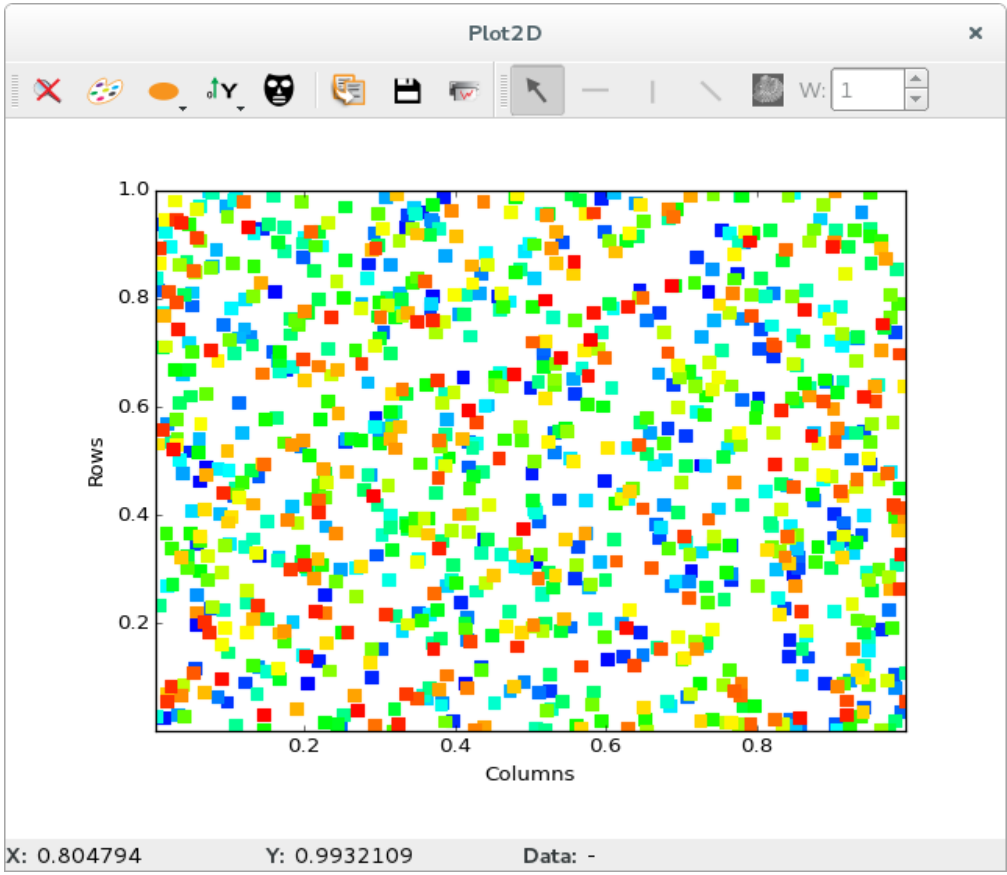
silx.gui.plot Scatter Objects

```
import numpy
import sys
from silx.gui import qt
from silx.gui.plot import Plot2D

app = qt.QApplication([])
win = Plot2D()

win.addScatter(x=numpy.random.random(1000),
               y=numpy.random.random(1000),
               value=numpy.arange(1000),
               legend="my scatter")

sc = win.getScatter("my scatter")
sc.setSymbol("s")                          # square
sc.setSymbolSize(50)
sc.setColormap({'name': 'temperature',
                'normalization': 'linear',
                'autoscale': True,
                'vmin': 0.0, 'vmax': 1,})
win.resetZoom()
win.show()
sys.exit(app.exec_())
```



silx.gui.plot OpenGL Rendering

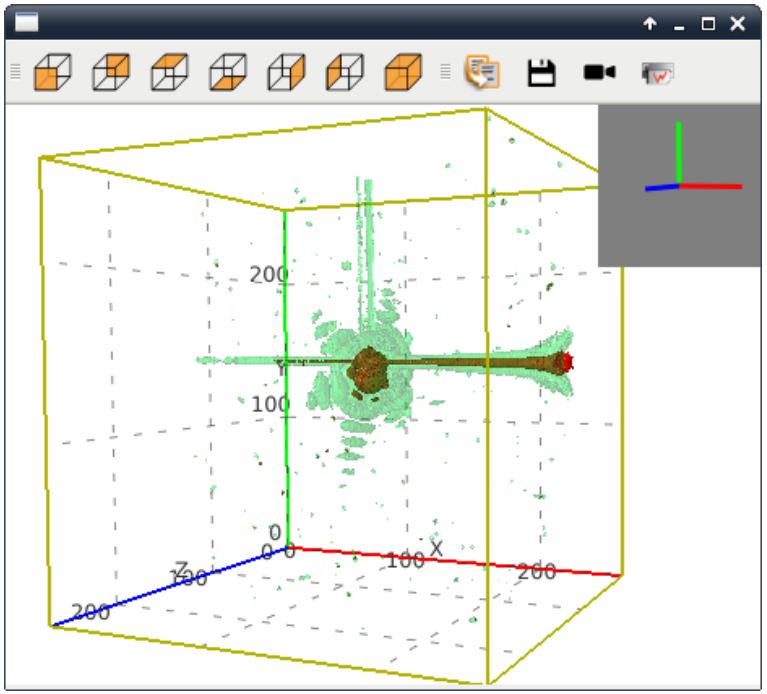
Matplotlib and OpenGL rendering backends in silx.gui.plot widgets:

- Usage: Set argument backend='gl‘ in widget constructor for:
  PlotWidget, PlotWindow, Plot1D, Plot2D, StackView, ImageView

- Example:

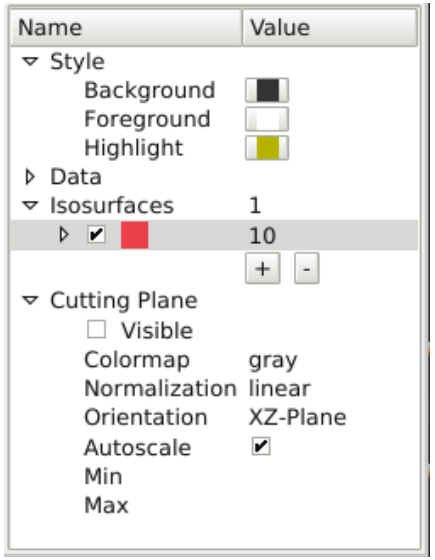
```
from silx import sx
plot = sx.Plot2D(backend='gl')
plot.show()
```

silx.gui.plot3d



First version of silx 3D visualisation:

- Dependencies:
  - PyQt.QtOpenGL
  - PyOpenGL 3.x
  - OpenGL 2.1 subset
- Qt widgets for 3D plotting:
  - ScalarFieldView (scalar field visualisation)
    - Iso-surfaces
    - Cutting plane
- Based on an internal 3D scene structure.

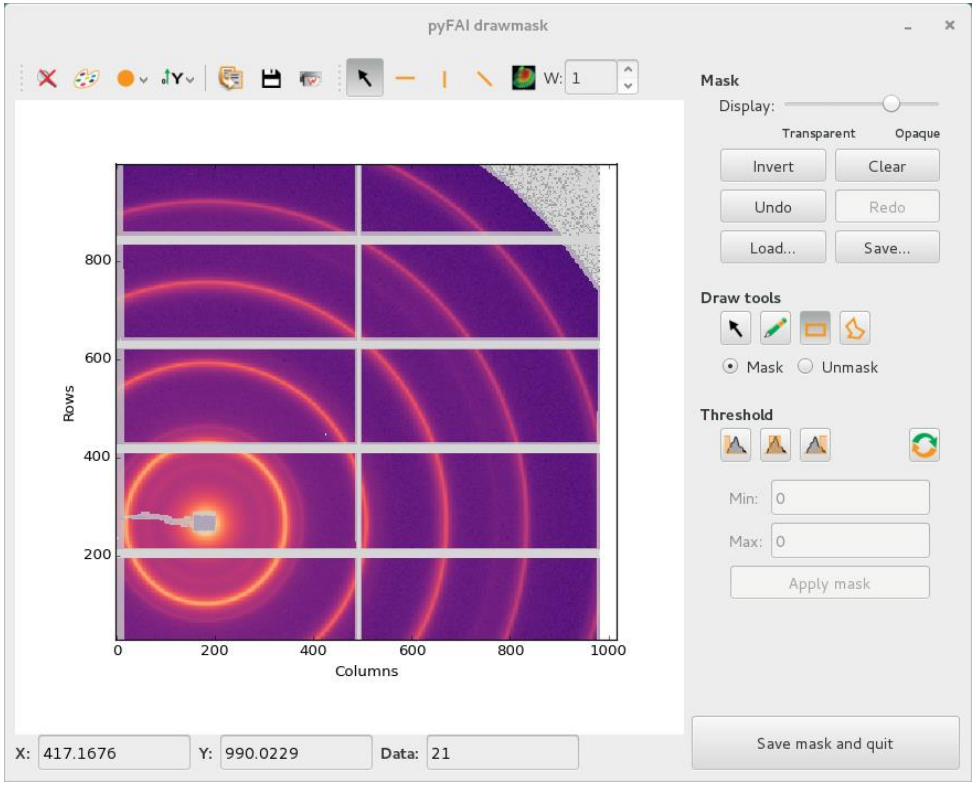


silx.math: miscellaneous mathematical functions

- Non-linear least squares with constraints on fitting parameters
  - Has a configuration widget for easy integration into GUIs
- 1D peak search
- Isosurface calculations with Marching Cubes algorithm
  - For 4D visualization (visualization of scalar fields)
- N-dimensional histograms based on look-up tables
- Fitting functions with automatic estimation of initial parameters
- 1D and 2D median filters

silx.image: image processing tools

- Basic shapes for masks
  - Line profiles
  - Polygons
  - Circle
- Bilinear interpolation
  - Used to scale up/down images to display
- Gaussian blurring of images
  - GPU accelerated via OpenCL
- Image registration and alignment (SIFT)
  - GPU accelerated via OpenCL
- Median Filter
  - GPU accelerated via OpenCL

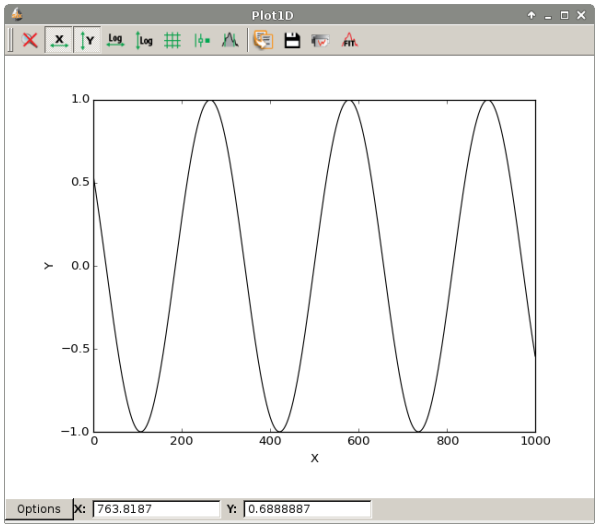


silx.sx: a module to simplify interactive use

pylab like module on steroids

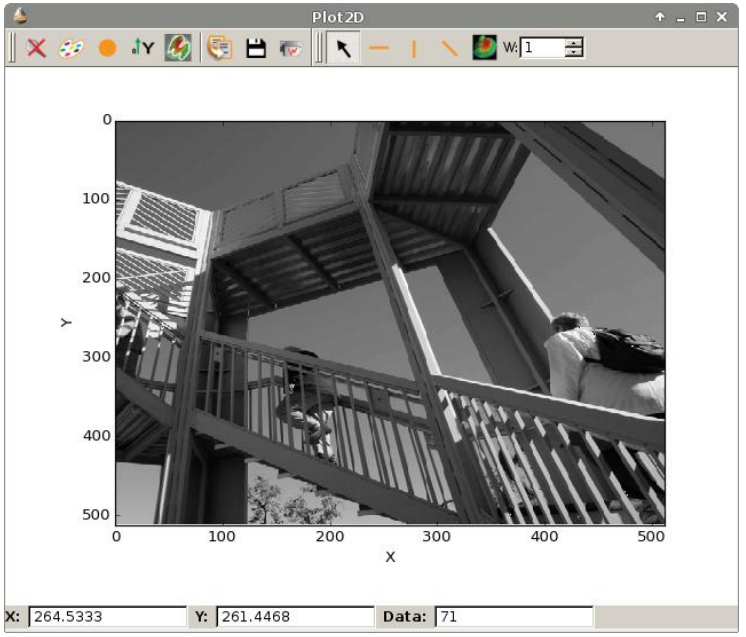
- 1D plotting: ROI, fitting & printing

```
>>> from silx import sx
>>> from numpy import sin, linspace
>>> sx.plot(sin(linspace(-10, 10, 1000)))
```



- 2D display: intensity, mask, profile

```
>>> from scipy.misc import ascent
>>> sx.imshow(ascent())
```



silx.io: input / output

- Built-in support of CSV, SPEC and TIFF
  - Images, SPEC files accessed in the same way as HDF5 files

    Unified widget dealing with ALL supported data formats!!!!!
  - Provide bridges SPEC ←→ HDF5 and octave ←→ HDF5
  - Utilities to save and restore configurations as HDF5, json or ini files
- HDF5 is supported via h5py
- Images (and many detector formats) are supported via FabIO

Silx HDF5 widget example

Name	Type
alltypes_stgs7o.h5	
arrays	
cube	int32
hypercube	int32
image	int32
list	int32
scalar	int32
dtypes	

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	10	11	12	13	14	15
2	20	21	22	23	24	25
3	30	31	32	33	34	35
4	40	41	42	43	44	45
5	50	51	52	53	54	55
6	60	61	62	63	64	65
7	70	71	72	73	74	75
8	80	81	82	83	84	85

Axis selection

Dimension 0 | Dimension 1 col | Dimension 2 row | 0 | limits: 0, 9

HDF5 | Curve | Image | Cube | Raw | Image stack

Create HDF5

Containing all types

Create

Async load

Tree options

Enable sorting

Multi-selection

Drop external file

Reorder files

Header options

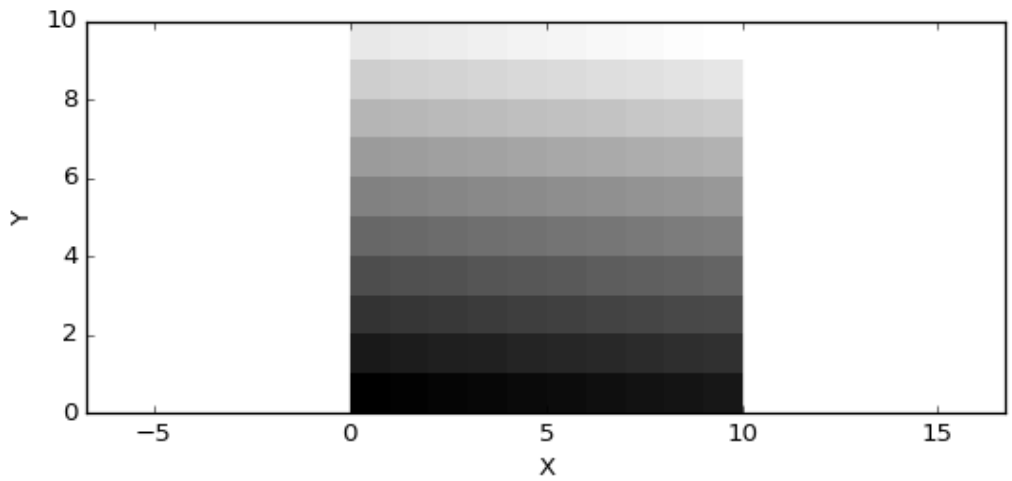
Auto-size headers

Popup to hide/show columns

Default columns

Silx HDF5 widget example

Name	Type
alltypes_stgs7o.h5	
arrays	
cube	int32
hypercube	int32
image	int32
list	int32
scalar	int32
dtypes	



X: 2.606498 **Y:** 9.359807 **Data:** 92

Axis selection

Dimension 0 — limits: 0, 9

Dimension 1 y

Dimension 2 x

HDF5 | Curve | Image | Cube | Raw | Image stack

Create HDF5

Containing all types

Create

Async load

Tree options

- [ ] Enable sorting
- [ ] Multi-selection
- [x] Drop external file
- [x] Reorder files

Header options

- [x] Auto-size headers
- [x] Popup to hide/show columns

Default columns

NXdataViewer

- Data viewer for viewing data in a Nexus NXdata group

- Supports:
  - Scalars, curves, images, scatters, image stack for 3D data
  - Uncertainties, displayed as error bars for 1D data
  - Axes scaling (via @axes)
  - Axes labels (via @long_name)
  - Forcing of predefined views for high dimensionality data (via @interpretation=scalar/spectrum/image)

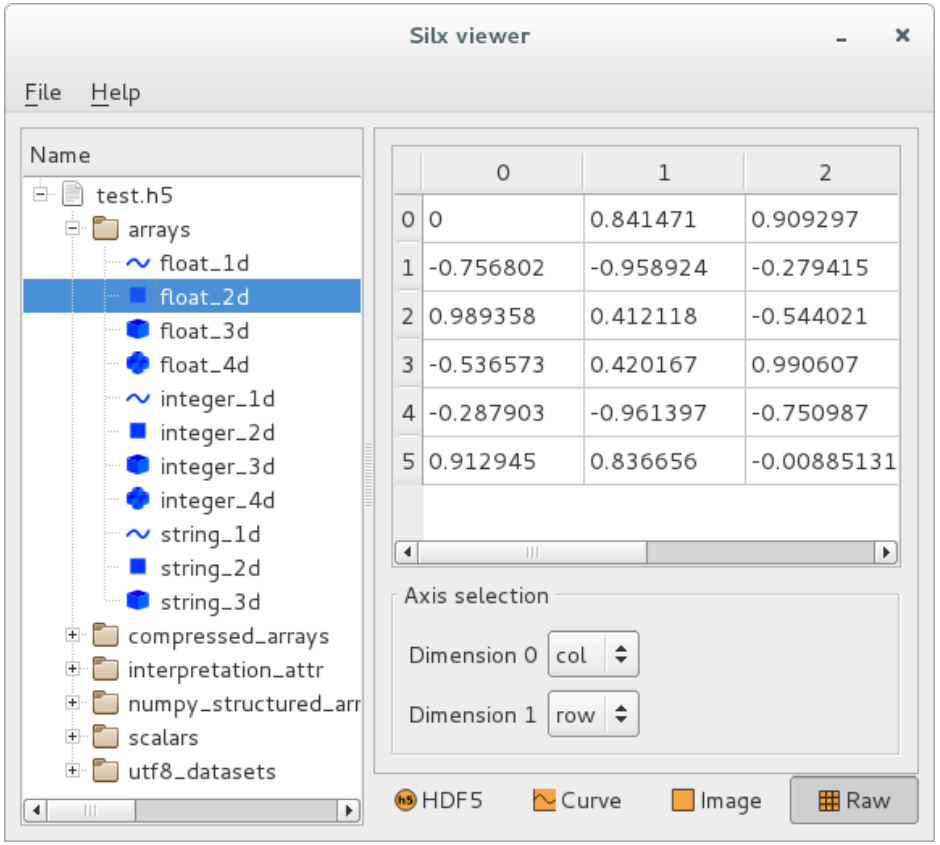
- See examples/hdf5widget.py for a demo
  (Create HDF5 > Containing NXdata groups)

Viewer Application

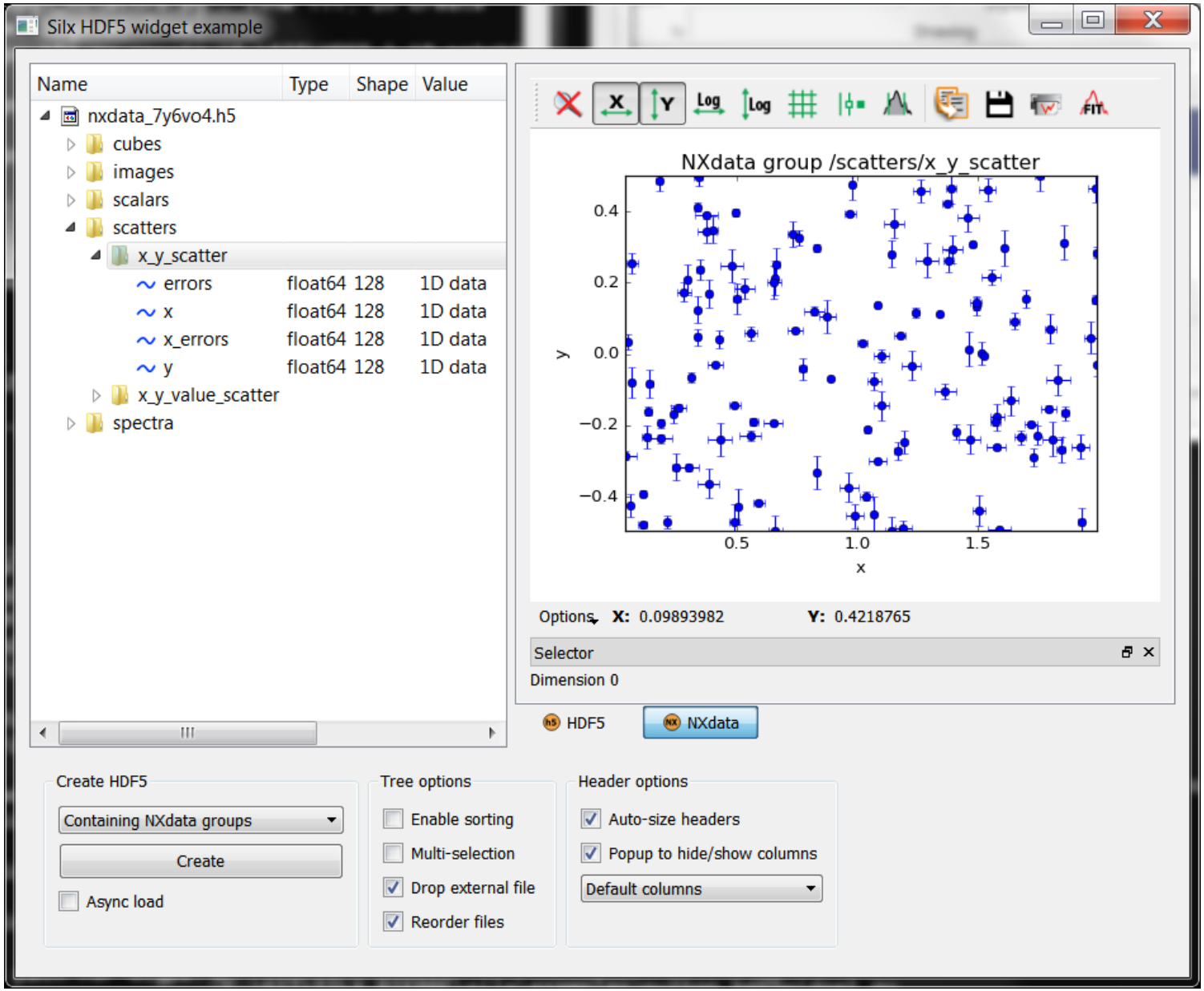
- Browse and display HDF5 files
  (plus any supported file as HDF5)

- File from:
  - *command line / open dialog / drag and drop*

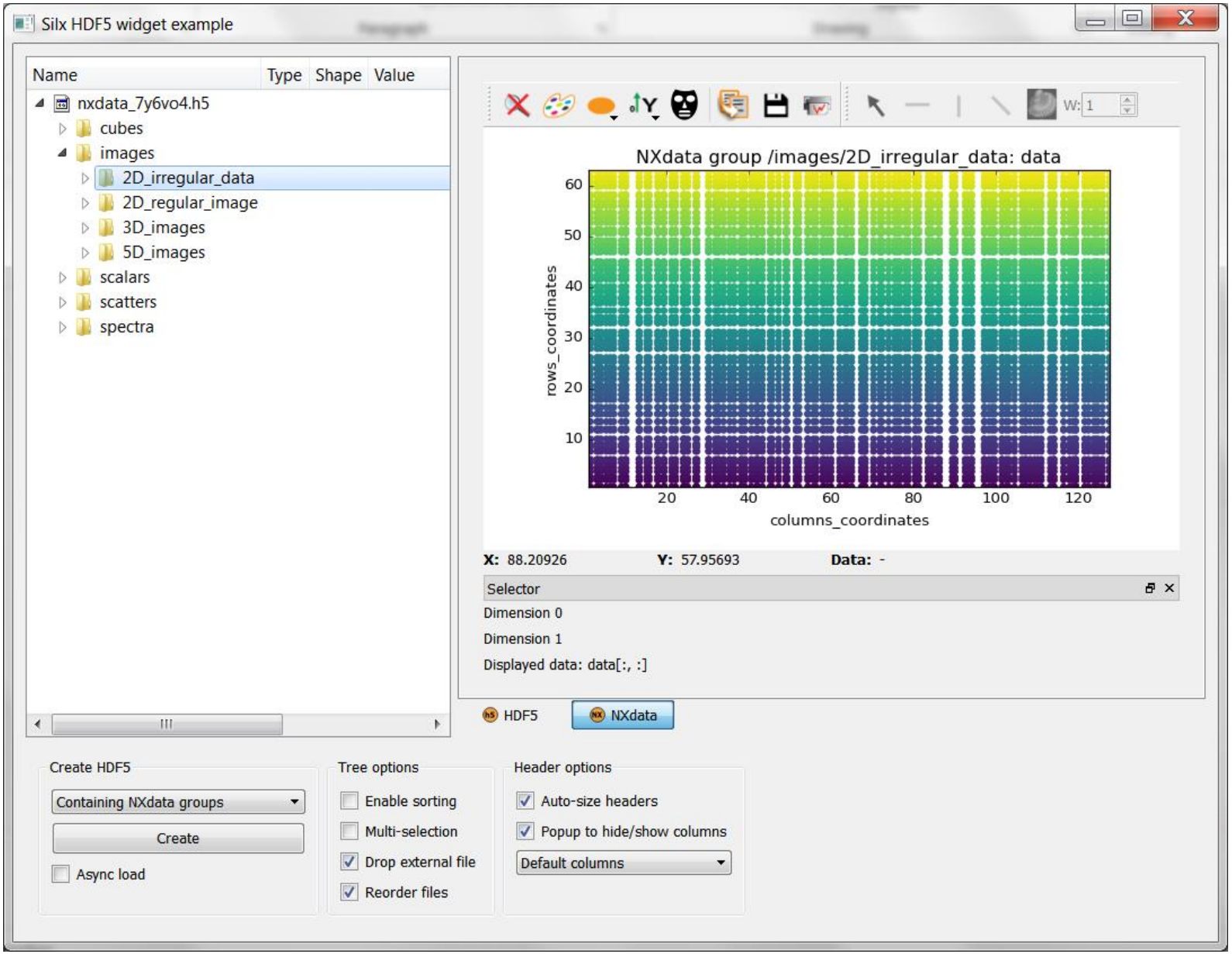
- Commands
  - *silx view <filename>*
  - *python -m silx view*
  - *python3 -m silx view*
  - *./bootstrap.py silx view*



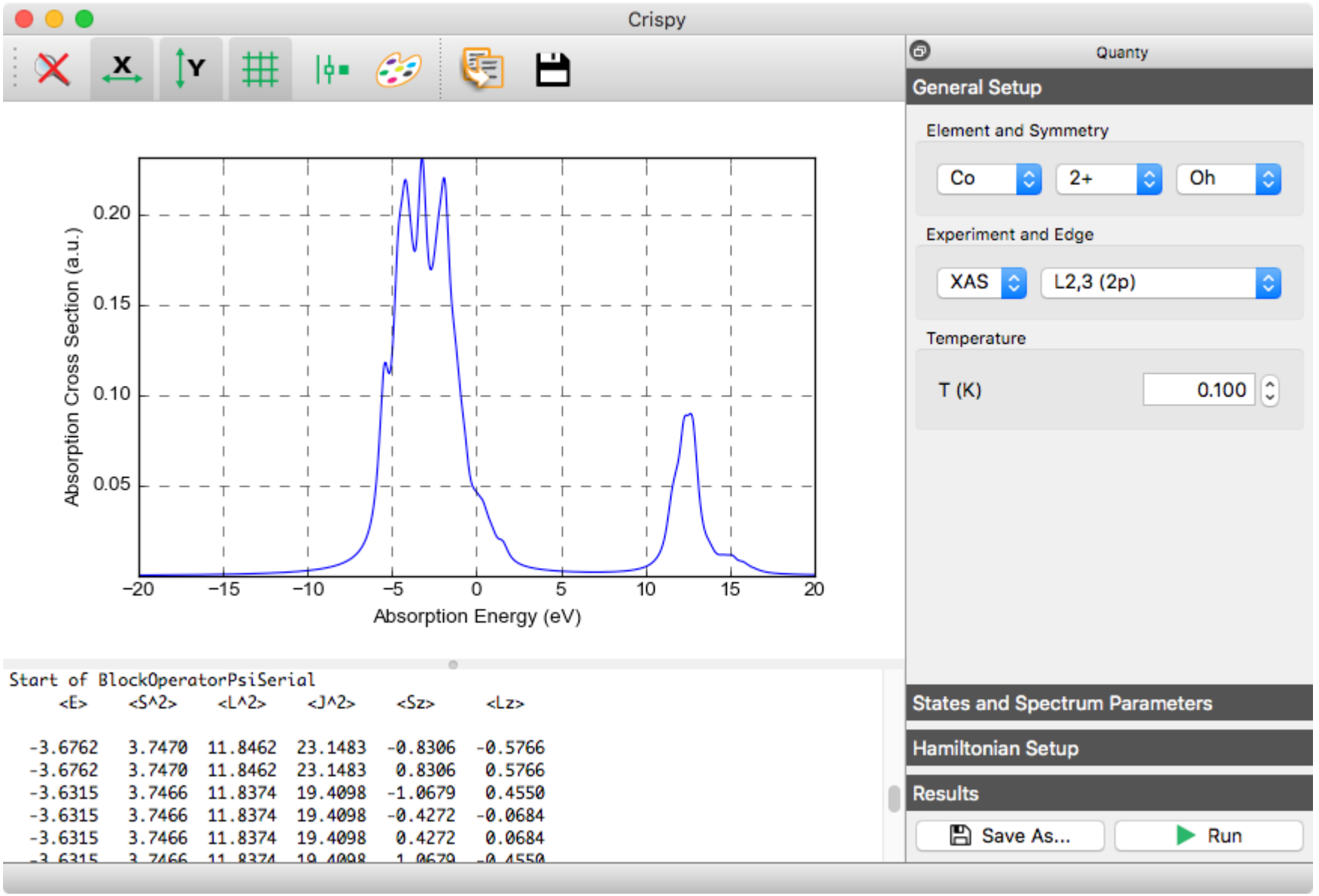
silx view – Generic Viewer Interpreting NXdata Groups



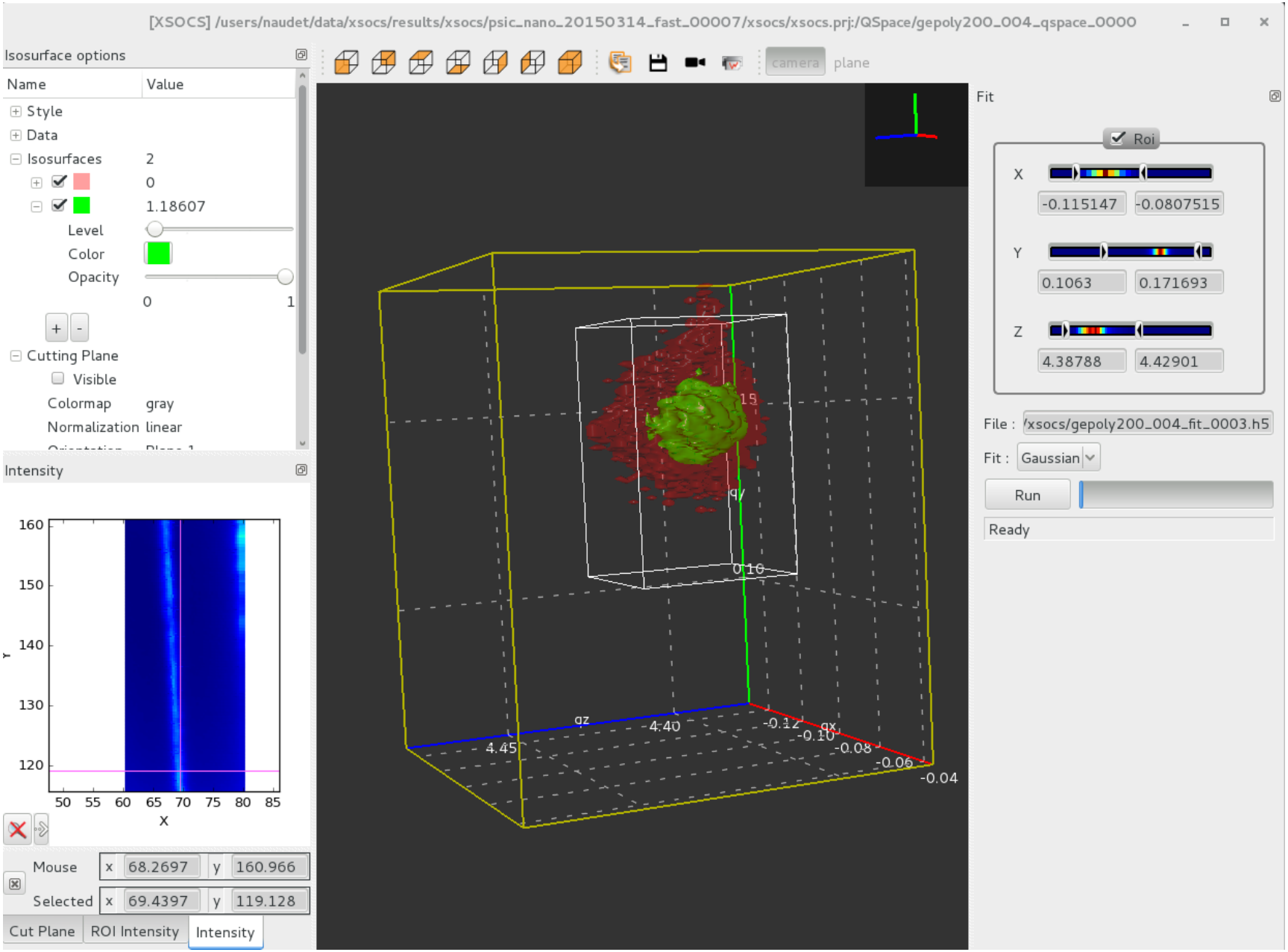
NXdataViewer



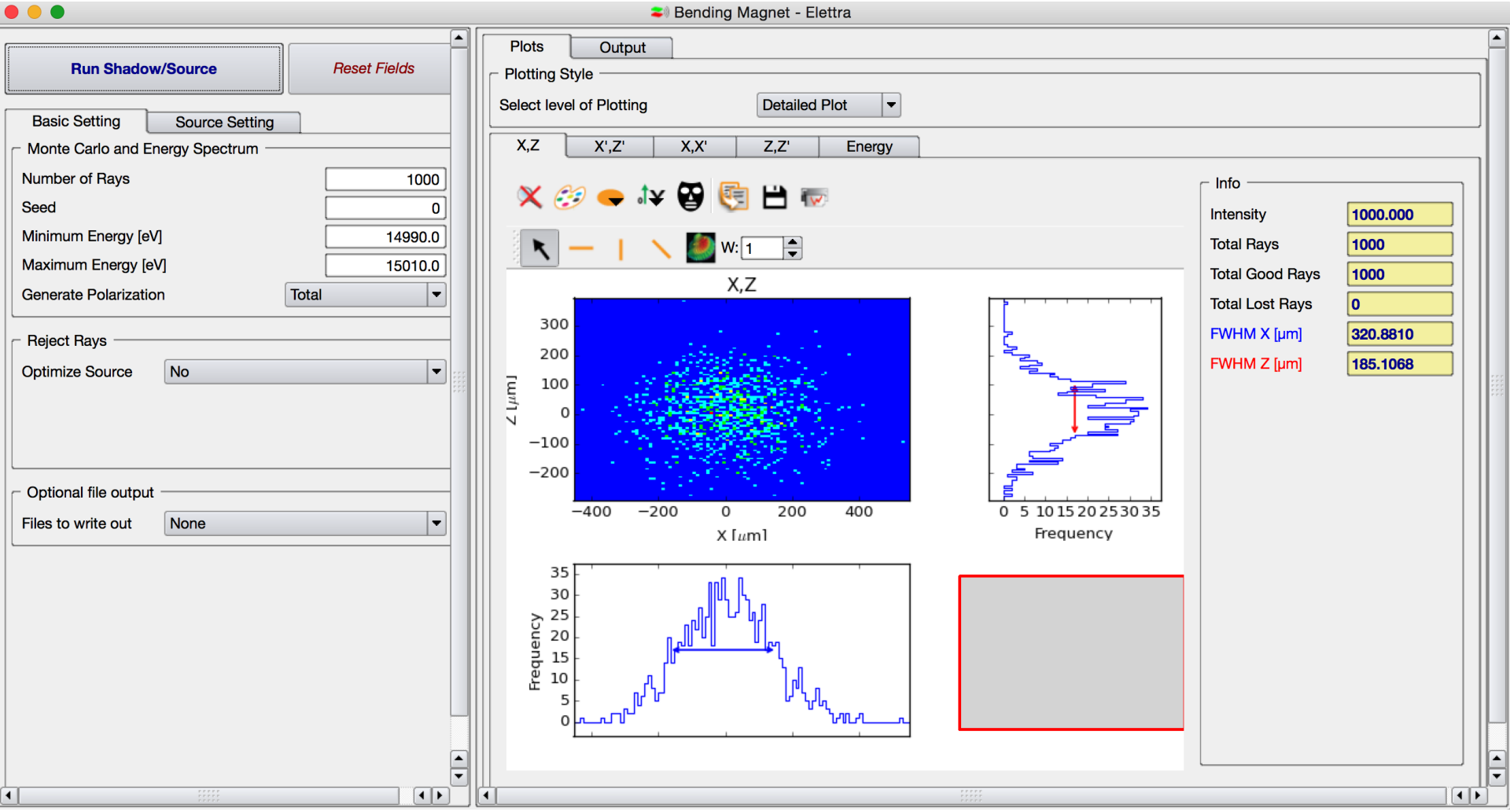
Applications - Crispy



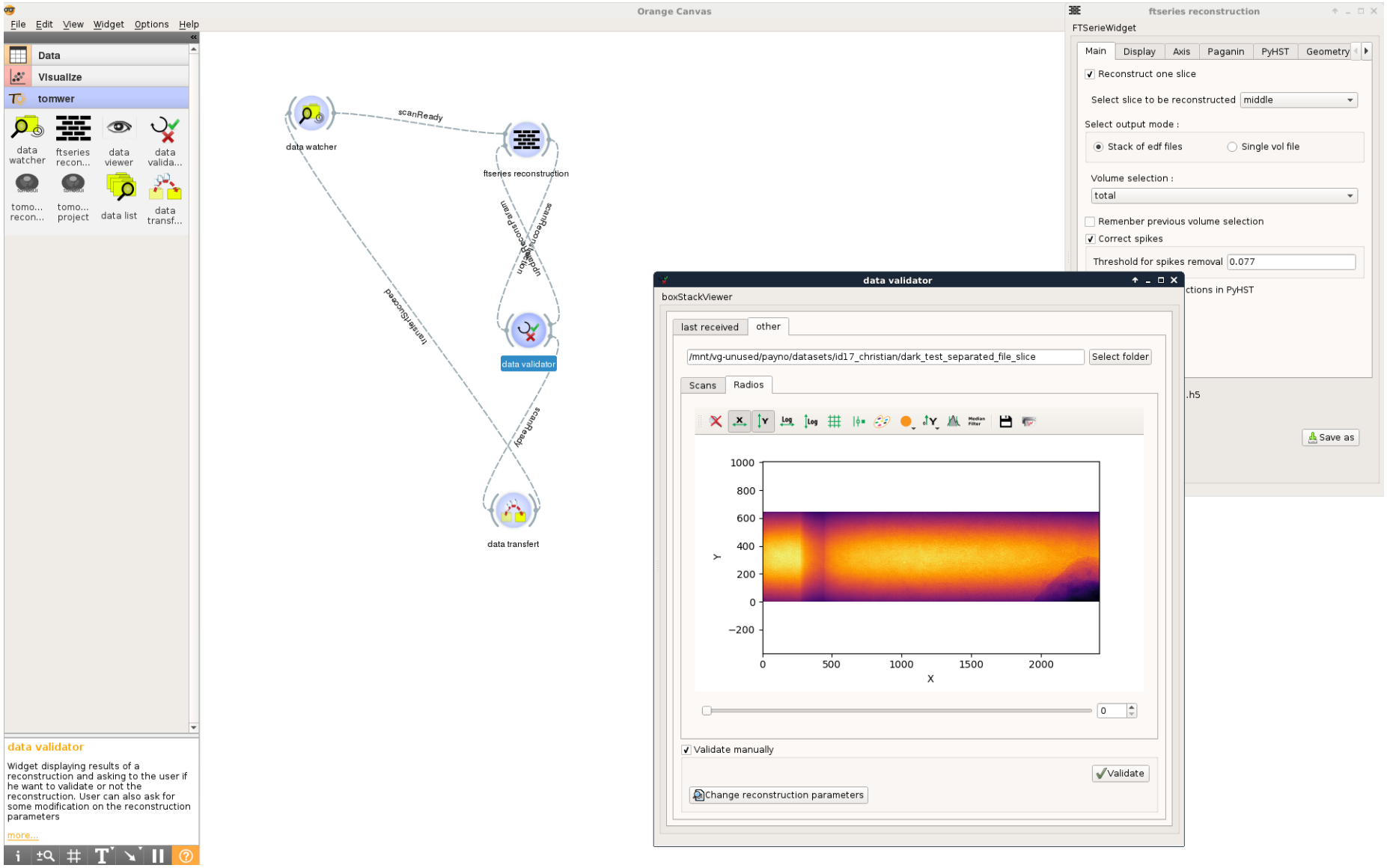
Applications - XSOCS



Applications - OASYS



Applications – Tomography Workflows



Applications – Nanomax@Max IV

NanoMAX Scan Viewer

nanomaxScan_stepscan_week48

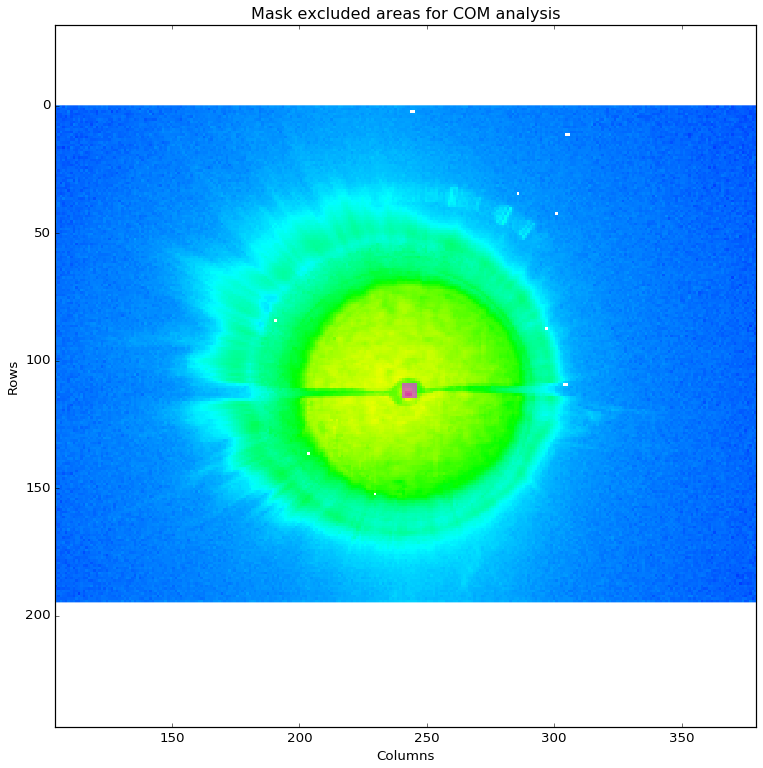
51

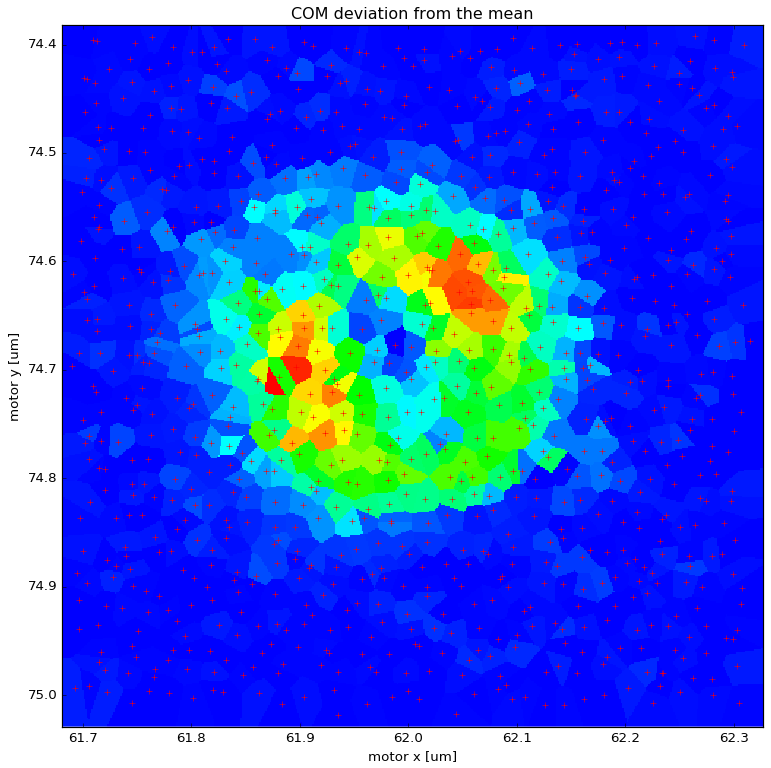
/home/alex/tmp/JW/JWX31C_1.h5

Browse... Load

XRD region of interest | XRD center of mass | XRF region of interest

positions nearest N: 15 COM: magnitude W: 1





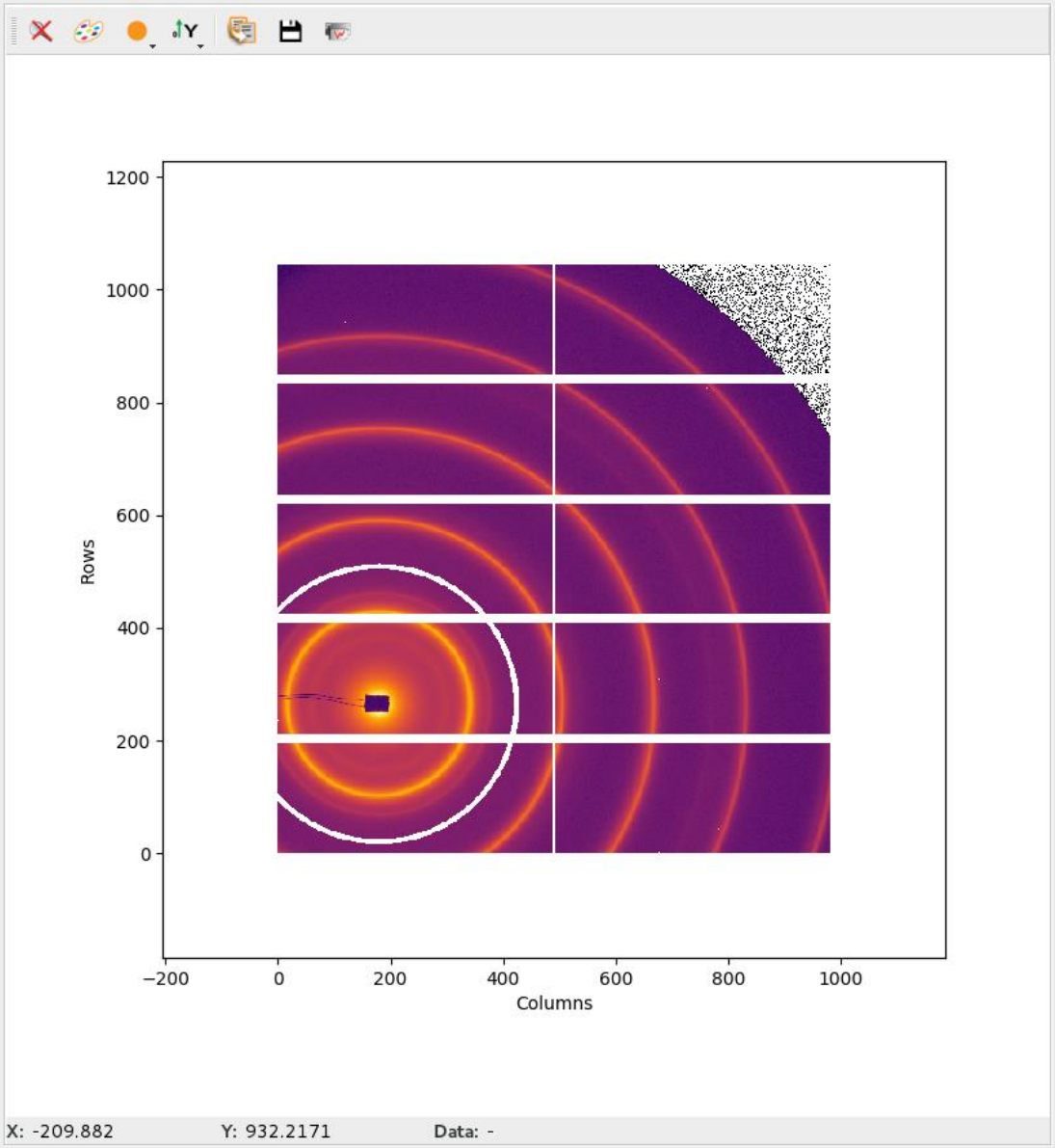
X: 166.8865 Y: 10.93991 Data: 0.1009365

X: 61.70928 Y: 74.50832 Data: 0.1558853

pyFAI Calibration - Settings

PyFAI Calibration

- Experiment settings
- Mask
- Peak picking
- Geometry fitting
- Cake & integration



Experient settings

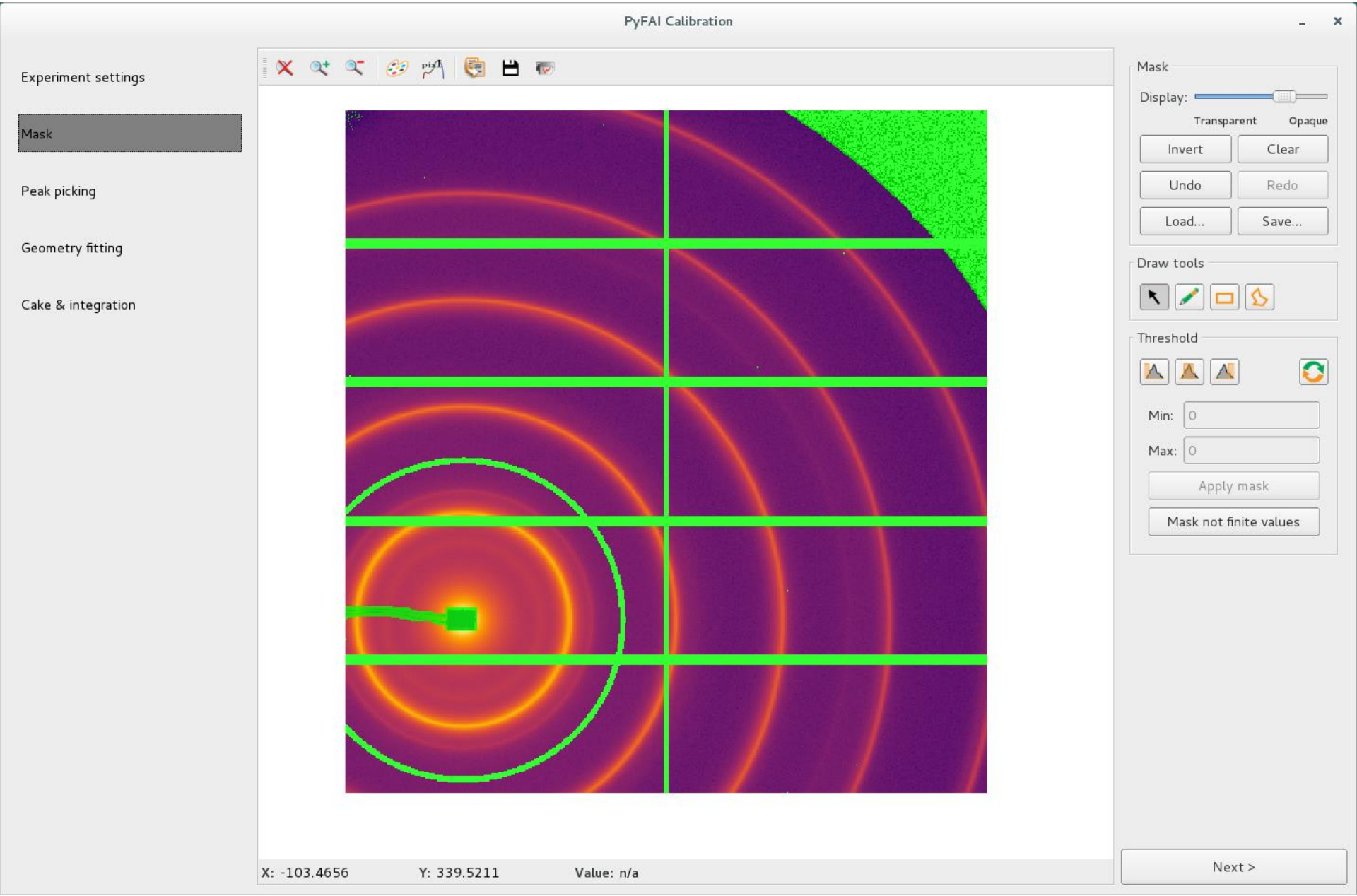
Energy:	12.3978139088	keV
Wavelength:	1.0	Å
Calibrant:	AgBh	
Detector:	pilatus1m	

Acquisition

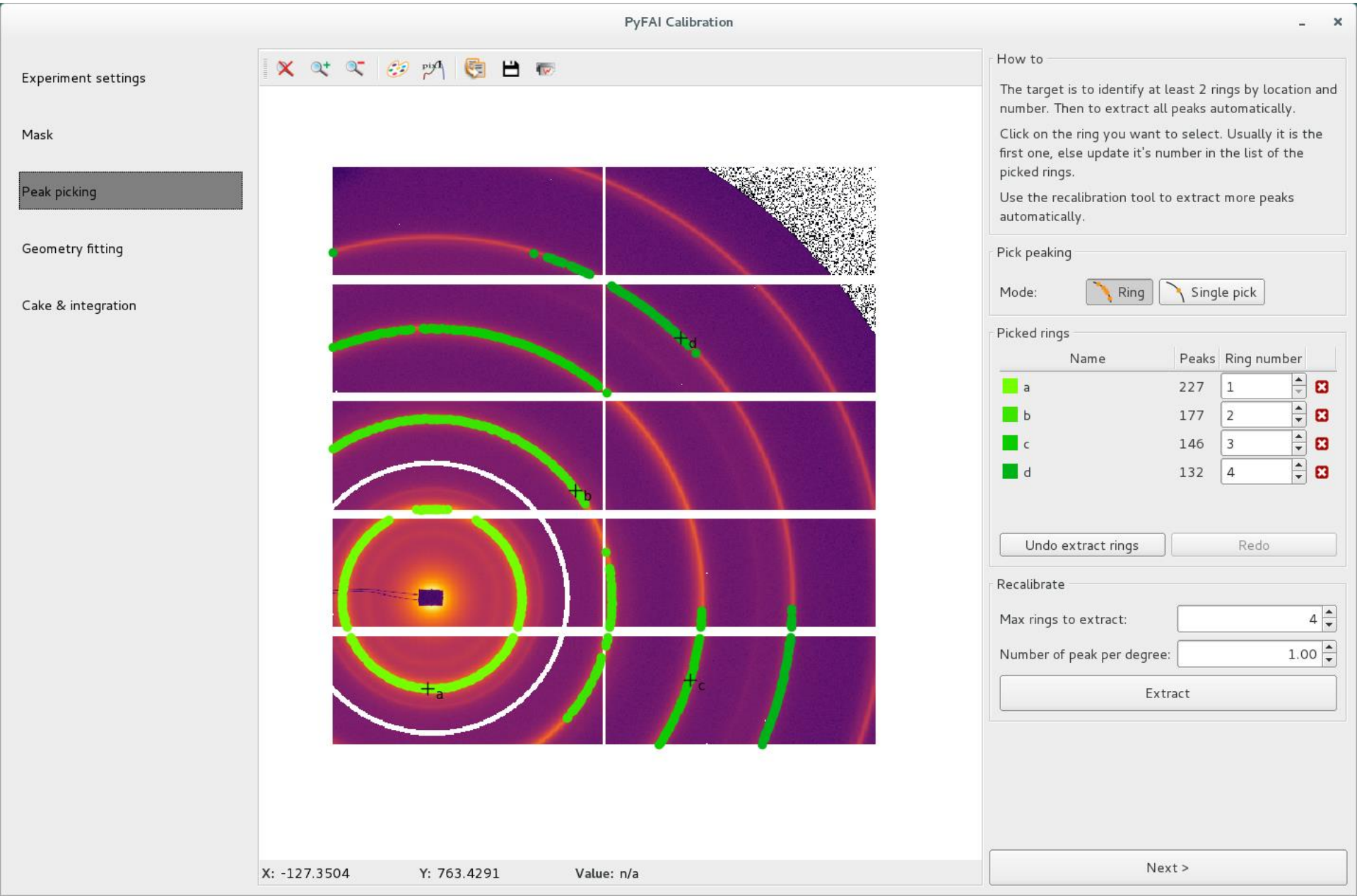
Image file:	alibration/sample-1/image.edf	...
Image size:	1043 × 981	px
Mask file:	alibration/sample-1/mask.msk	...
Dark file:		...

Next >

pyFAI Calibration - Mask



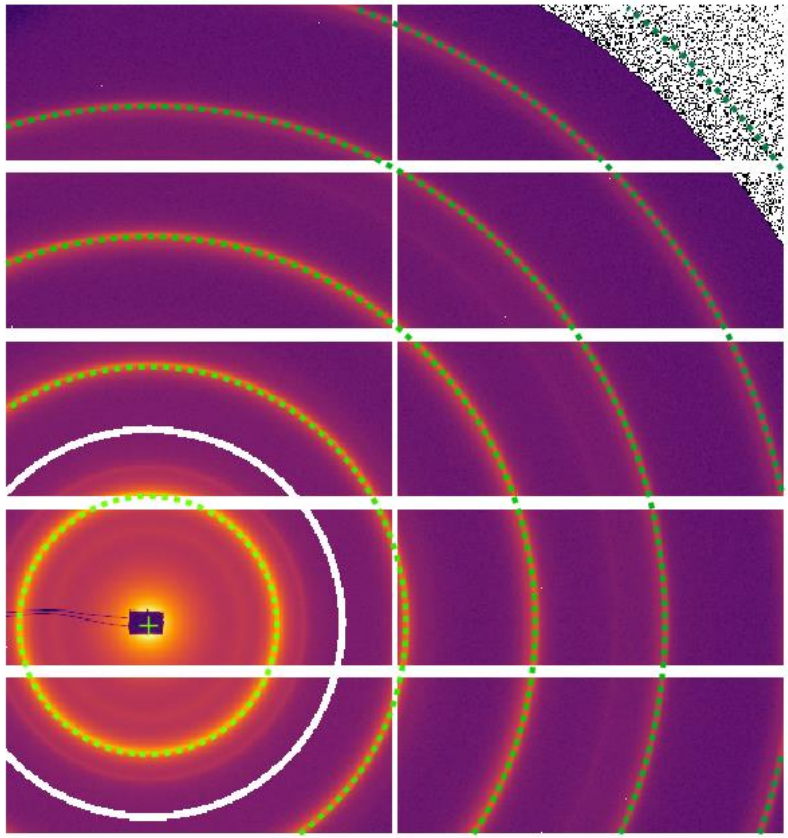
pyFAI Calibration – Peak Picking



pyFAI Calibration – Geometry Fitting

PyFAI Calibration

- Experiment settings
- Mask
- Peak picking
- Geometry fitting
- Cake & integration



X: -124.0285 Y: 890.3549 Value: n/a

How to

The target is to identify all rings of the image.

The algorithm is iterative. It will adjust parameters to improve the fit. You can lock values to avoid modification of them.

You can reset the state to start again from the begining.

If rings are well identified on the image you can check the integration on the next step.

Experiment settings

Wavelength:	1.0	Å

Geometry

Distance:	1.63092014216	m
PONI1:	-0.0357913539286	m
PONI2:	-0.00987467071189	m
Rotation 1:	-0.0250032819825	rad
Rotation 2:	0.0496666955827	rad
Rotation 3:	2.05963351345e-05	rad

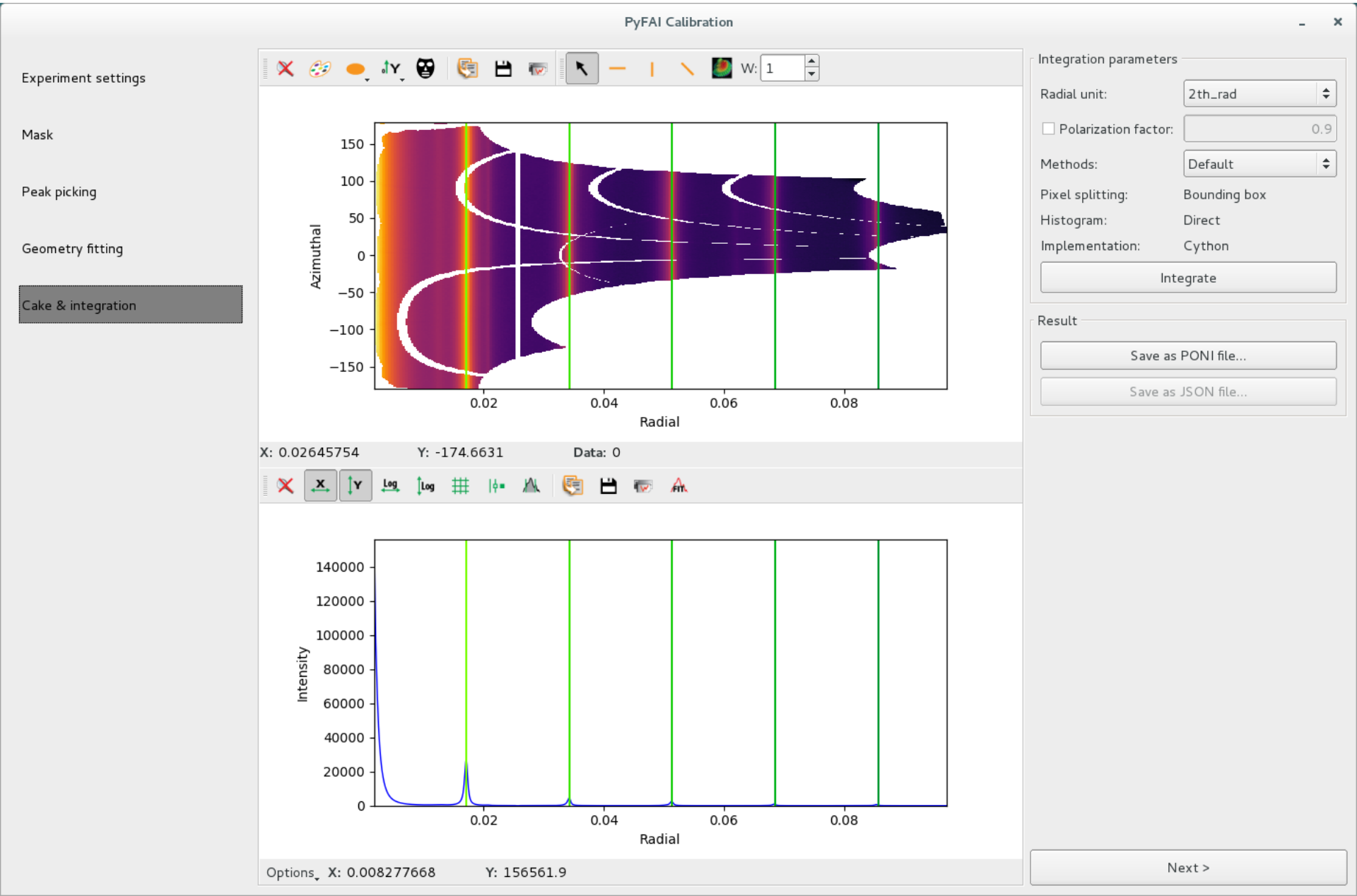
Action

Reset

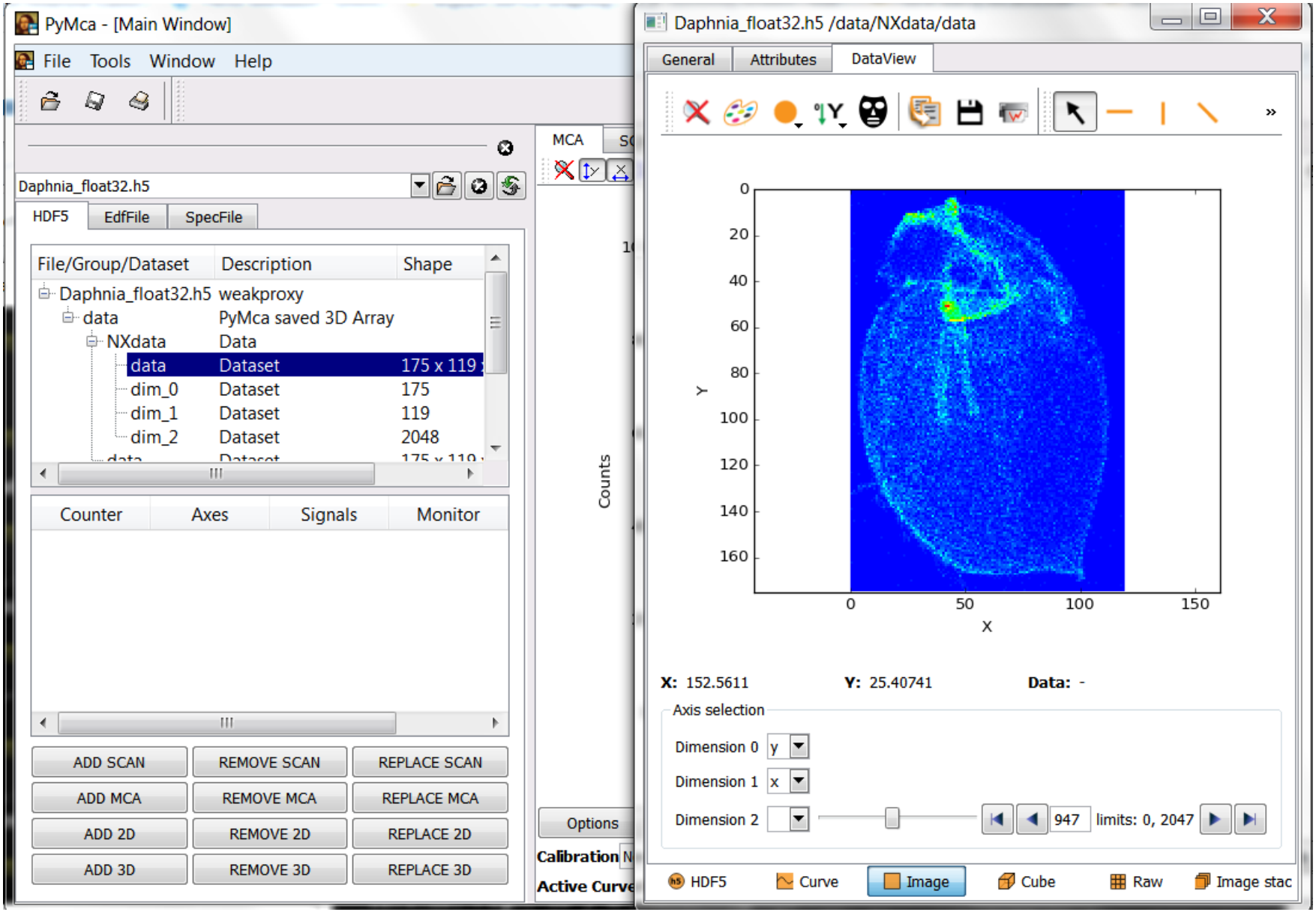
Fit

Next >

pyFAI Calibration – Cake and Integrtion



PyMca - silx DataViewer replacing PyMca TableView



- This release
  - Filtered Back Projection in OpenCL
  - Print Preview
  - Plot Context Menu
  - silx convert
- Late 2017
  - 3D SceneGraph
  - pyFAI Calibration GUI
  - PyMca using silx Plot
- 2018
  - pyFAI release with pyFAI GUI
  - PyMca using silx 3D graphics
- Let the library grow according to the needs of applications

ROLE OF NON-CORE DEVELOPERS

- Identify something you are interested on
- Try to achieve it
- Wow! I can do what I want, what next?
  - Start again
  - Make suggestions
  - Contribute with a demo/recipe
- I cannot do it
  - Ask help

ROLE OF CORE DEVELOPERS

- Help non-core developers
- Create issues
  - Bugs
  - Documentation
  - Desired features
- Fix issues
  - Bugs
  - Documentation
  - Unlikely for new features
- Review pull requests

HANDS ON!

- Try to start with a single entry point www.silx.org
  - You should be able to install 0.5.0 version
- For this code camp we'll use 0.6.0a, you can either:
  - clone the repository (and use your compilation chain)
  - install a nightly built package (debian)
  - use a pre-built binary wheel:
    - http://www.silx.org/pub/wheelhouse/