8th Silx Code Camp
October 11, 2018

This talk

- Introduction
  - Novelties (version 0.9.0)
- Status of silx (version 0.8.0)
- Goals of the code camp
  - For users
  - For core developers
- Hands on!

silx.math: Median Filter updates

- `silx.math.medfilt[1d|2d]`
  - Added support of Not-A-Number
  - Added `'constant'` padding mode
  - Optimized performance

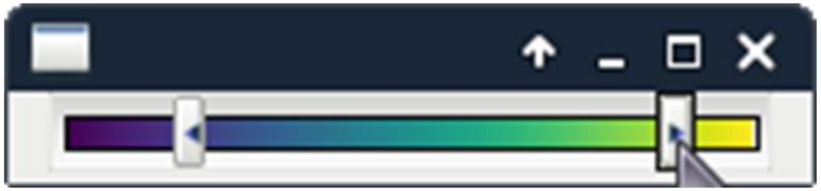
silx.gui: Qt / numpy image conversion

`silx.gui.utils:`

- `convertArrayToQImage(array)`
- `convertQImageToArray(image)`

silx.gui.widgets.RangeSlider

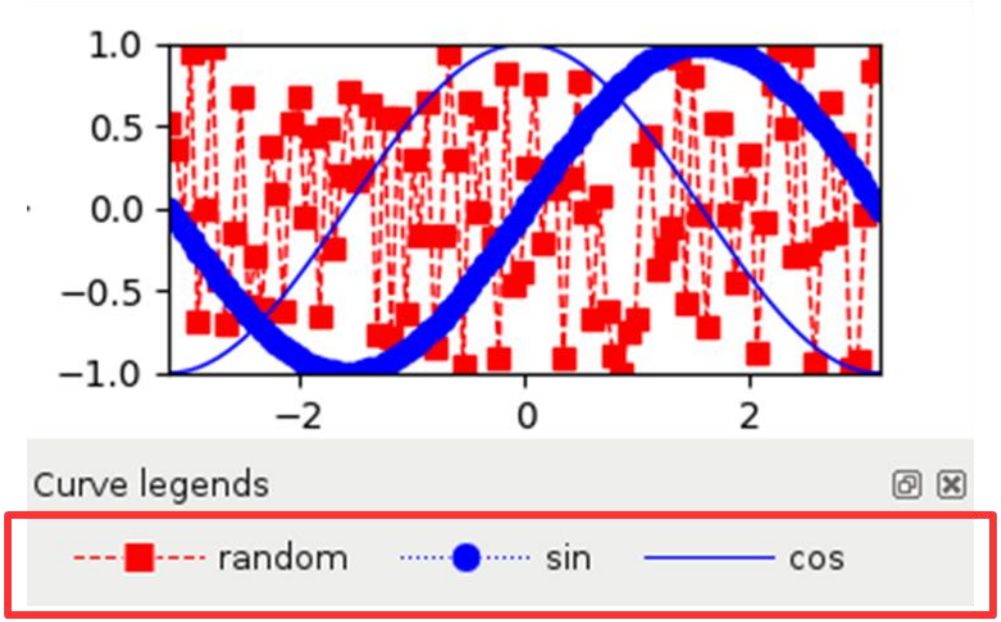
`silx.gui.widgets.RangeSlider:`



- 2 sliders defining a range with settable color-mapped background.
- Initial version developed by Damien Naudet in XSocs application.

silx.gui.plot: CurveLegnedWidget

silx.gui.plot.CurveLegendWidget:



- Display legends of curves in a plot
- Compact alternative to `LegendSelector`.

silx.gui.plot: PlotWidget API

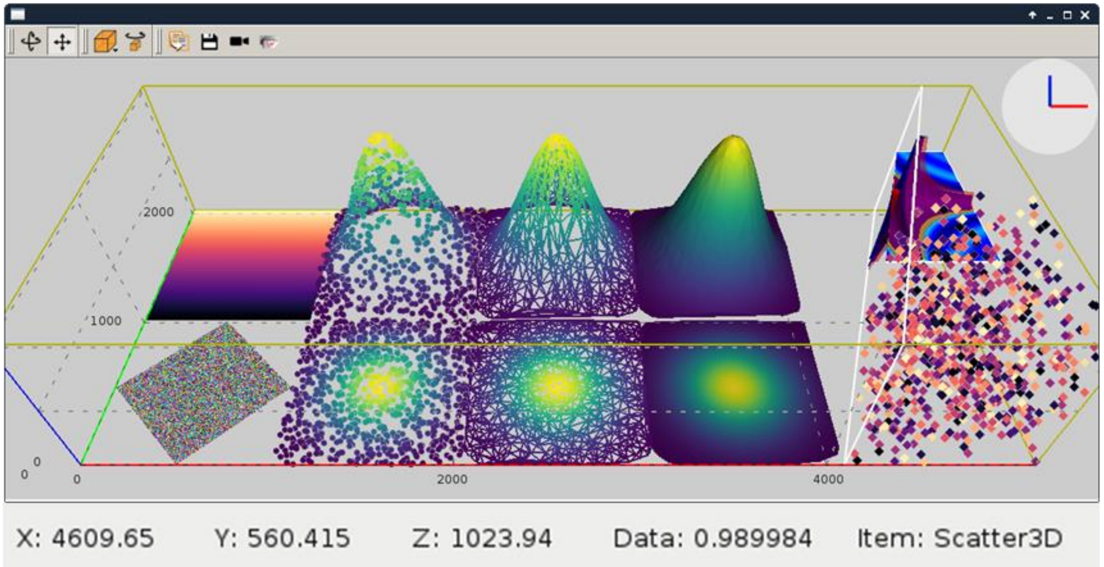
`silx.gui.plot.PlotWidget:`

- Add support of line style and width to line markers
- Get all items in the plot: `getItems`.
- Follow plot content update: `sigItemAdded` and `sigItemAboutToBeRemoved`.

silx.gui.plot3d: PositionInfoWidget

`silx.gui.plot3d.tools.PositionInfoWidget:`

- Widget displaying data at mouse position on double-click.



X: 4609.65 Y: 560.415 Z: 1023.94 Data: 0.989984 Item: Scatter3D

silx.gui.plot3d: Picking

`silx.gui.plot3d.SceneWidget:` Add picking of 3D items at a position on the screen:

```
    pickItems(x, y, condition=None)
```

Implementation choices:

- CPU-based ray-casting
- No preprocessing (e.g., space partitioning)
- Pure Python/numpy implementation

Dependencies

- Qt bindings:
  - **`PyQt5`**: Main target (GPL)
  - **`PySide2`**: Added “beta” support (LGPL)
  - **`PyQt4`**: Deprecated
  - **`PySide`**: Deprecated (to be dropped in v0.10)
- New required dependencies:
  - **`h5py`**
  - **`fabio`**

Installation

- Easier installation of all dependencies:

```
pip install silx[full]
```

- Windows standalone application

This talk

- Introduction
  - Novelties (version 0.9.0)
- Status of silx (version 0.8.0)
- Goals of the code camp
  - For users
  - For core developers
- Hands on!

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 (FBP, registration, median filter, …)
- third-party: External utilities
- utils: Internal utilities
- sx: Convenience module for interactive use

silx.resources

Container of icons, opencl programs, …

Provisions for simplifying handling of frozen binaries

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")
```

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 and Functional APIs

- 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 and Functional APIs

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.))
```

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.

silx.gui.plot API

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

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

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)
```

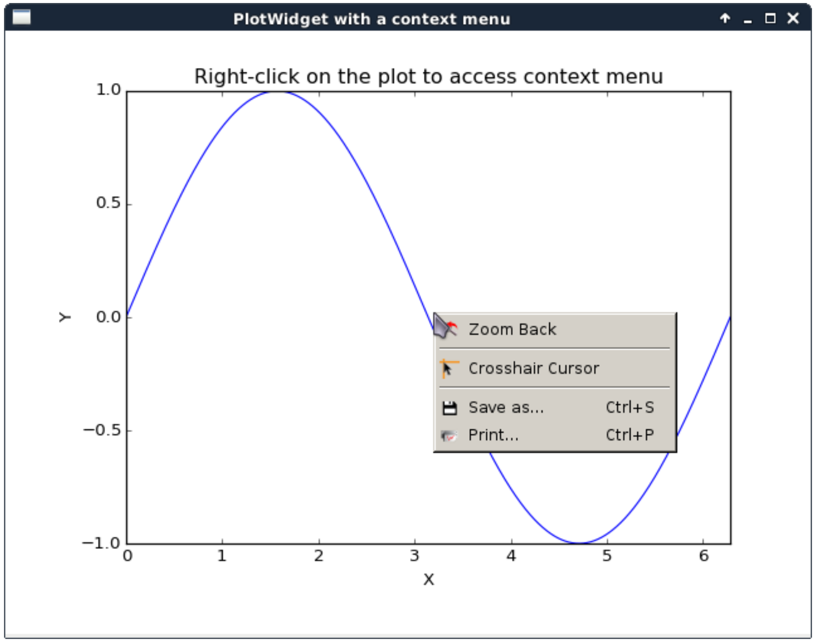
  - A demo is available at *examples/plotLimits.py*

- Helper to synchronize axes

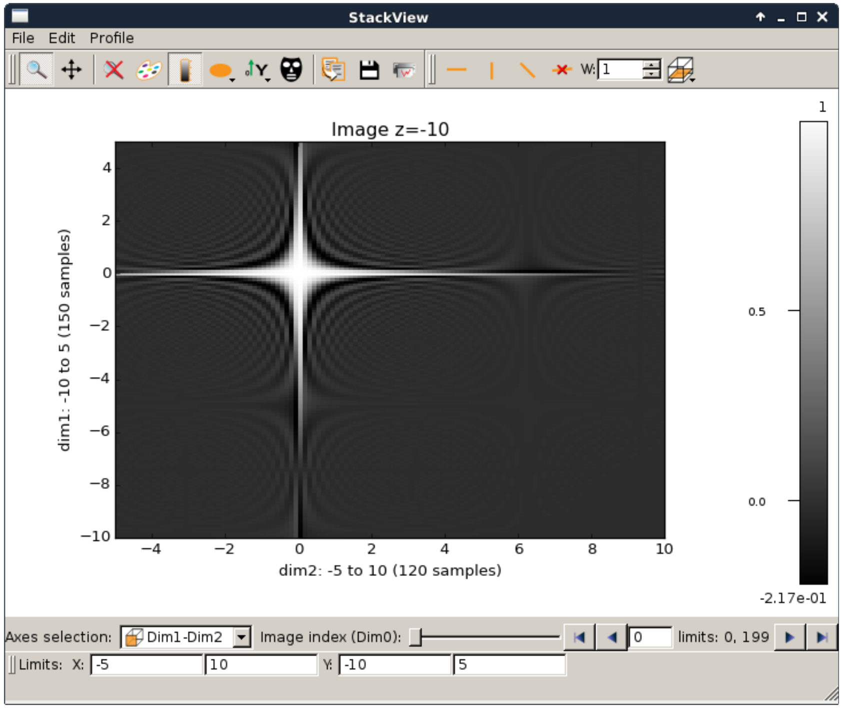
```
from silx.gui.plot.utils.axis import SyncAxes
sync = SyncAxes([plot1.getXAxis(),
                 plot2.getXAxis(),
                 plot3.getXAxis()])
```

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

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



- PlotWindow, Plot2D
  - Add colorbar



Axes selection: Dim1-Dim2 Image index (Dim0): 0 limits: 0, 199

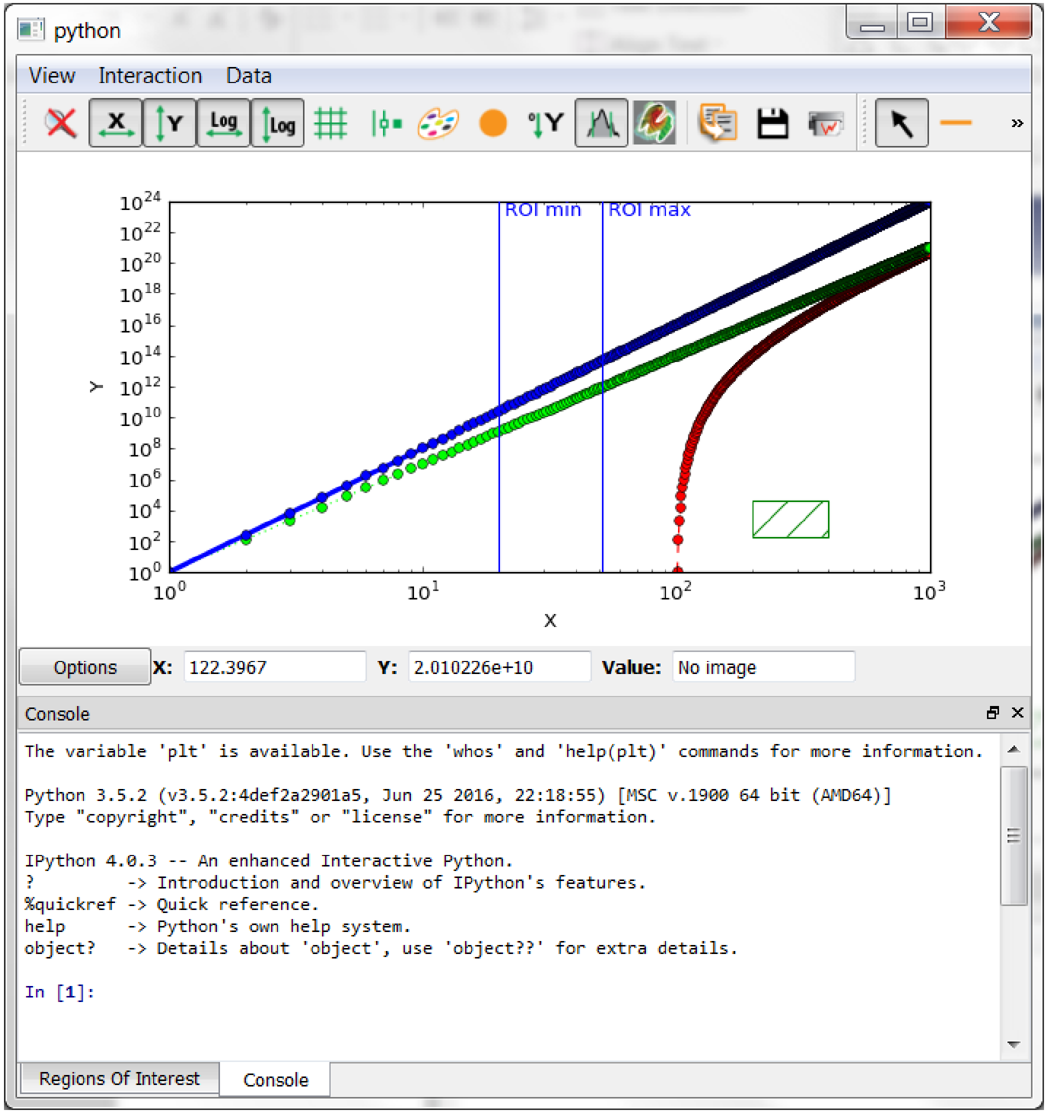
Limits: X: -5 10 Y: -10 5

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

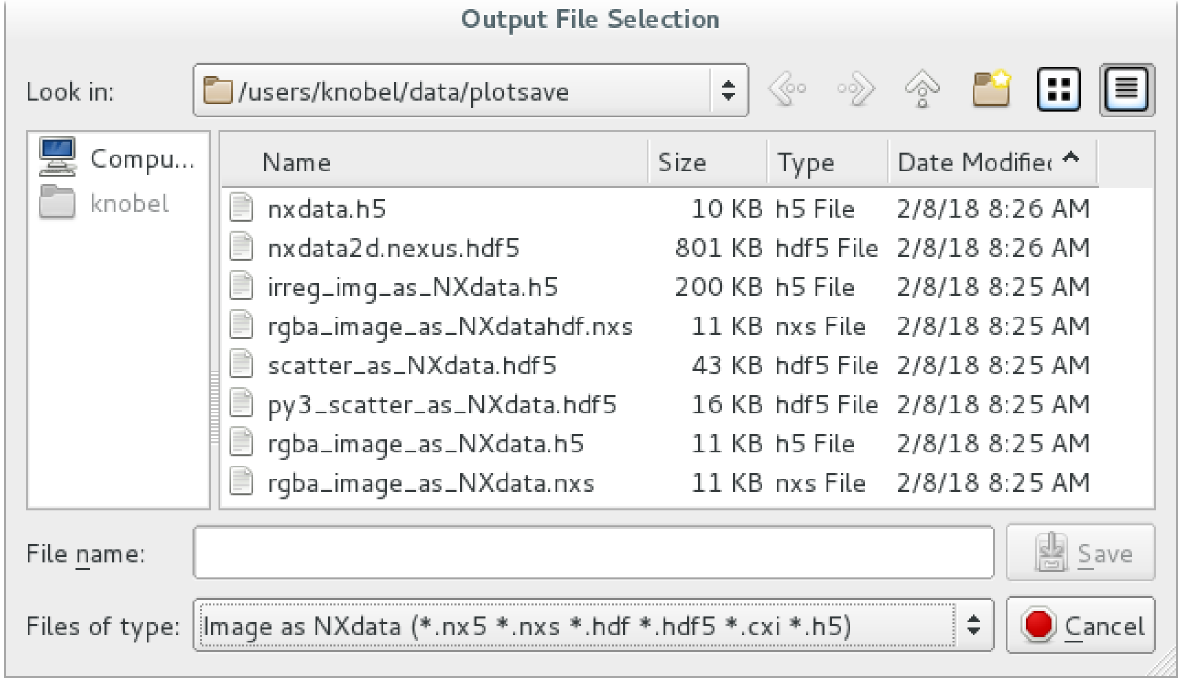


X: 122.3967 Y: 2.010226e+10 Value: No image

```
The variable 'plt' is available. Use the 'whos' and 'help(plt)' commands for more information.

Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.0.3 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Plot SaveAction : add save as NXdata

- Save active curve, active scatter or active image to *NXdata*



Output File Selection

Look in: /users/knobel/data/plotsave

Name	Size	Type	Date Modifie
nxdata.h5	10 KB	h5 File	2/8/18 8:26 AM
nxdata2d.nexus.hdf5	801 KB	hdf5 File	2/8/18 8:26 AM
irreg_img_as_NXdata.h5	200 KB	h5 File	2/8/18 8:25 AM
rgba_image_as_NXdatahdf.nxs	11 KB	nxs File	2/8/18 8:25 AM
scatter_as_NXdata.hdf5	43 KB	hdf5 File	2/8/18 8:25 AM
py3_scatter_as_NXdata.hdf5	16 KB	hdf5 File	2/8/18 8:25 AM
rgba_image_as_NXdata.h5	11 KB	h5 File	2/8/18 8:25 AM
rgba_image_as_NXdata.nxs	11 KB	nxs File	2/8/18 8:25 AM

File name:

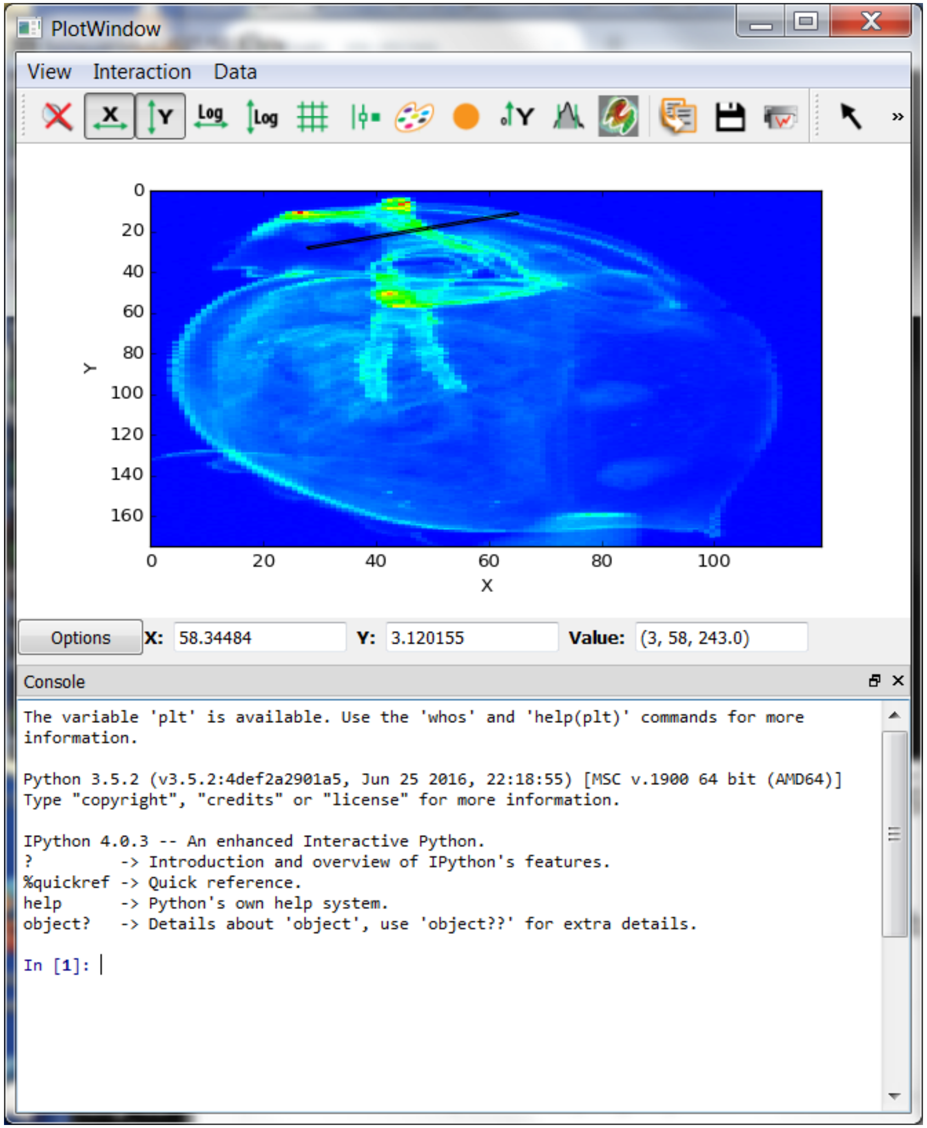
Files of type: Image as NXdata (*.nx5 *.nxs *.hdf *.hdf5 *.cxi *.h5)

- Can save some parts of plot state (title, axis labels, active data...) but not all (no curve style, colormap info, additional data items...)
- Future improvements: add a dialog to specify output group in an existing HDF5 file

silx.gui: Plot 2D

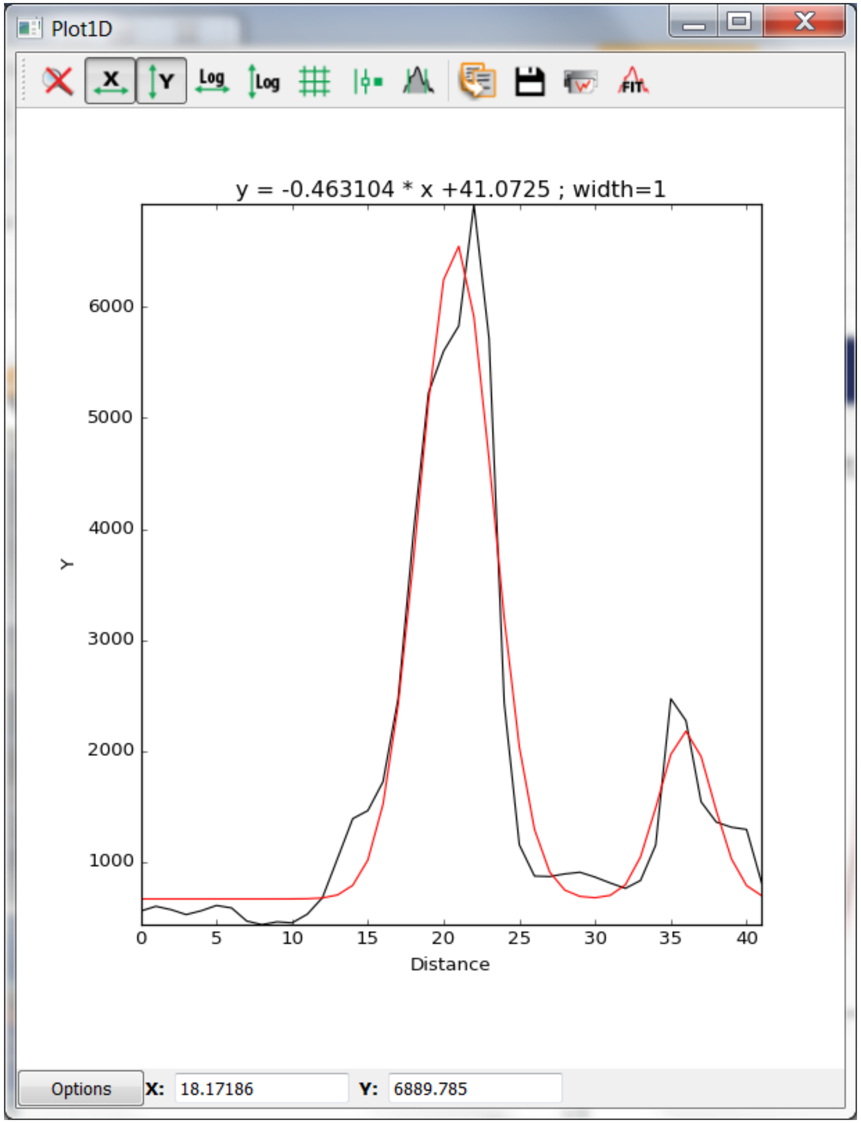
- Visualize 2D data (Images and Stacks of Images)
  - Support Median Filters, Profiles and Masks on them
- Visualize 3D data as scatter plots
  - Support Masks on them
- Apply different colormaps
- Plot an image with associated histograms
- Visualize 3D scalar fields (Isosurfaces)

Full-featured widgets



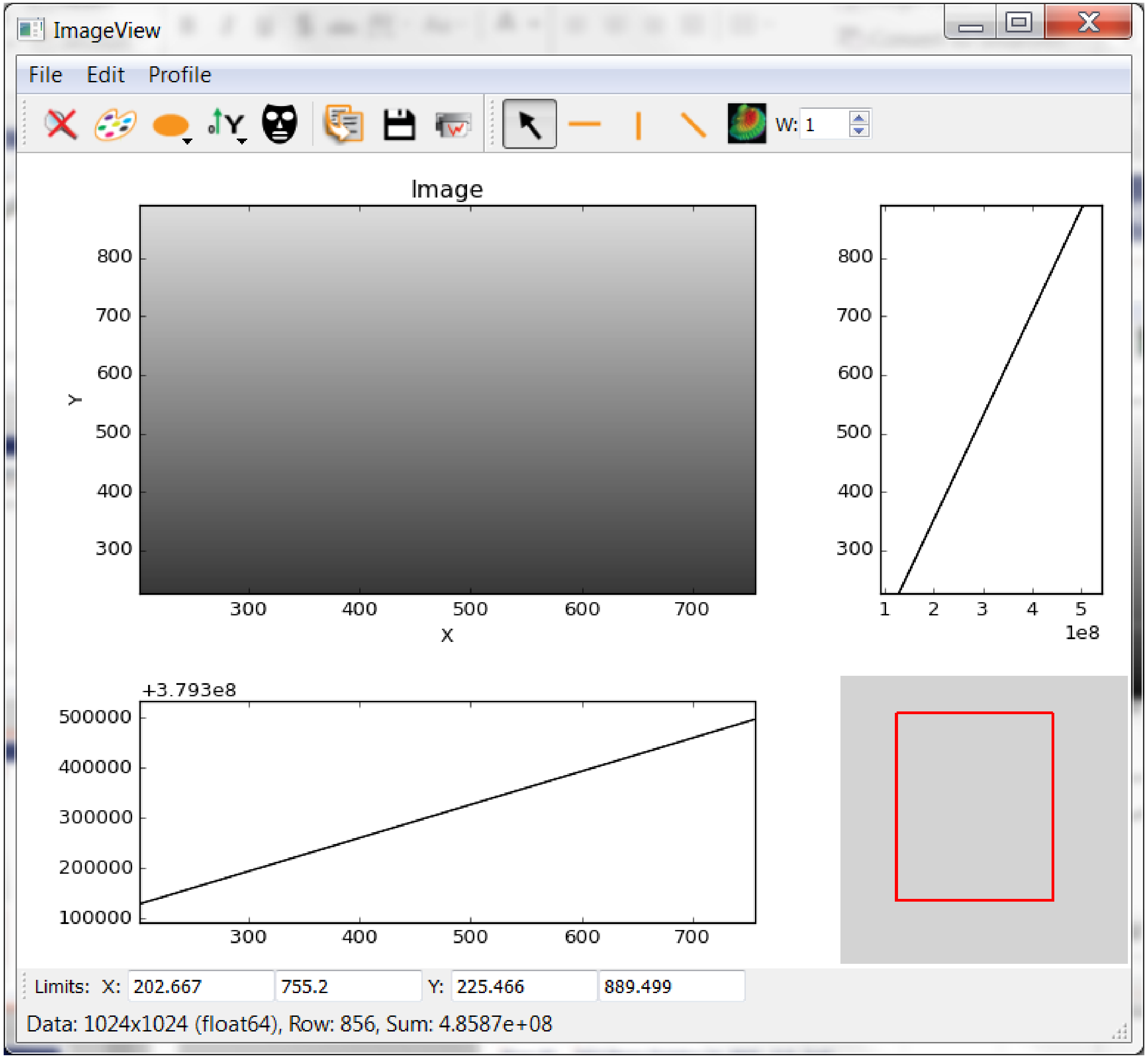
X: 58.34484 Y: 3.120155 Value: (3, 58, 243.0)

```
The variable 'plt' is available. Use the 'whos' and 'help(plt)' commands for more
information.

Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.0.3 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: |
```



X: 18.17186 Y: 6889.785

Full-featured Widgets

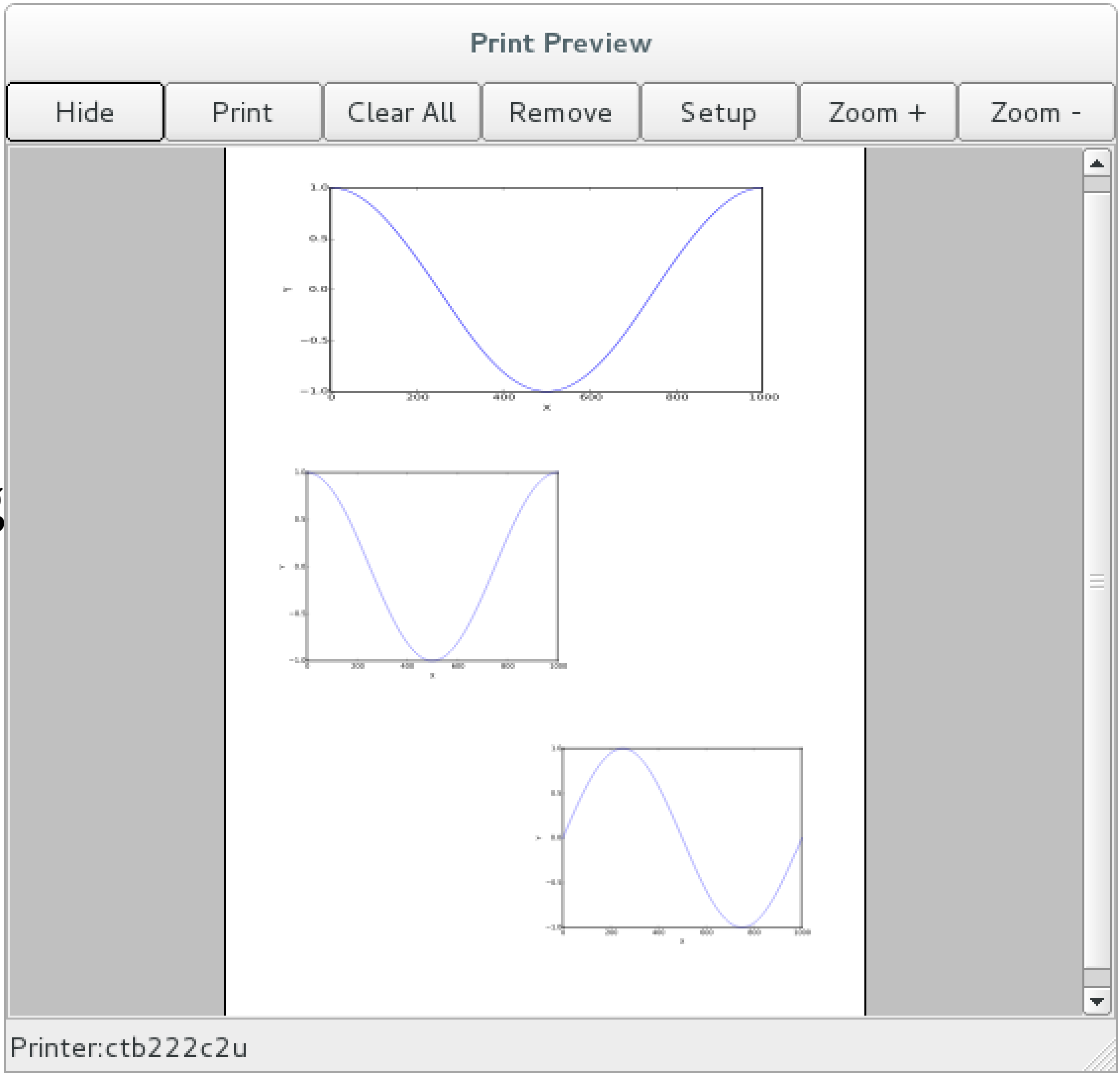


Limits: X: 202.667 755.2 Y: 225.466 889.499

Data: 1024x1024 (float64), Row: 856, Sum: 4.8587e+08

Print Preview

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

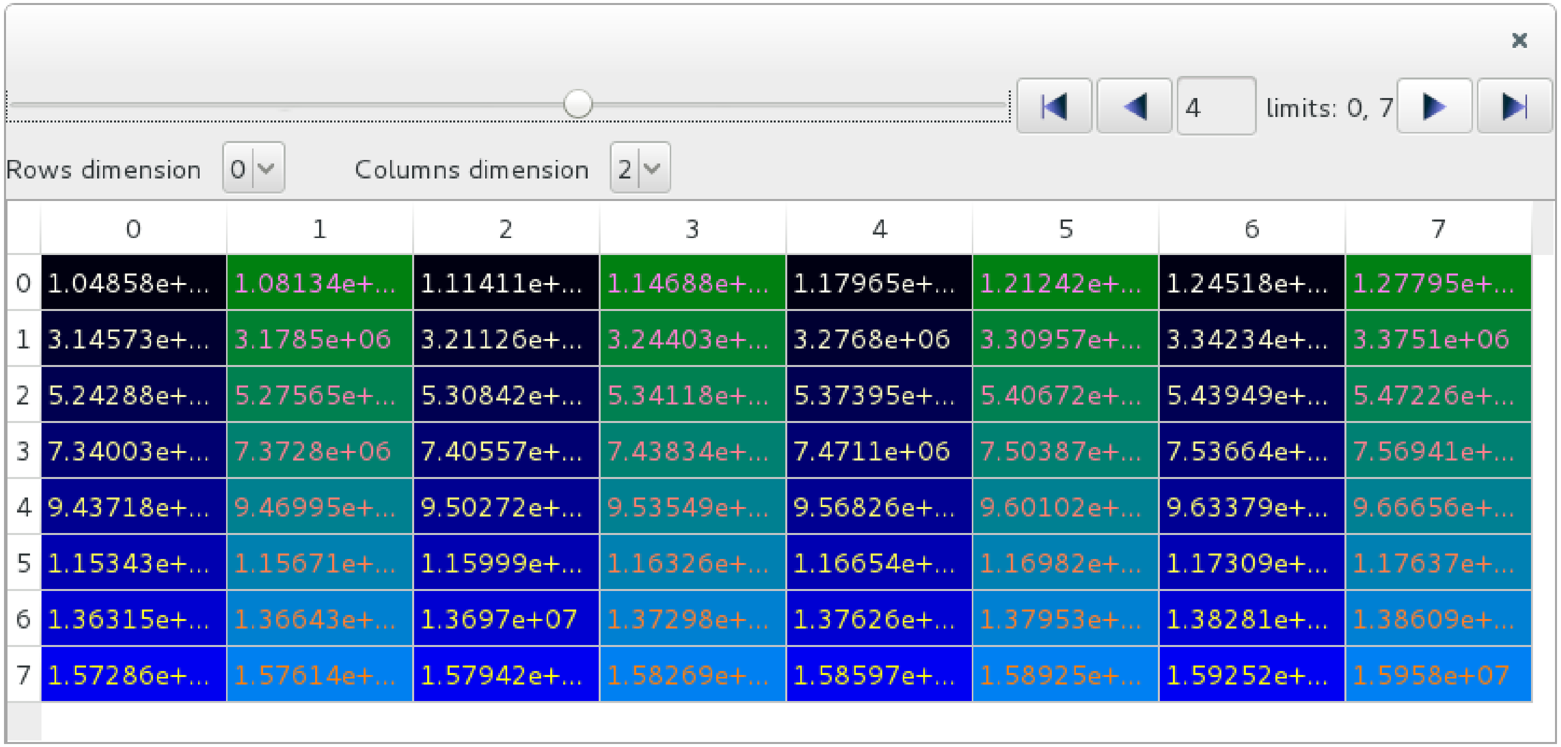


Printer:ctb222c2u

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



4 limits: 0, 7

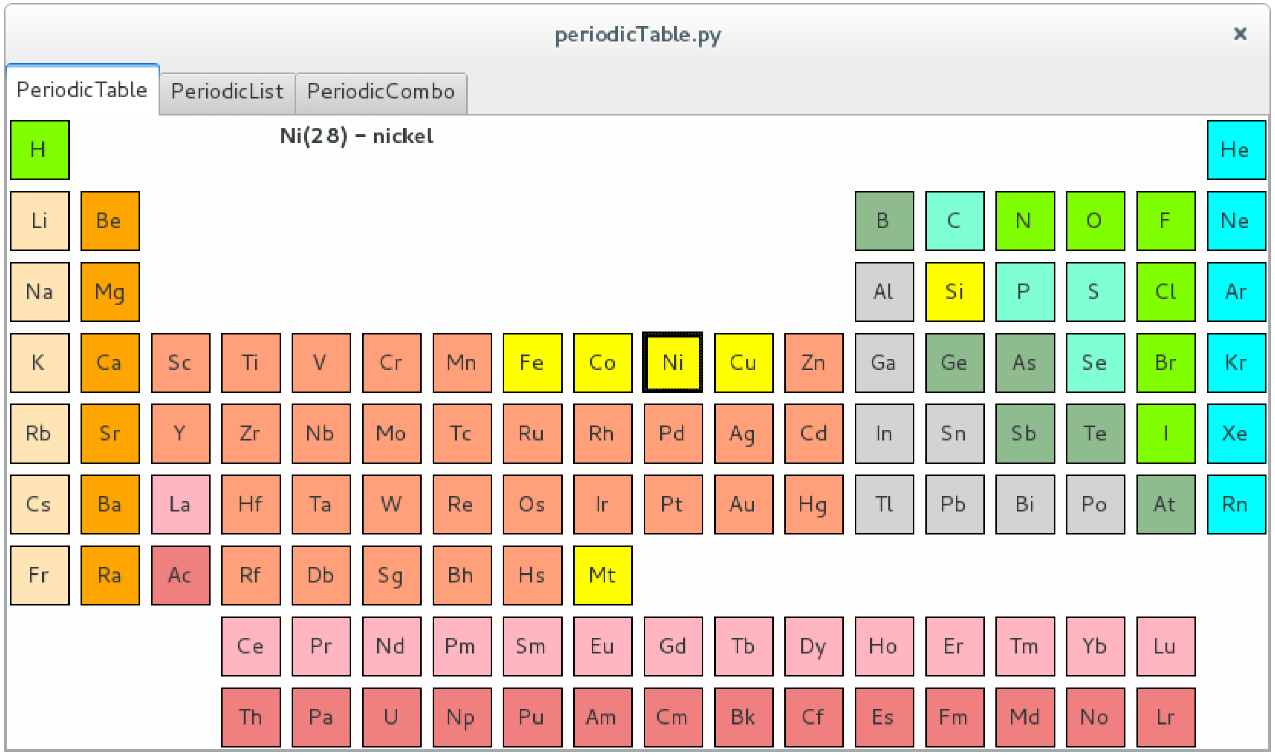
Rows dimension 0 Columns dimension 2

	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

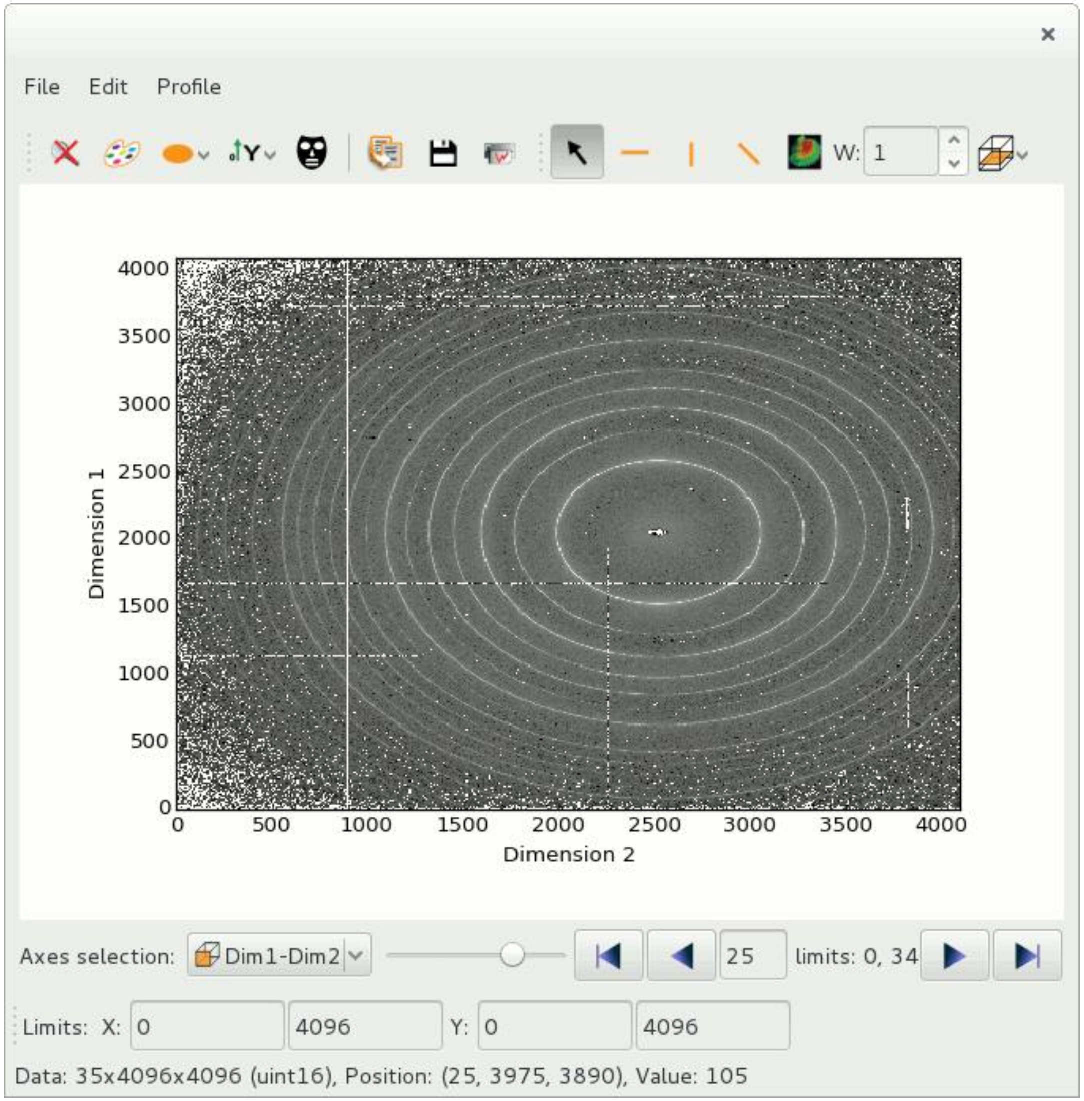


Ni(28) - nickel

H																	He
Li	Be											B	C	N	O	F	Ne
Na	Mg											Al	Si	P	S	Cl	Ar
K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
Rb	Sr	Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe
Cs	Ba	La	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg	Tl	Pb	Bi	Po	At	Rn
Fr	Ra	Ac	Rf	Db	Sg	Bh	Hs	Mt									
			Ce	Pr	Nd	Pm	Sm	Eu	Gd	Tb	Dy	Ho	Er	Tm	Yb	Lu	
			Th	Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No	Lr	

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)



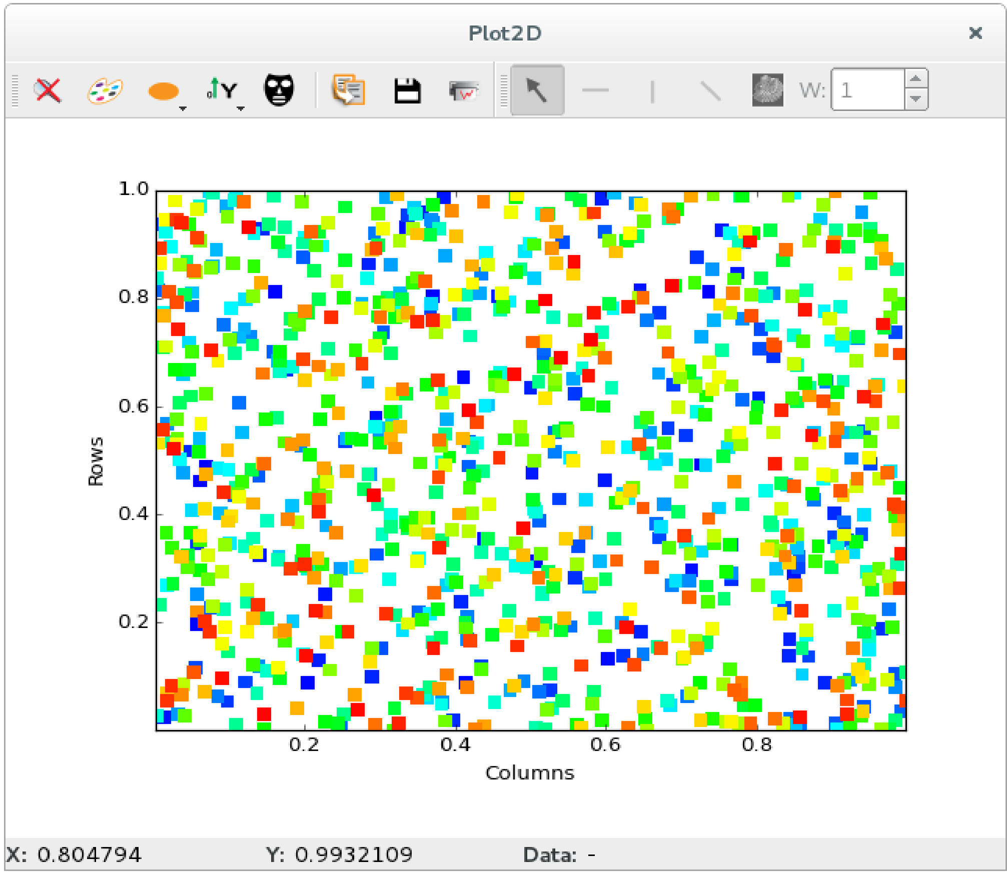
Axes selection: Dim1-Dim2 25 limits: 0, 34

Limits: X: 0 4096 Y: 0 4096

Data: 35x4096x4096 (uint16), Position: (25, 3975, 3890), Value: 105

silx.gui.plot Scatter Objects

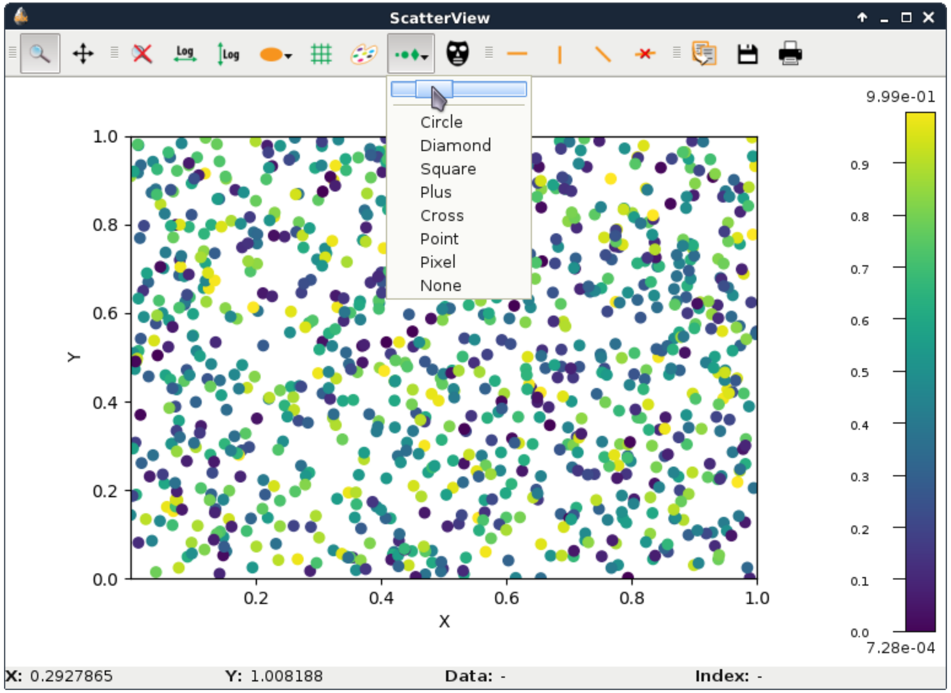
```
import numpy
import sys
from silx.gui import qt
from silx.gui.plot import Plot2D

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_())
```



X: 0.804794 Y: 0.9932109 Data: -

ScatterView: Features

- Standard plot control, colorbar
- Points size/shape control
- Mask
- Profile



```
from silx.gui.plot.ScatterView import
ScatterView
```

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

Stats Widget

Deal with:

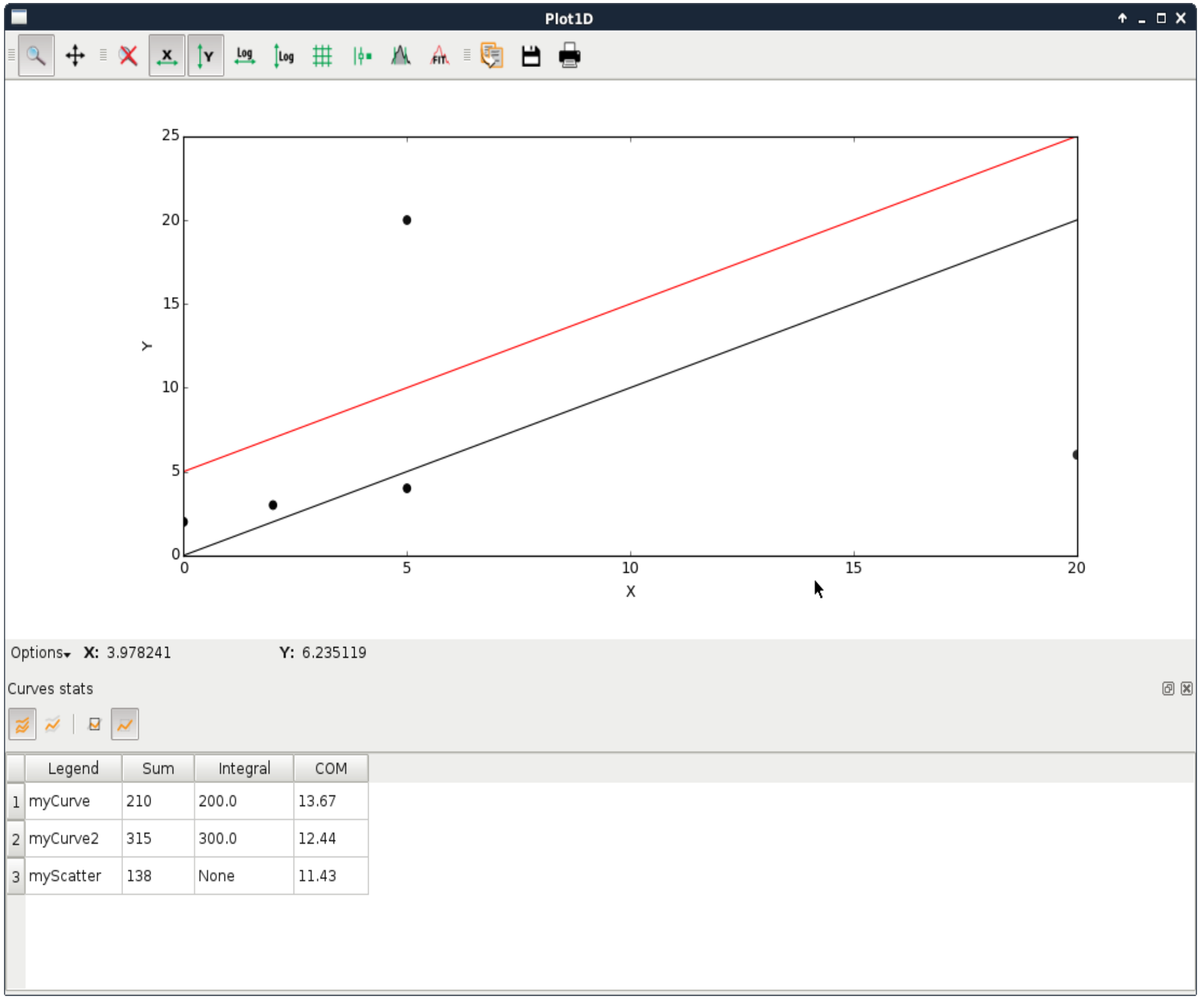
- curves
- Images
- Scatters
- Histograms

Can calculate on:

- All items or active items
- Full data range or visible one (no interpolation !!!)

Example:

examples/plotStats.py



Options▾ **X:** 3.978241 **Y:** 6.235119

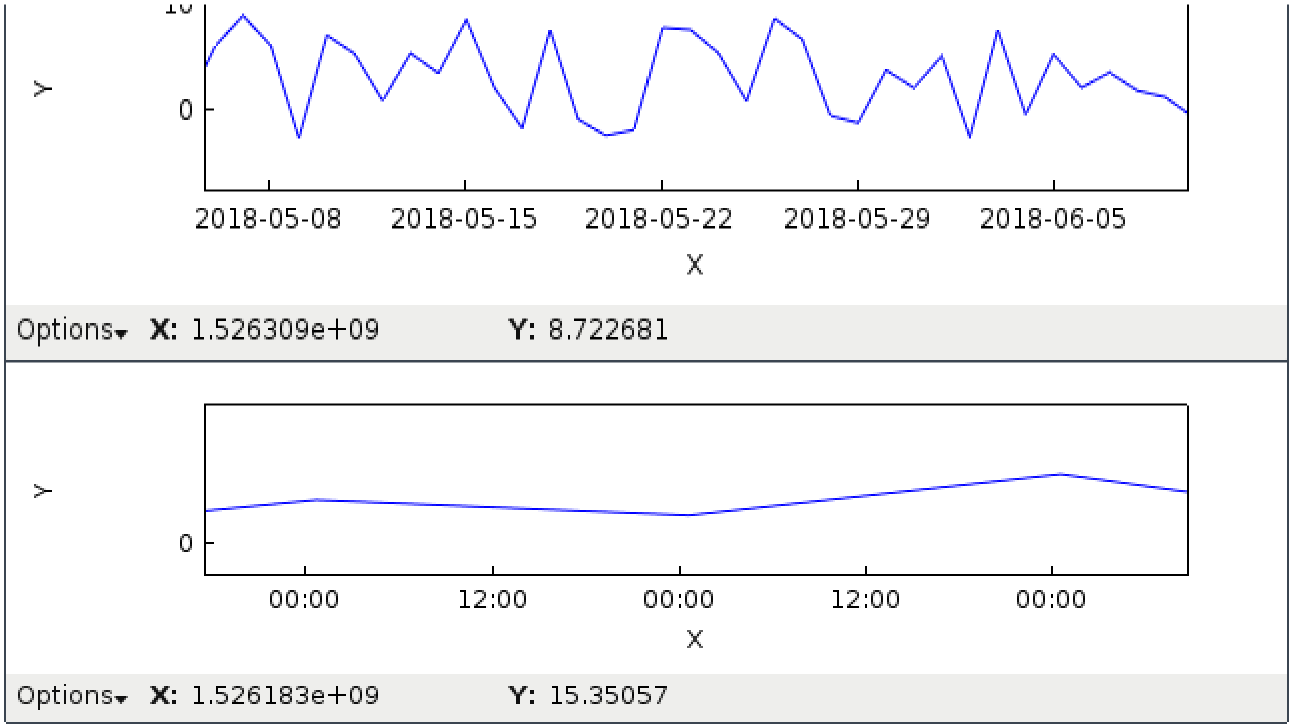
Curves stats

	Legend	Sum	Integral	COM
1	myCurve	210	200.0	13.67
2	myCurve2	315	300.0	12.44
3	myScatter	138	None	11.43

silx.gui.plot Time series

- X axis labels displayed as dates or times depending on scale
- Thanks to Pepijn Kenter (SRON: Netherlands Institute for Space Research)

Doc: http://www.silx.org/doc/silx/dev/modules/gui/plot/items.html#silx.gui.plot.items.Axis.setTickMode



Options▾ **X:** 1.526309e+09 **Y:** 8.722681

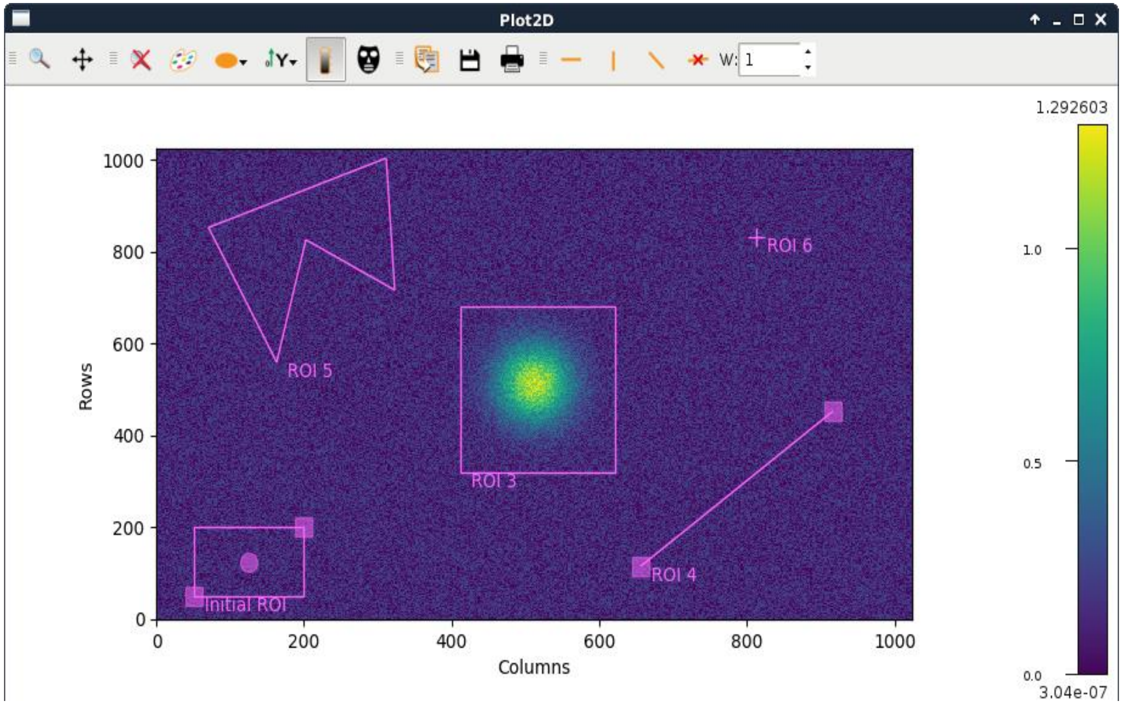
Options▾ **X:** 1.526183e+09 **Y:** 15.35057

Interactive Regions of Interest

- `silx.gui.plot.tools.roi:`
  - Regions of interest on a plot with different shapes
  - Editable interactively

Doc: http://www.silx.org/doc/silx/dev/modules/gui/plot/tools.html#module-silx.gui.plot.tools.roi

Sample code: plotInteractiveImageROI.py



Plot Widget Toolbars

- Idea: Make plot widgets more modular:
  - Allow to reuse `QAction` and `QToolBar`:

```
from silx.gui import qt
from silx.gui.plot import PlotWidget, tools
[...]
window = qt.QMainWindow()       # Create a window
plot = PlotWidget(window)       # Create a plot
window.setCentralWidget(plot)   # Place plot in window

# Add plot zoom/pan toolbar to the window
window.addToolBar(tools.InteractiveModeToolBar(parent=window, plot=plot))

# Add copy/save/print toolbar to the window
window.addToolBar(tools.OutputToolBar(parent=window, plot=plot))
[...]
window.show()
```

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

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:

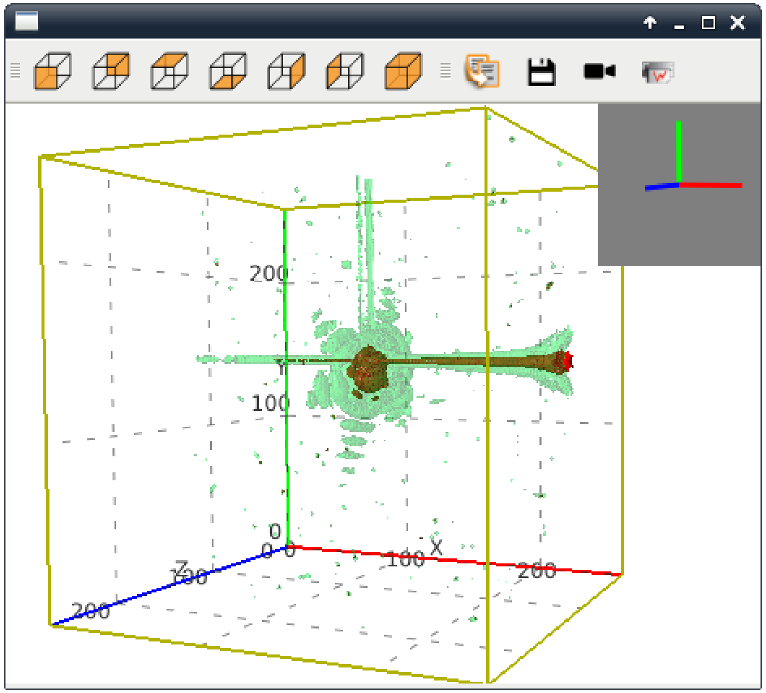
```
from silx import sx
plot = sx.Plot2D(backend='gl')
plot.show()
```

silx.gui.plot3d

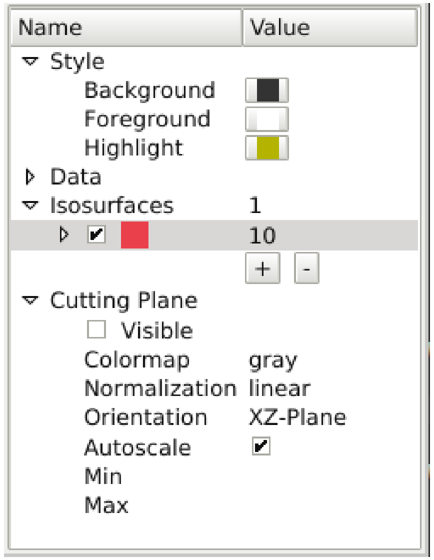
Silx 3D Visualization

- Dependencies

  - PyQt.QtOpenGL
  - PyOpenGL 3.x
  - OpenGL 2.1 subset



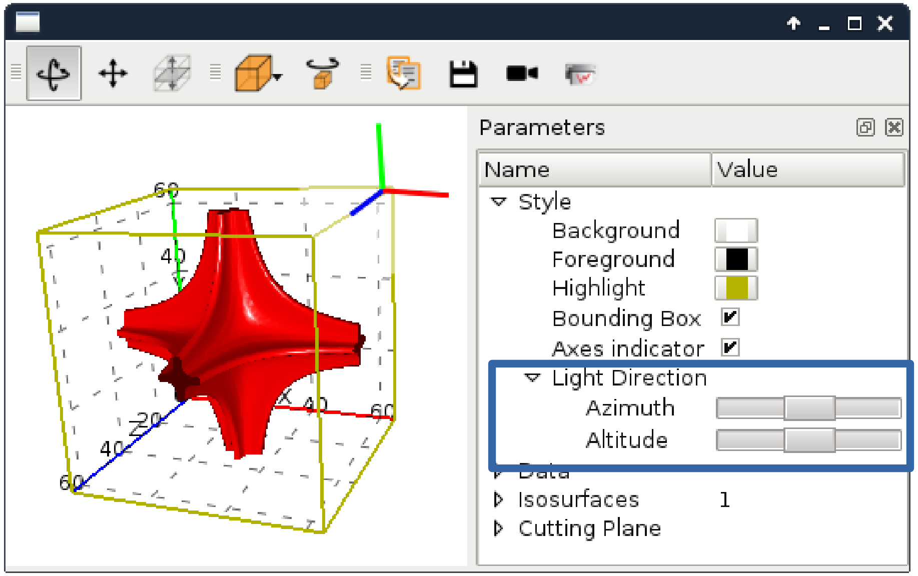
- Qt widgets for 3D plotting
  - ScalarFieldView (scalar field visualization)
  - Iso-surfaces
  - Cutting plane
- Based on an internal 3D scene structure



Name	Value
▿ Style	
Background	■
Foreground	□
Highlight	■
▹ Data	
▿ Isosurfaces	1
▹ ☑ 🟥	10
	+ -
▿ Cutting Plane	
□ Visible	
Colormap	gray
Normalization	linear
Orientation	XZ-Plane
Autoscale	☑
Min	
Max	

silx.gui.plot3d: ScalarFieldView

- Add light control

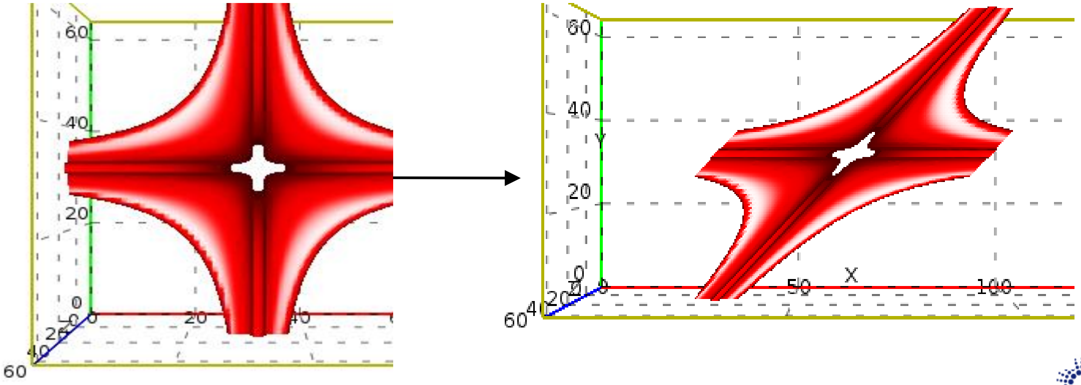


Parameters

Name	Value
▽ Style	
Background	□
Foreground	■
Highlight	■
Bounding Box	☑
Axes indicator	☑
▽ Light Direction	
Azimuth	
Altitude	
▷ Data	
▷ Isosurfaces	1
▷ Cutting Plane	

- Support of 3x3 matrix transform (for non-orthogonal axes support) to 3D scalar field visualization widget (`ScalarFieldView`):

```
scalarFieldView.setTransformMatrix((
      (1., 1., 0.),
      (0., 1., 0.),
      (0., 0., 1.)))
```

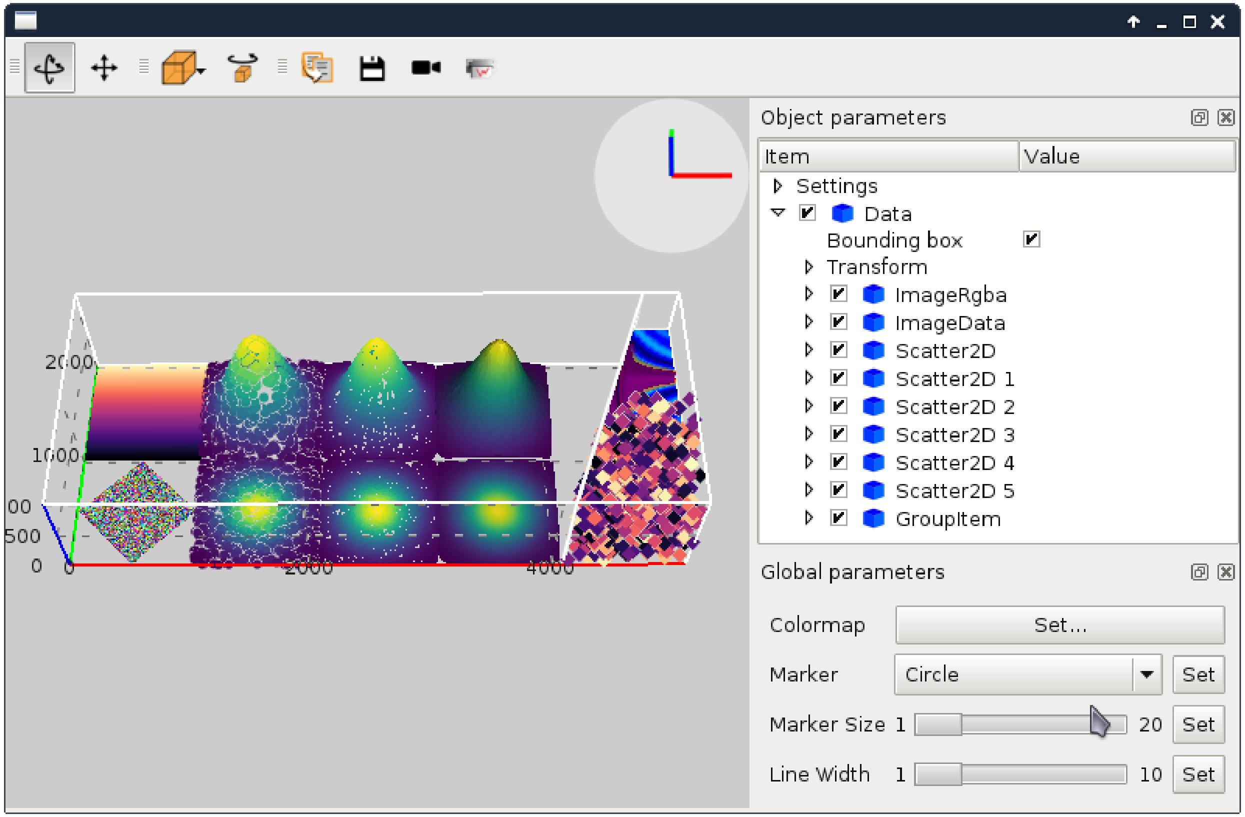


6th silx code camp – Jan 18, 2018 - silx team

silx.gui.plot3d: Scene widgets

General purpose 3D visualization widget and associated tools:

- Goal: Replacement candidate for PyMca OpenGL tab



Object parameters

Item	Value
▷ Settings	
▽ ☑ Data	
Bounding box	☑
▷ Transform	
▷ ☑ ImageRgba	
▷ ☑ ImageData	
▷ ☑ Scatter2D	
▷ ☑ Scatter2D 1	
▷ ☑ Scatter2D 2	
▷ ☑ Scatter2D 3	
▷ ☑ Scatter2D 4	
▷ ☑ Scatter2D 5	
▷ ☑ GroupItem	

Global parameters

Colormap: Set...

Marker: Circle — Set

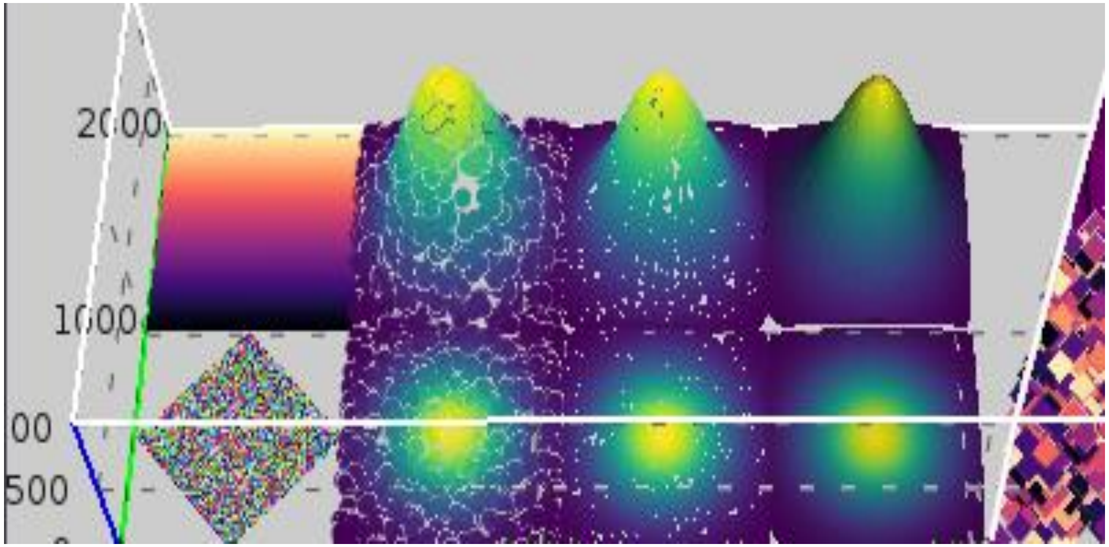
Marker Size 1 — 20 — Set

Line Width 1 — 10 — Set

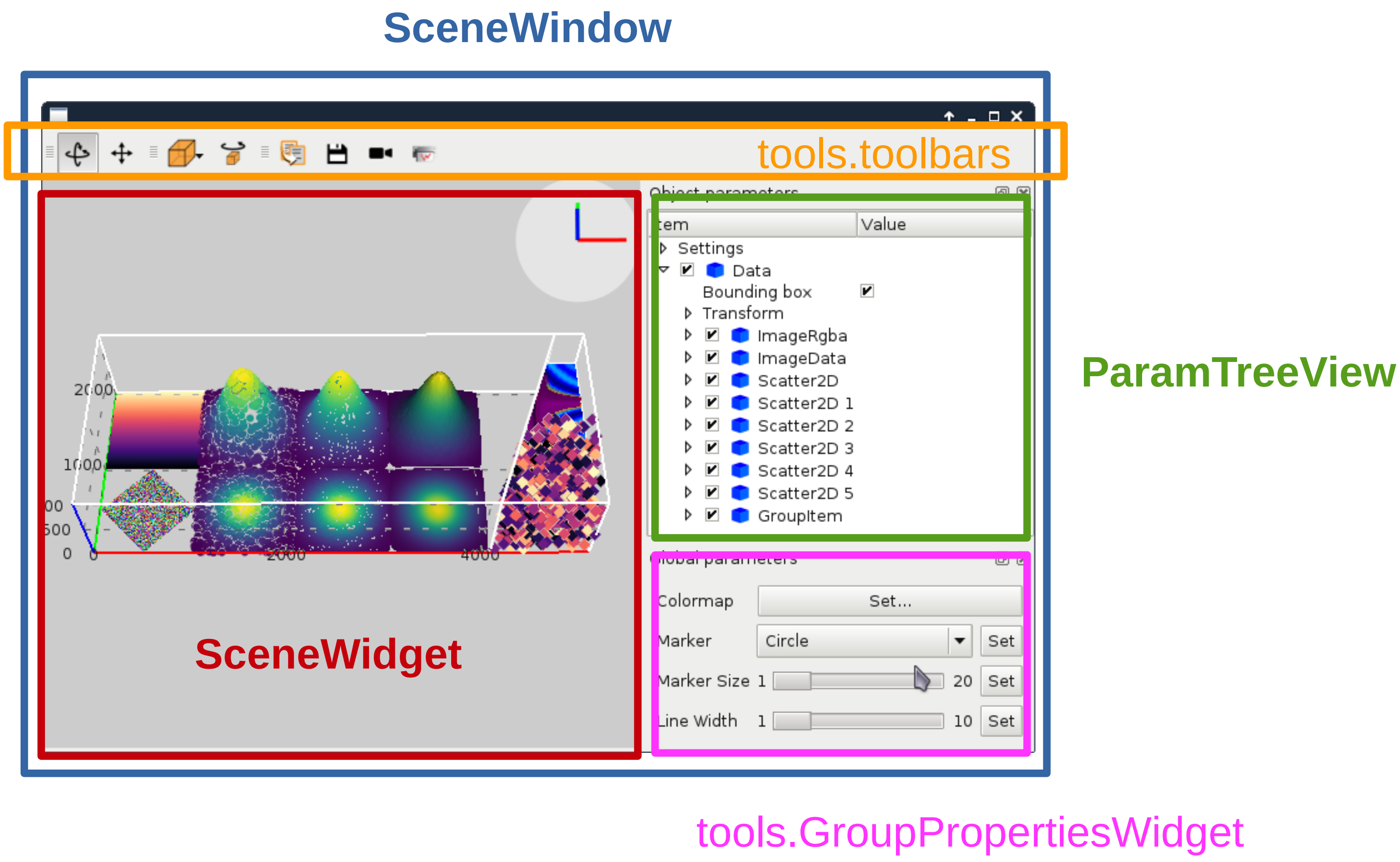
silx.gui.plot3d: Scene available items

`silx.gui.plot3d.items`:

- Images: `ImageData`, `ImageRgba`
- Scatter plots: `Scatter2D`, `Scatter3D`
- Scalar fields (with a cut plane and isosurfaces): `ScalarField3D`
- A clipping plane: `ClipPlane`
- 3D meshes: `Mesh`
- Groups: `GroupItem`, `GroupWithAxesItem`



silx.gui.plot3d: Scene widgets structure



SceneWindow

tools.toolbars

ParamTreeView

SceneWidget

tools.GroupPropertiesWidget

Object parameters

Item	Value
▷ Settings	
▽ ☑ Data	
Bounding box	☑
▷ Transform	
▷ ☑ ImageRgba	
▷ ☑ ImageData	
▷ ☑ Scatter2D	
▷ ☑ Scatter2D 1	
▷ ☑ Scatter2D 2	
▷ ☑ Scatter2D 3	
▷ ☑ Scatter2D 4	
▷ ☑ Scatter2D 5	
▷ ☑ GroupItem	

Global parameters

Colormap Set...

Marker Circle Set

Marker Size 1 20 Set

Line Width 1 10 Set

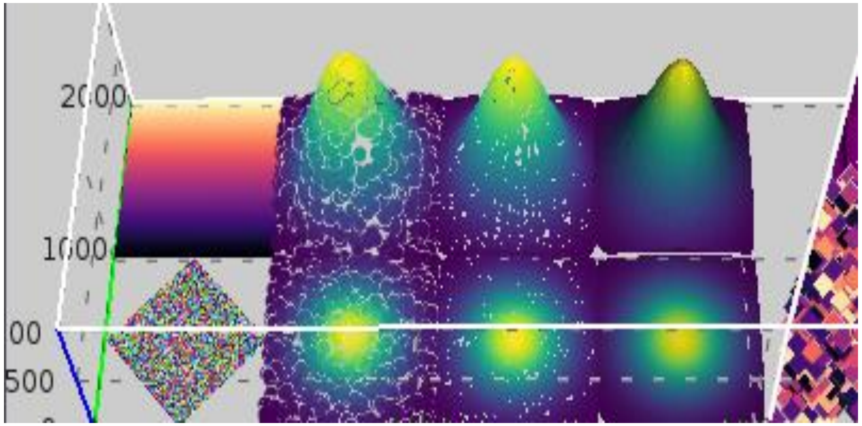
silx.gui.plot3d: ParamTreeView

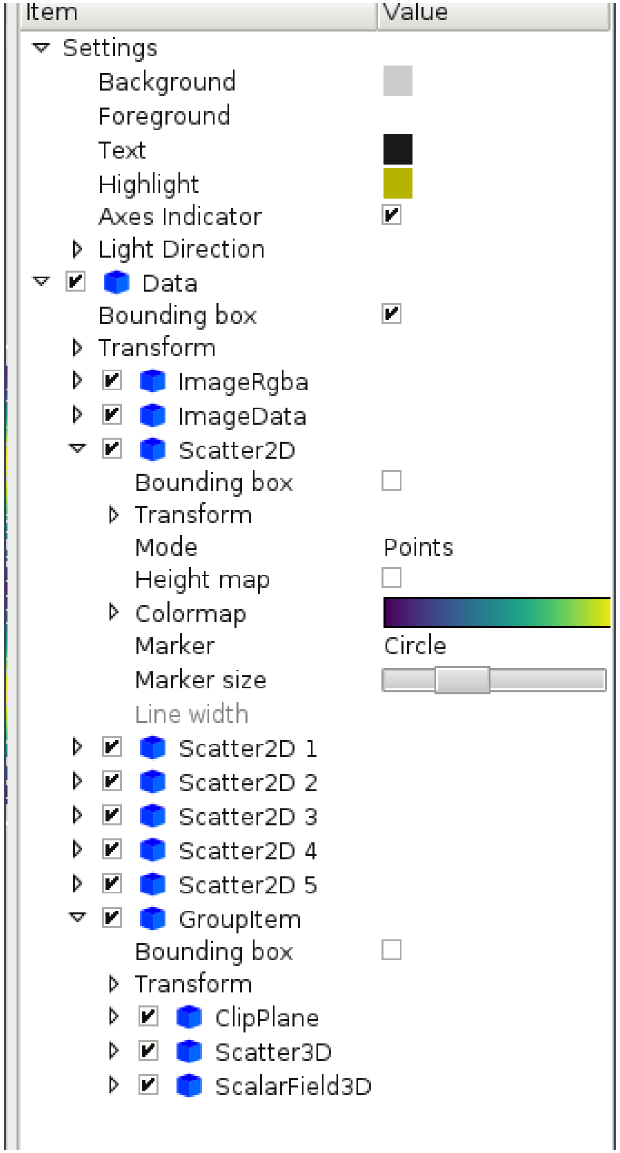
Content/Parameter tree based on:

- `silx.gui.plot3d.ParamTreeView`
- `SceneWidget.model()`

- If there is interest,

this can be adapted to 1D, 2D `PlotWidget`

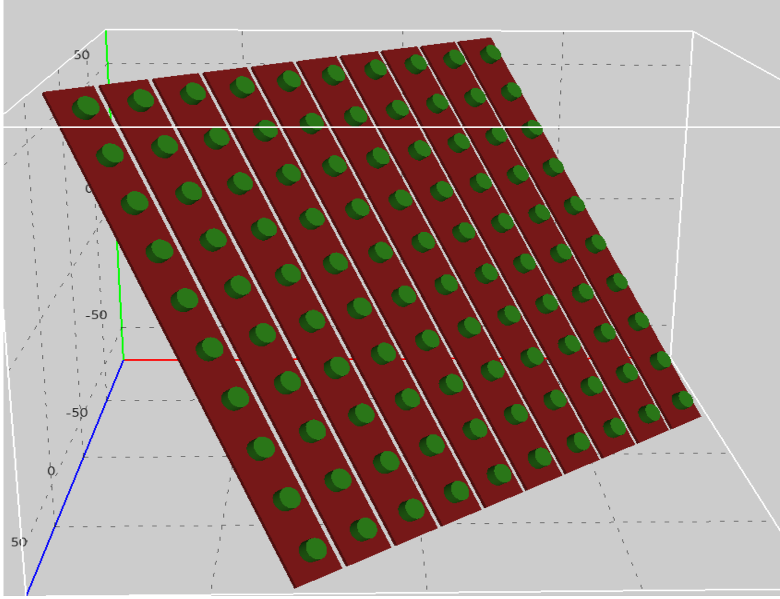




Item	Value
▽ Settings	
Background	⬜
Foreground	
Text	⬛
Highlight	🟨
Axes Indicator	☑
▷ Light Direction	
▽ ☑ Data	
Bounding box	☑
▷ Transform	
▷ ☑ ImageRgba	
▷ ☑ ImageData	
▽ ☑ Scatter2D	
Bounding box	☐
▷ Transform	
Mode	Points
Height map	☐
▷ Colormap	
Marker	Circle
Marker size	
Line width	
▷ ☑ Scatter2D 1	
▷ ☑ Scatter2D 2	
▷ ☑ Scatter2D 3	
▷ ☑ Scatter2D 4	
▷ ☑ Scatter2D 5	
▽ ☑ GroupItem	
Bounding box	☐
▷ Transform	
▷ ☑ ClipPlane	
▷ ☑ Scatter3D	
▷ ☑ ScalarField3D	

silx.gui.plot3d: Simple 3D Shapes

- Simple shapes: Cubes, cylinders, hexagons
- Allows to render many similar shapes at once
- Thanks to Guillaume Communie (ISDD/Detector & Electronics)



silx.gui.plot3d: Future

- Interaction:
  - Item selection
  - Picking of data
  - Selection/edition of Region of Interest (line, box)
- Display of statistical indicators (at least for 3D scalar fields)
- Additional scene items:
  - Surface plot for images
  - 3D complex data as colormapped isosurfaces
  - Vector field
  - …
- Testing: Lack of automated tests
- Visual improvements: transparency, ticks and labels layout...
- Optimizations:
  - Benchmarking
  - Threaded computation of isosurfaces, delaunay

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

Median Filter (C++)

silx.math.medianfilter

medfilt(data, kernel_size=3, bool conditional=False)

- 1D-2D median filter

  - data: 1D or 2D numpy array
  (specialized functions medfilt1d and medfilt2d available)
  - kernel_size int or tuple
  - Conditional if True apply conditional median filtering
  (apply only if pixel value is window minimum or maximum)

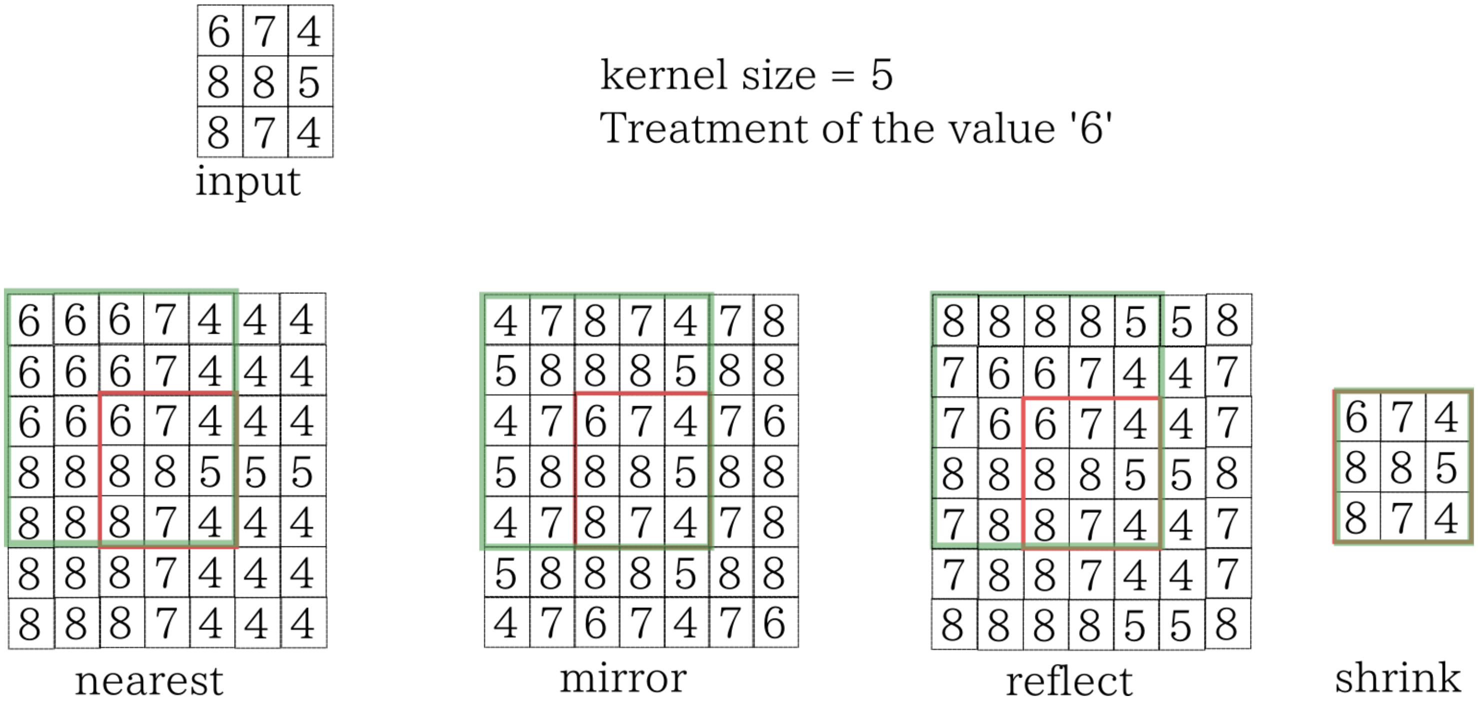
- Example:

```
from silx.math.medianfilter import medfilt2d
dataOut = medfilt2d(image,
                    kernel_size=(3, 3),
                    conditional=False)
```

Median Filter (silx.math.medianfilter)

Previously only 'nearest' mode.

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



kernel size = 5

Treatment of the value '6'

6	7	4
8	8	5
8	7	4

input

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')
```

Median Filter (GPU)

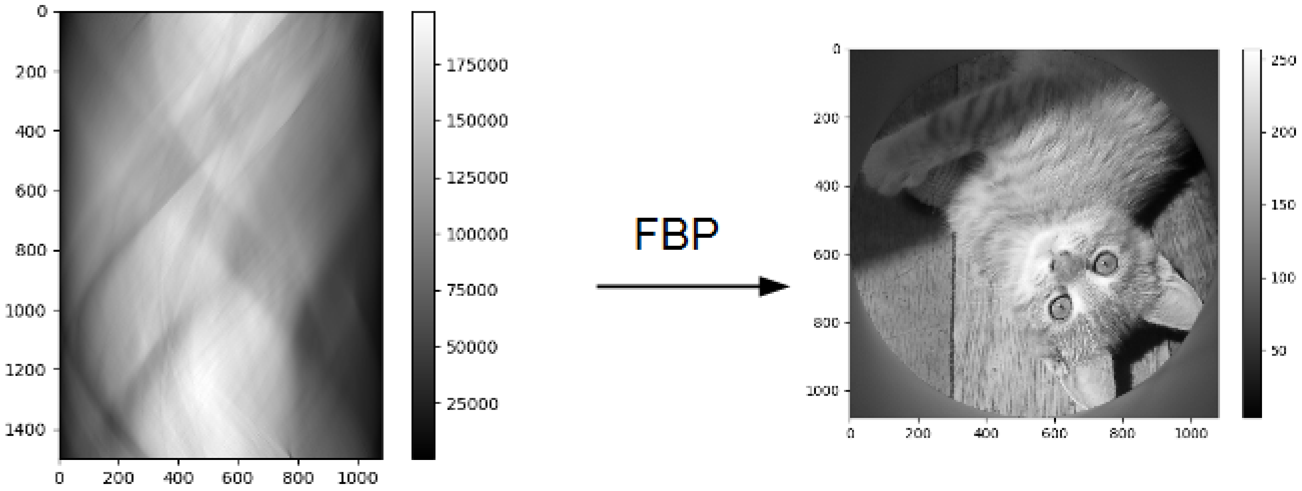
silx.opencl.medfilt2d

- OpenCL implementation of the median filter
  - Works best on GPU, and large neighborhood
  - PR pending (not yet merged)

```
from silx.opencl import medfilt2d
from scipy.misc import ascent
from scipy.ndimage import filters

img = ascent().astype(“float32”)
%timeit filters.median_filter(img, (55,55))

import silx.image
%timeit silx.image.medfilt2d(img, (55,55))

from silx.opencl import medifilt
%timeit medfilt.medfilt2d(img, (55,55))
```

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)

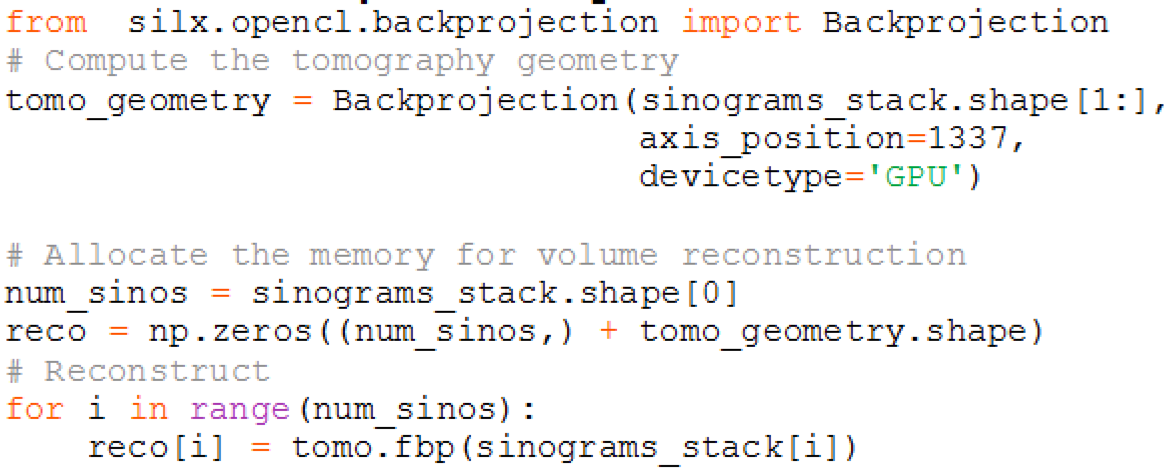


sinogram

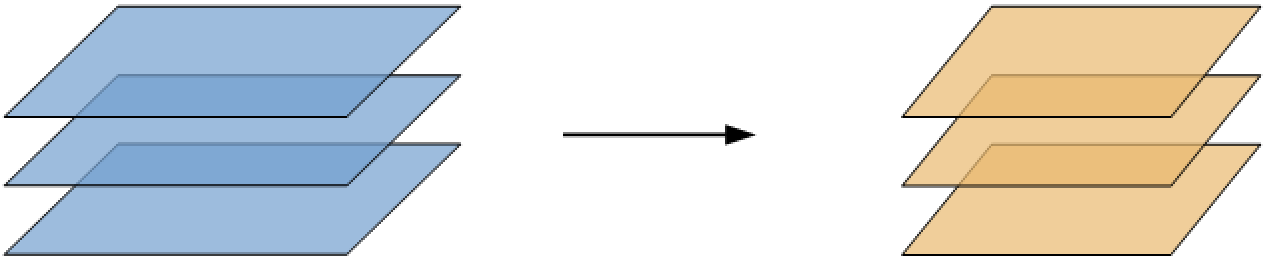
slice

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]

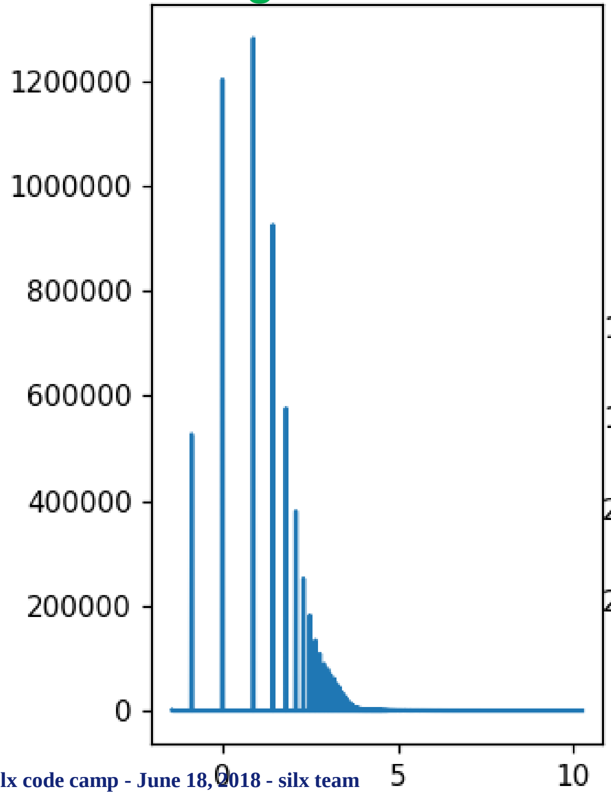


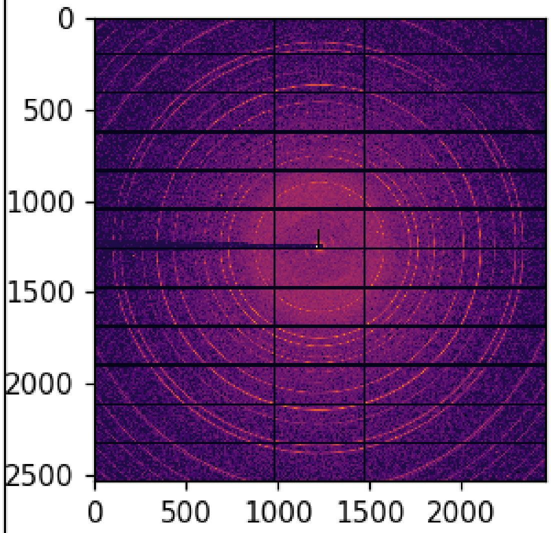
```
from  silx.opencl.backprojection import Backprojection
# Compute the tomography geometry
tomo_geometry = Backprojection(sinograms_stack.shape[1:],
                               axis_position=1337,
                               devicetype='GPU')

# 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])
```



CoDec : Byte offset for CBF processing on GPU

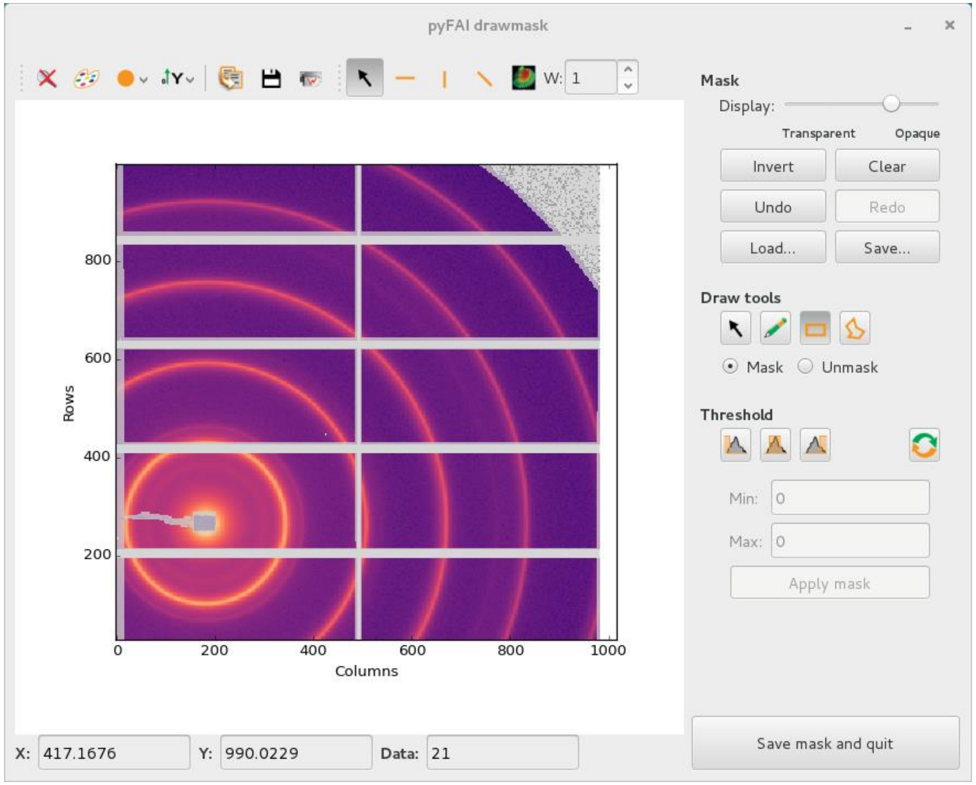
- `silx.opencl.codec.byte_offset`
  - OpenCL-based CBF compression
- 10x speed-up for compression/decompression of CBF streams
  - Compatible with the new Image processing framework
  - Compatible with pyFAI azimuthal integration
- Accepted in J. Synchrotron Radiation https://doi.org/10.1107/S1600577518000607





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

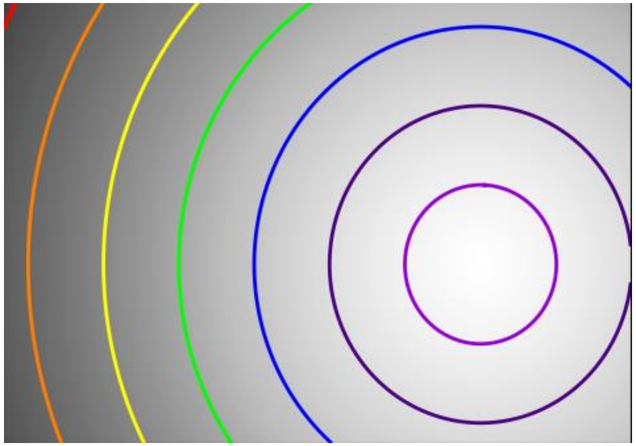


X: 417.1676 Y: 990.0229 Data: 21

- Marching Squares
- Median Filter
  - GPU accelerated via OpenCL

Marching Squares Algorithm

- **Designed to speed up PyFAI calibration GUI**
  - Cython + OpenMP
  - Support masking
  - Optimization to reach many contours from the same gradient image
  - Reach contours or pixels
- **Example:** `examples/findContours.py`



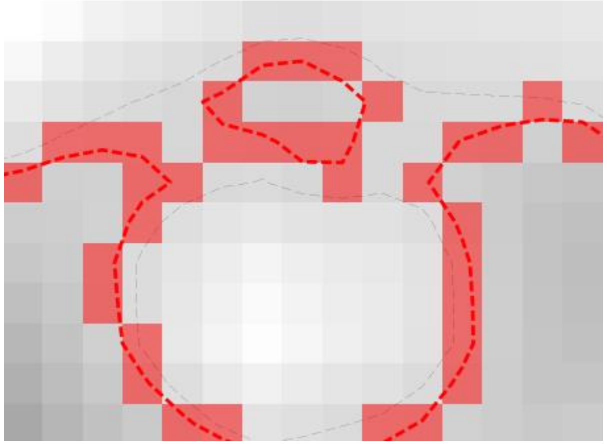


Image processing on opencl devices (GPU)

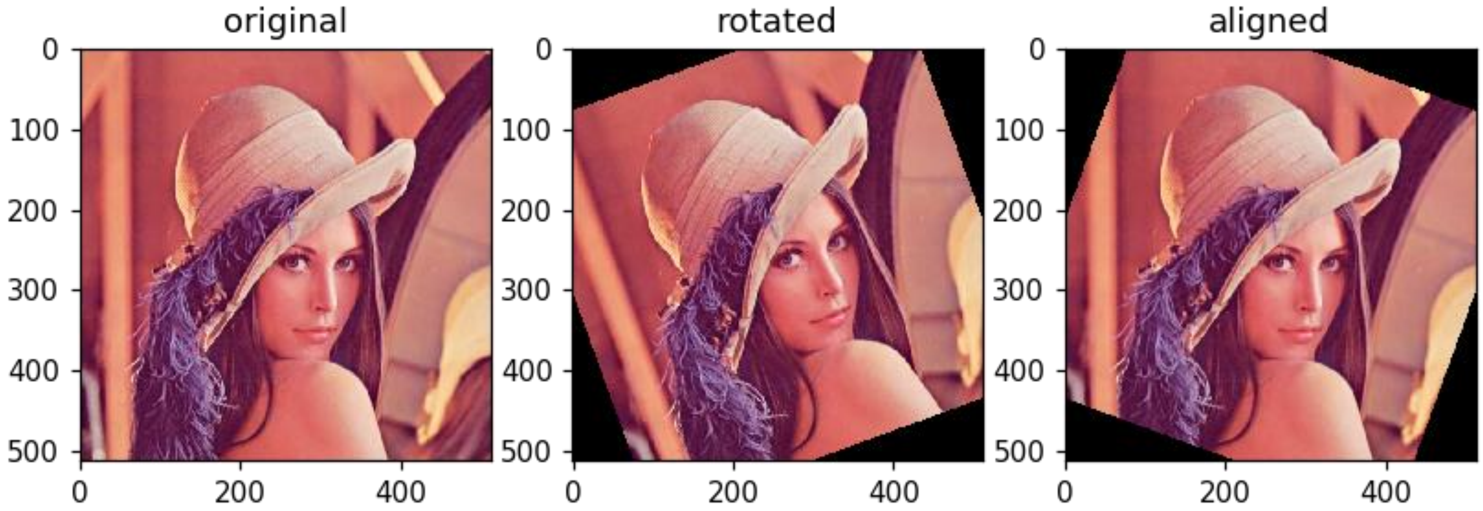
- New image processing framework:
  - Allows to exchange buffers on the device
  - Allows the creation of work-flow without copying data back & forth
  - Better performances

- Few image treatments implemented:
  - Buffer conversion to float arrays from any integer
  - Min/Max search (double-reduction)
  - Image normalization
  - Image histogram

- Tutorial available:
  - https://github.com/kif/silx/blob/1199_ocl_image/doc/source/Tutorials/Image.ipynb

Sift: Image alignment

- Use the “image” framework.
- Major re-work for compatibility with PyOpenCL > 2015
- Compatibility with “spectre” corrections
- Many memory-leak corrected
- New tutorial based on jupyter notebook.
  https://github.com/silx-kit/silx/blob/master/doc/source/Tutorials/Sift/sift.ipynb

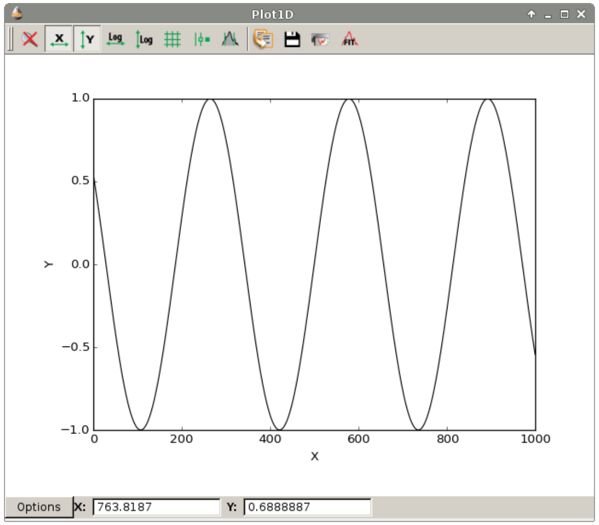


silx.sx: a module to simplify interactive use

pylab like module on steroids

- 1D plotting: ROI, fitting & printing

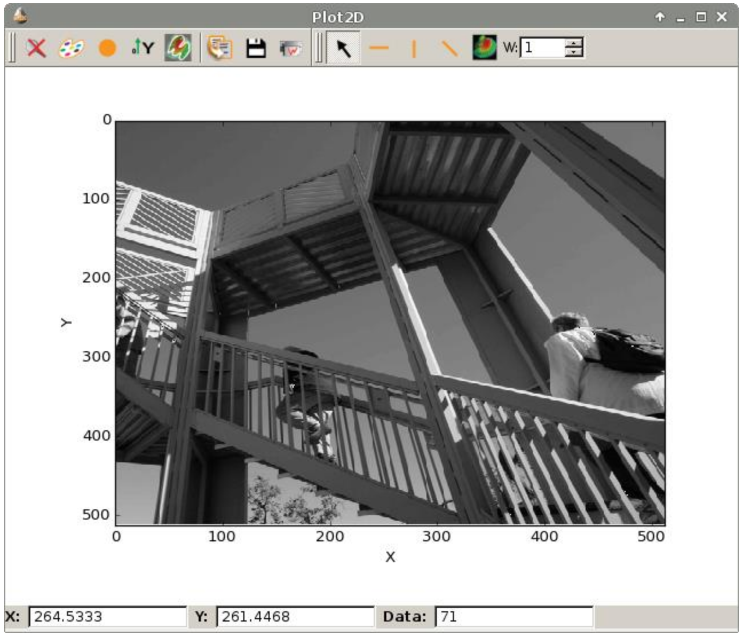
```
>>> from silx import sx
>>> from numpy import sin, linspace
>>> sx.plot(sin(linspace(-10, 10, 1000)))
```



X: 763.8187 Y: 0.6888887

- 2D display: intensity, mask, profile

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



W: 1

X: 264.5333 Y: 261.4468 Data: 71

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.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.nxdata

- New functions
  - is_NXentry_with_default_NXdata(group)
  - is_NXroot_with_default_NXdata(group)
  - get_default(group)
    - Returns default silx.io.NXdata object or None. Group parameter can be NXdata, NXentry or NXroot.

  - save_NXdata(filename, signal, axes=None,
    ```
          signal_name=“data”, axes_names=None,
          signal_long_name=None, axes_long_names=None,
          signal_errors=None, axes_errors=None,
          title=None, interpretation=None,
          nxentry_name="entry", nxdata_name=None)
    ```

silx.io.convert

- Module

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

- Application

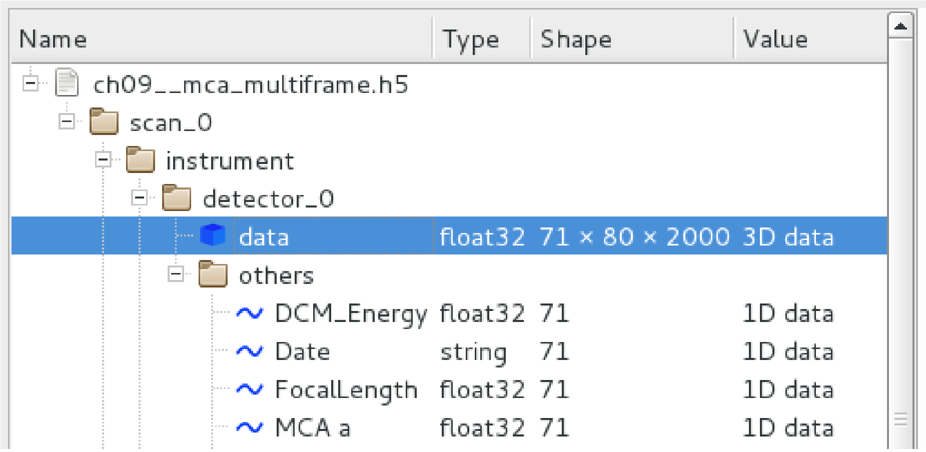
- New command line application to convert files to HDF5

silx convert –help
silx convert filename

silx convert

- Convert series of single frame images (EDF, TIFF...) into a HDF5 multiframe stack

```
silx convert --file-pattern ch09__mca_0005_0000_%d.edf -o ch09__mca_multiframe.h5
```



Name	Type	Shape	Value
ch09__mca_multiframe.h5			
scan_0			
instrument			
detector_0			
data	float32	71 × 80 × 2000	3D data
others			
DCM_Energy	float32	71	1D data
Date	string	71	1D data
FocalLength	float32	71	1D data
MCA a	float32	71	1D data

```
silx convert -h
```

silx convert

- Merging SPEC and EDF files.
  - Step 1. Convert the SPEC file to HDF5 file

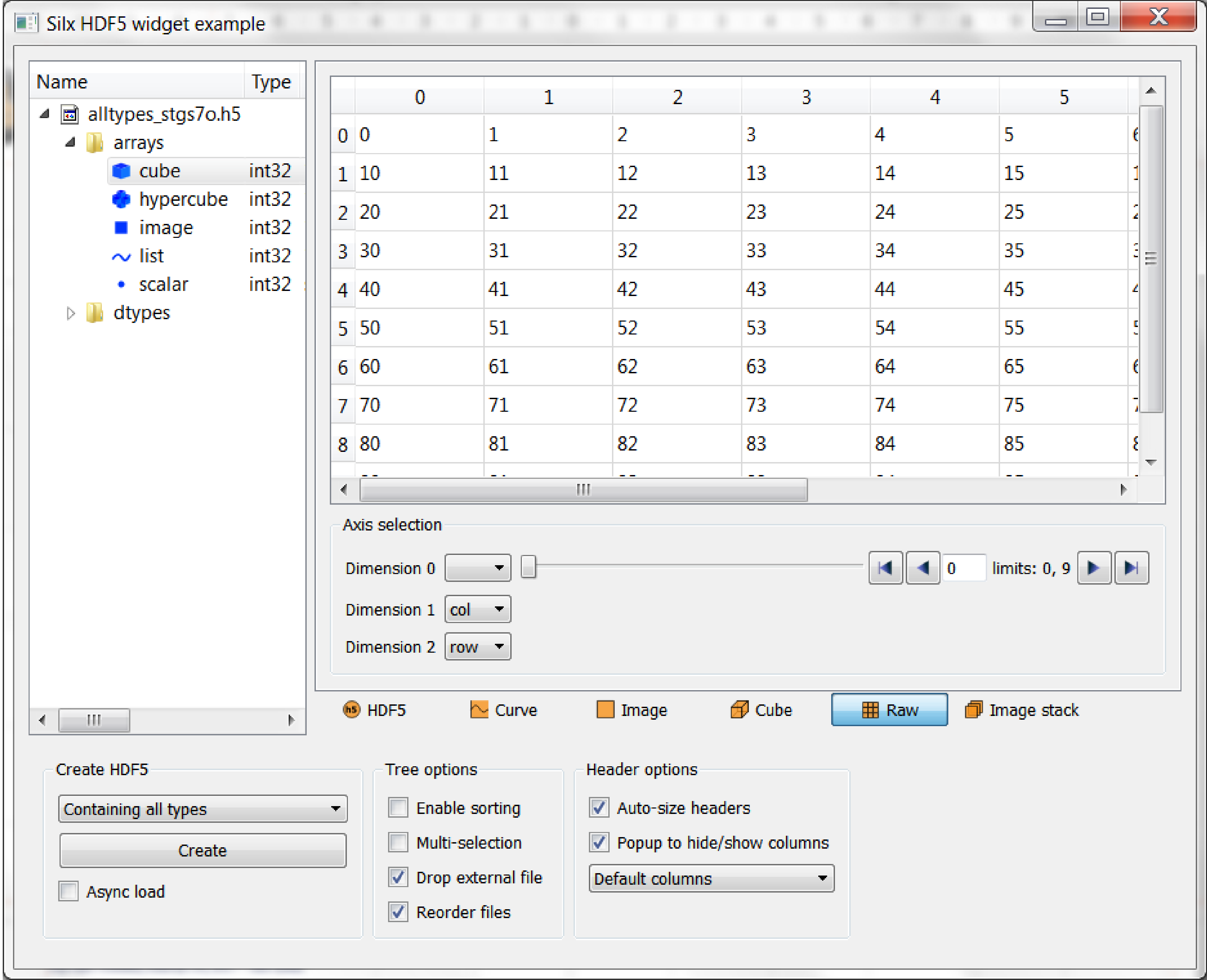
```
silx convert spec_file_name -o hdf5_file_name.h5
```

  - Step 2. Convert the EDF files selecting target path in generated HDF5 file

```
silx convert --file-pattern=root_%04d.edf --begin=100 --end=199 \
          --mode=r+ -o hdf5_file_name.h5::/1.1/instrument/detector_0
```

  - Hint Multiple indices supported (indexed files, indexed directories, …)

```
root_ssss_dddd_nnnn.edf

--file-pattern=root_%04d_%04d_$04d.edf -begin=1,0,0 -end=1,0,99
```



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 | 0 | limits: 0, 9

Dimension 1: col

Dimension 2: row

Create HDF5

Containing all types

☐ Async load

Tree options

☐ Enable sorting

☐ Multi-selection

☑ Drop external file

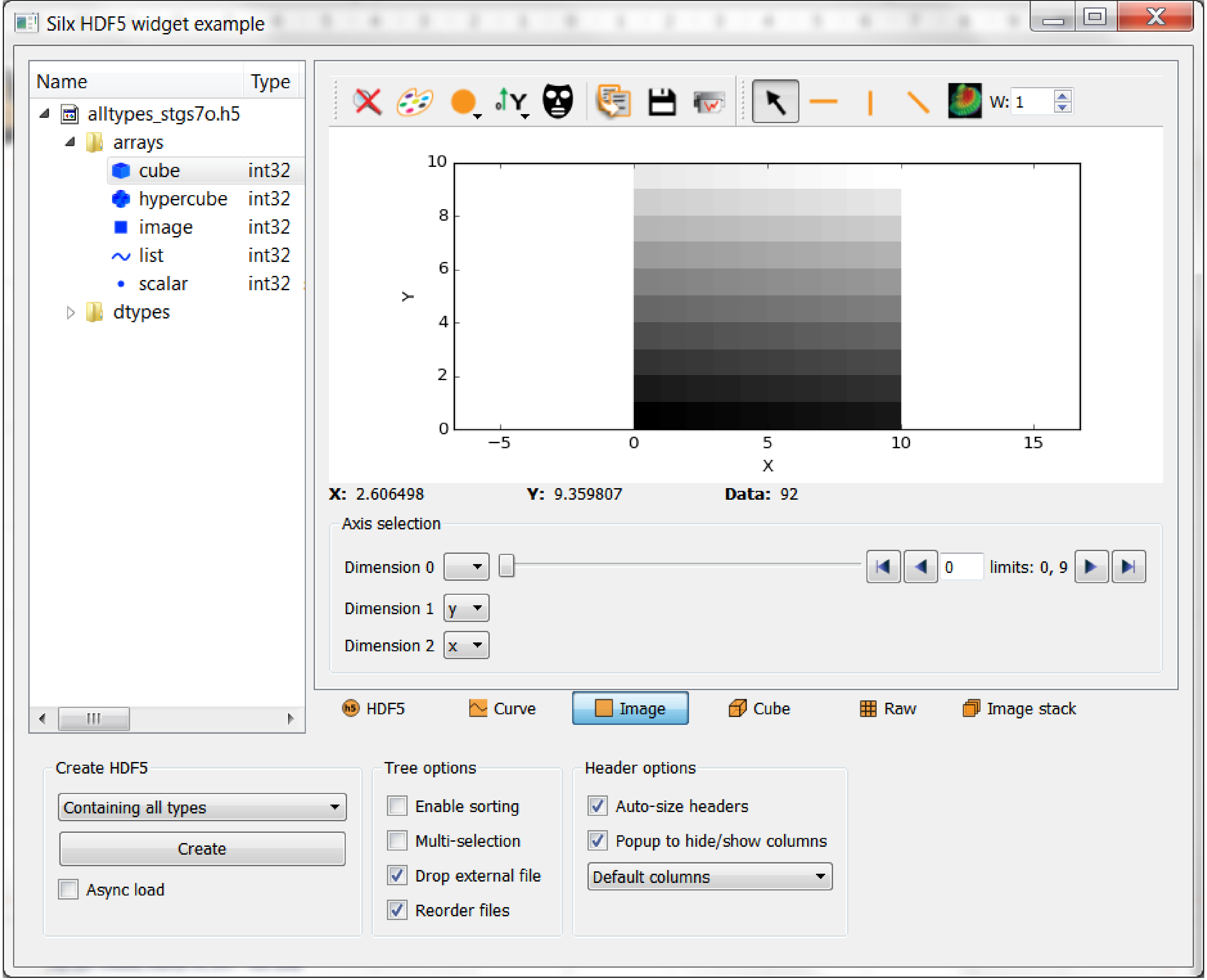
☑ Reorder files

Header options

☑ Auto-size headers

☑ Popup to hide/show columns

Default columns



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 0 limits: 0, 9

Dimension 1 y

Dimension 2 x

Create HDF5

Containing all types

□ Async load

Tree options

□ Enable sorting

□ Multi-selection

☑ Drop external file

☑ Reorder files

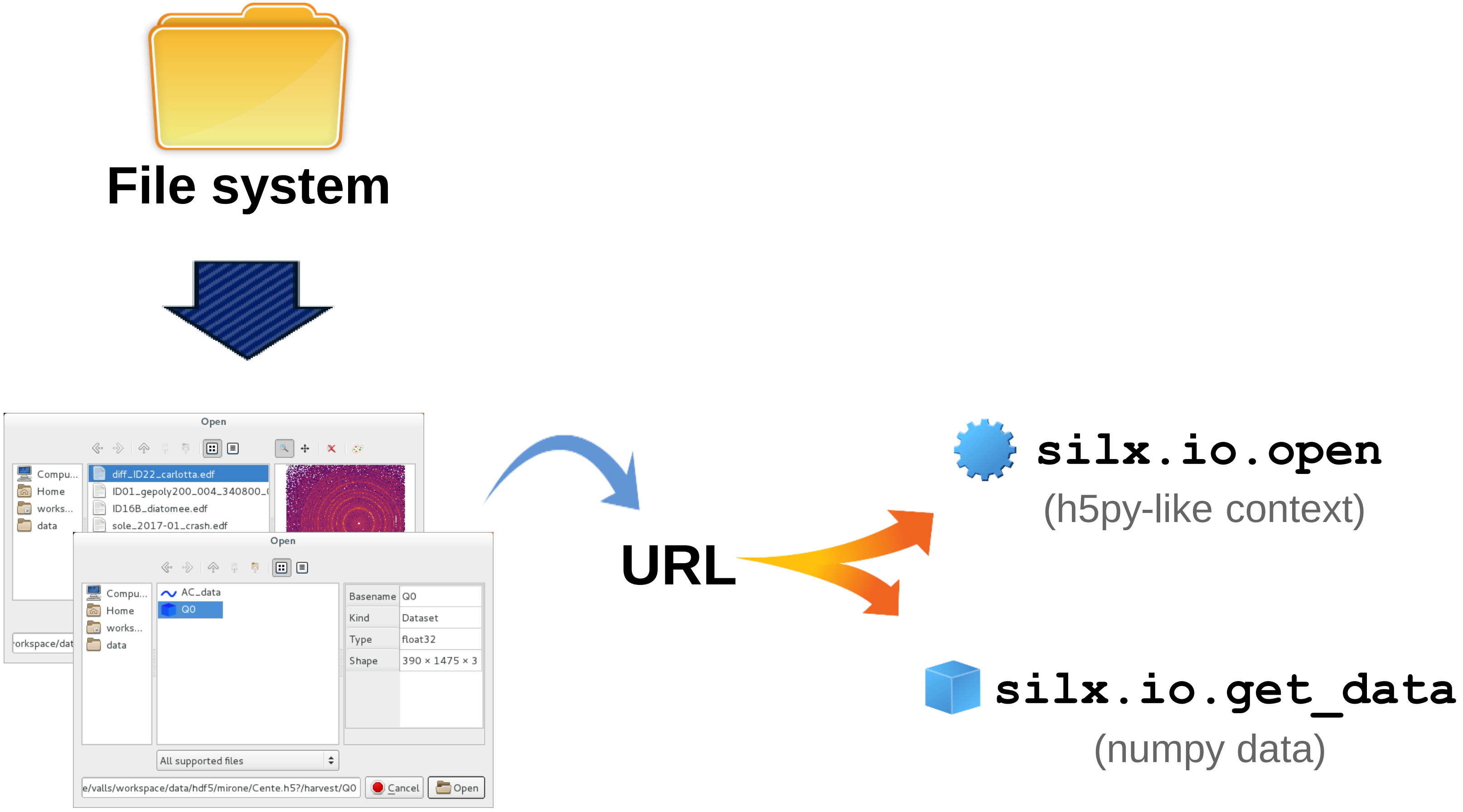
Header options

☑ Auto-size headers

☑ Popup to hide/show columns

Default columns

Dialog to reach data



File system

Open

Compu...
Home
works...
data

diff_ID22_carlotta.edf
ID01_gepoly200_004_340800_(
ID16B_diatomee.edf
sole_2017-01_crash.edf

'orkspace/dat

Open

Compu...
Home
works...
data

AC_data
Q0

Basename	Q0
Kind	Dataset
Type	float32
Shape	390 × 1475 × 3

All supported files

e/valls/workspace/data/hdf5/mirone/Cente.h5?/harvest/Q0

URL

silx.io.open

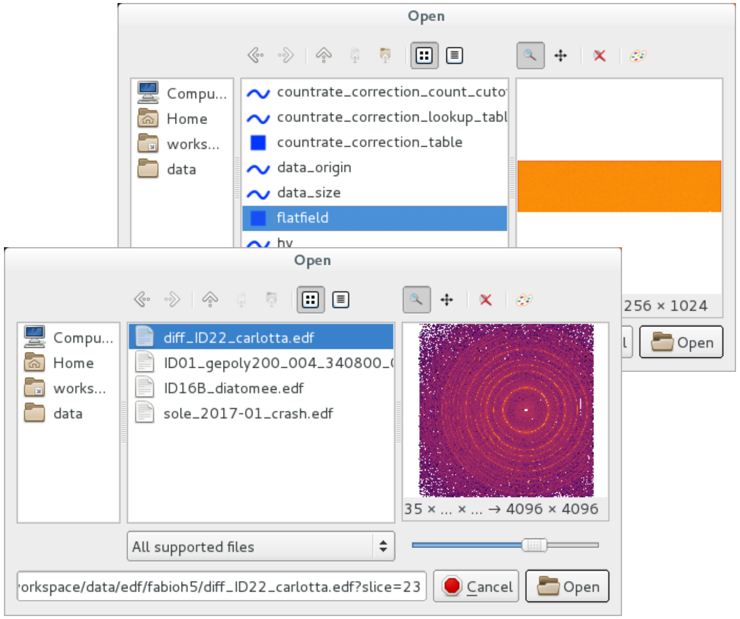
(h5py-like context)

silx.io.get_data

(numpy data)

silx.gui.dialog

Data dialogs



Open

- Compu...
- Home
- works...
- data

- countrate_correction_count_cuto
- countrate_correction_lookup_tabl
- countrate_correction_table
- data_origin
- data_size
- flatfield
- bv

256 × 1024

Open

Open

- Compu...
- Home
- works...
- data

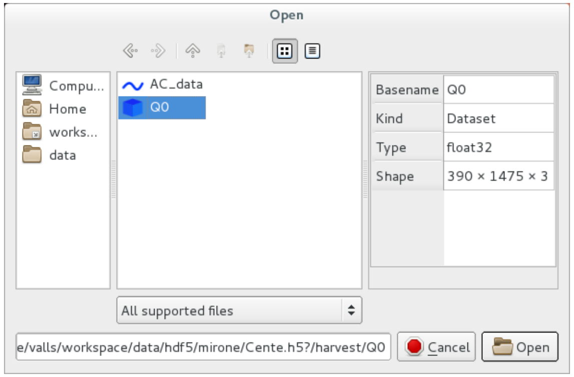
- diff_ID22_carlotta.edf
- ID01_gepoly200_004_340800_(
- ID16B_diatomee.edf
- sole_2017-01_crash.edf

35 × ... × ... → 4096 × 4096

All supported files

orkspace/data/edf/fabioh5/diff_ID22_carlotta.edf?slice=23

Cancel Open



Open

- Compu...
- Home
- works...
- data

- AC_data
- Q0

Basename	Q0
Kind	Dataset
Type	float32
Shape	390 × 1475 × 3

All supported files

e/valls/workspace/data/hdf5/mirone/Cente.h5?/harvest/Q0

Cancel Open

`ImageFileDialog`

- Specialised to select an image
- Support slicing of hypercubes
- Support h5-like
- Support raw image files (edf, tiff, cbf)

`DataFileDialog`

- Select anything from h5-like structure
- Filter to select only datasets or groups

Data URLs

- **Custom schemes**
  - `silx:///home/user/foo.edf?path=/group/&slice=5`
  - `fabio:///home/user/foo.edf?slice=5`
  - Also available for relative paths
- **Reach data from datasets and fabio URLs**

```
data = silx.io.get_data(url)
```

- **Reach data from other URLs**

```
with silx.io.open(url) as node:
    print(node)
```

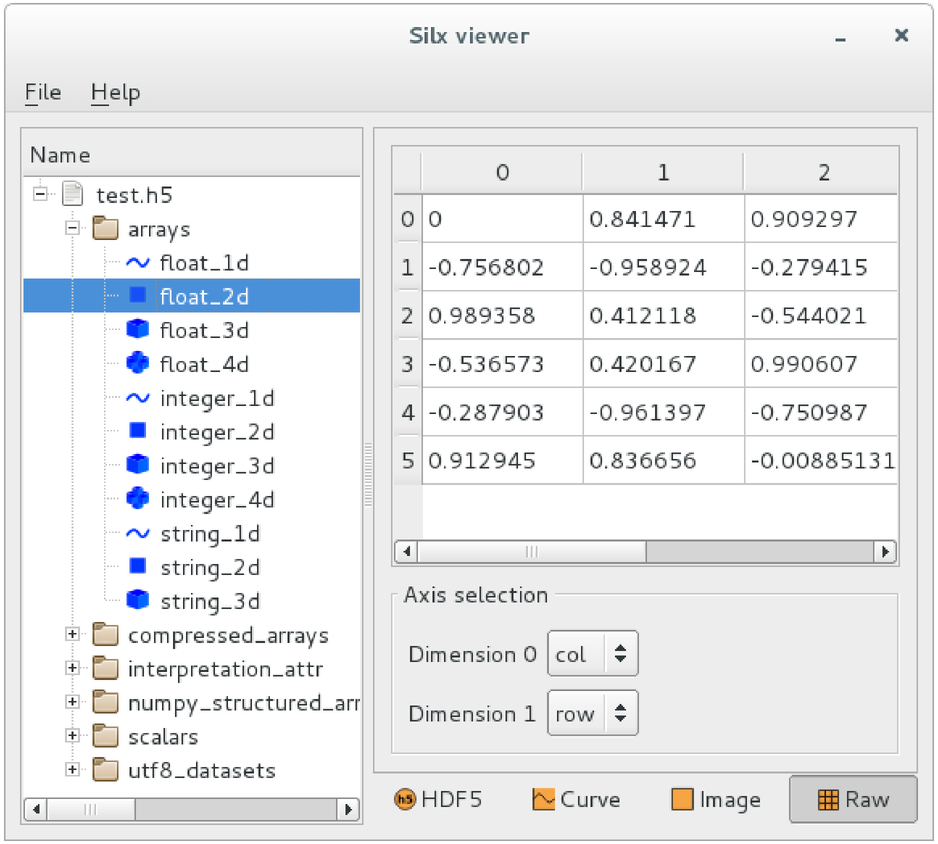
- **An object is provided to parse our URLs**
  - `silx.io.url.DataUrl`
- **We also support h5pyd URLs**
  - `http://127.0.0.1:5000/tall.public.hdfgroup.org`

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



- test.h5
  - arrays
    - float_1d
    - float_2d
    - float_3d
    - float_4d
    - integer_1d
    - integer_2d
    - integer_3d
    - integer_4d
    - string_1d
    - string_2d
    - string_3d
  - compressed_arrays
  - interpretation_attr
  - numpy_structured_arr
  - scalars
  - utf8_datasets

	0	1	2
0	0	0.841471	0.909297
1	-0.756802	-0.958924	-0.279415
2	0.989358	0.412118	-0.544021
3	-0.536573	0.420167	0.990607
4	-0.287903	-0.961397	-0.750987
5	0.912945	0.836656	-0.00885131

Axis selection

Dimension 0: col

Dimension 1: row

NXdataViewer

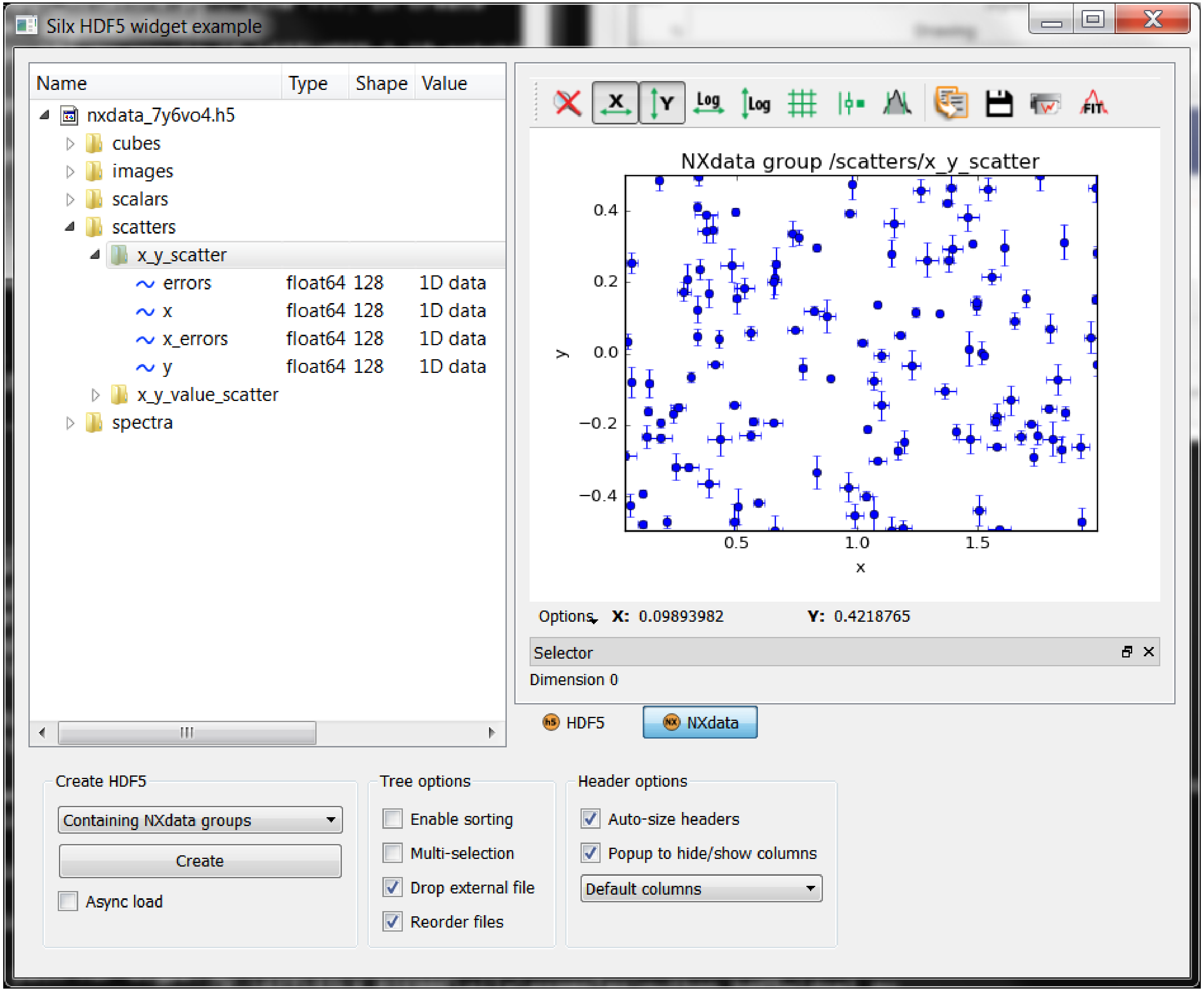
- Data viewer for viewing data in a Nexus NXdata group

- 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)

silx view – Generic Viewer Interpreting NXdata Groups



Silx HDF5 widget example

Name	Type	Shape	Value
nxdata_7y6vo4.h5			
cubes			
images			
scalars			
scatters			
x_y_scatter			
errors	float64	128	1D data
x	float64	128	1D data
x_errors	float64	128	1D data
y	float64	128	1D data
x_y_value_scatter			
spectra			

Options X: 0.09893982 Y: 0.4218765

Selector

Dimension 0

Create HDF5

Containing NXdata groups

Create

□ Async load

Tree options

□ Enable sorting

□ Multi-selection

☑ Drop external file

☑ Reorder files

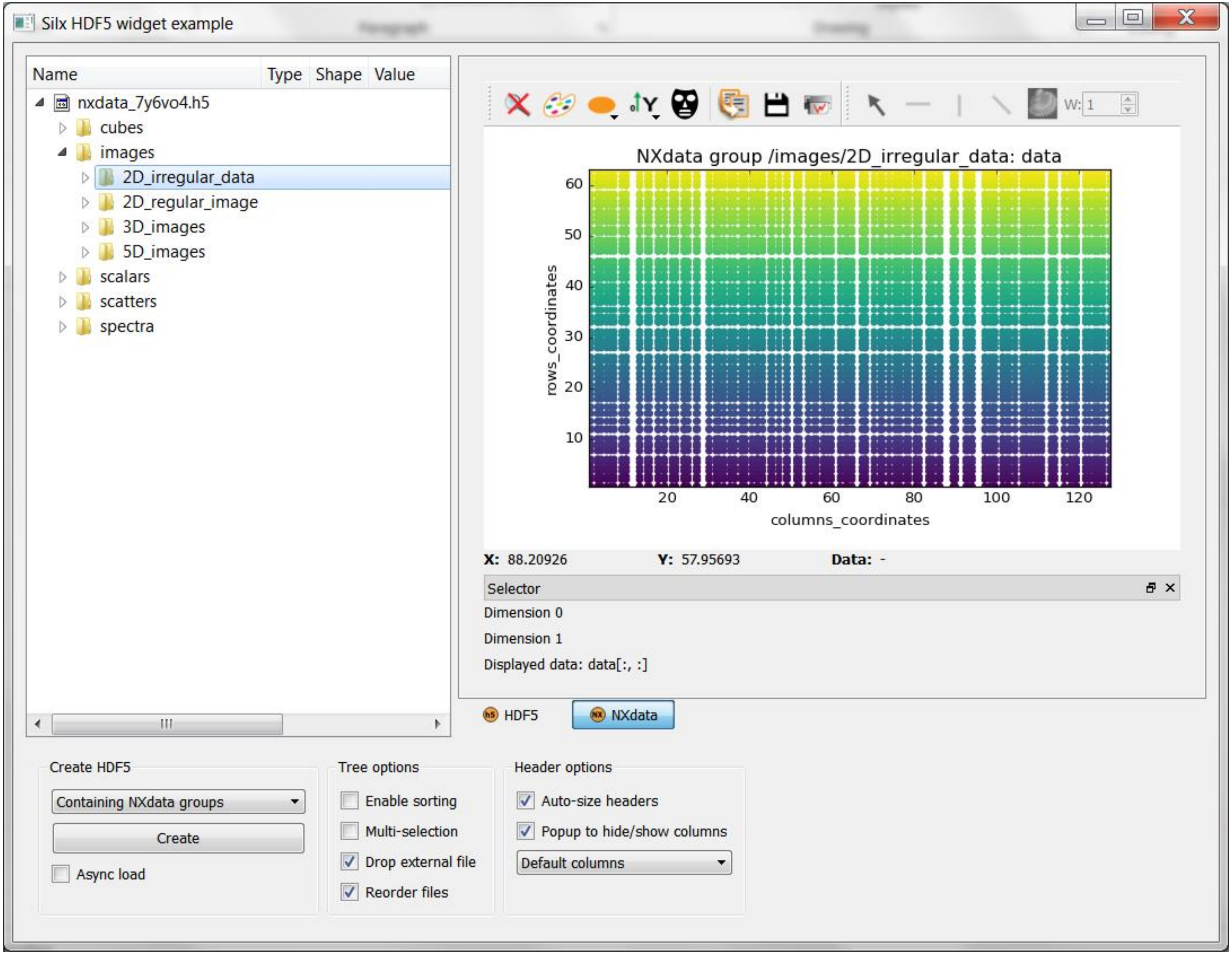
Header options

☑ Auto-size headers

☑ Popup to hide/show columns

Default columns

NXdataViewer



Silx HDF5 widget example

Name	Type	Shape	Value
nxdata_7y6vo4.h5			
cubes			
images			
2D_irregular_data			
2D_regular_image			
3D_images			
5D_images			
scalars			
scatters			
spectra			

X: 88.20926 **Y:** 57.95693 **Data:** -

Selector

Dimension 0

Dimension 1

Displayed data: data[:, :]

Create HDF5

Containing NXdata groups

□ Async load

Tree options

□ Enable sorting

□ Multi-selection

☑ Drop external file

☑ Reorder files

Header options

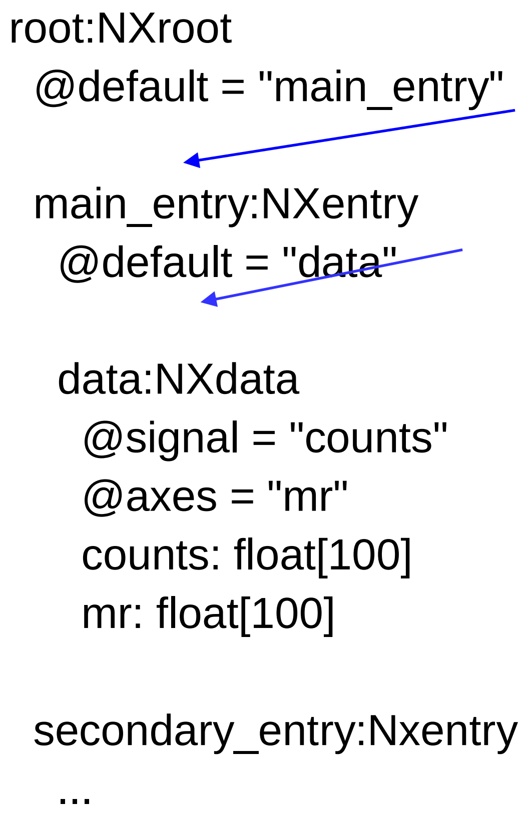
☑ Auto-size headers

☑ Popup to hide/show columns

Default columns

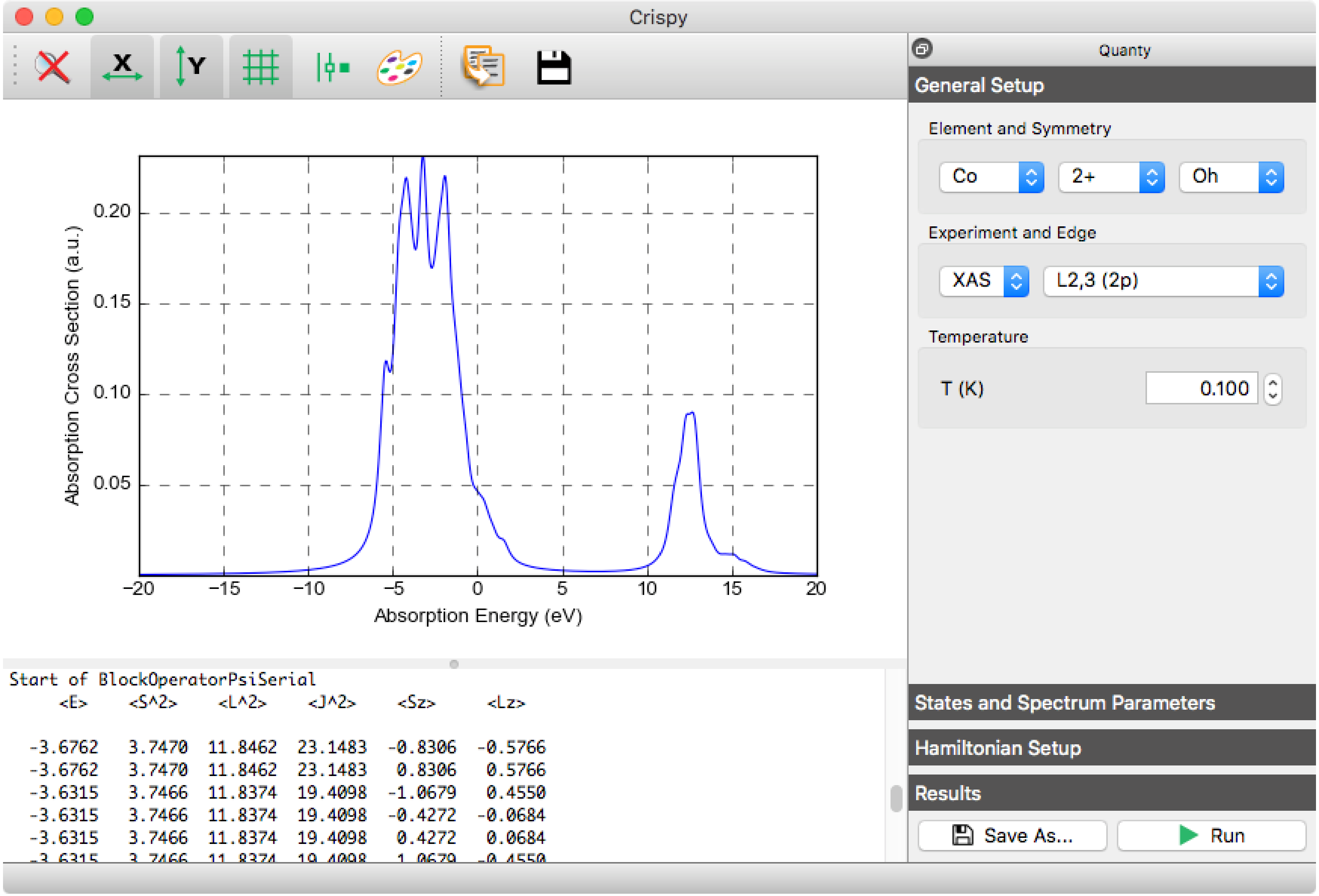
silx view

- Display *NXdata* view when viewing a *NXentry* or a *NXroot* group defining a @default attribute pointing to a valid *NXdata* group.



```
root:NXroot
  @default = "main_entry"

  main_entry:NXentry
    @default = "data"

    data:NXdata
      @signal = "counts"
      @axes = "mr"
      counts: float[100]
      mr: float[100]

  secondary_entry:Nxentry
    ...
```

Applications - Crispy



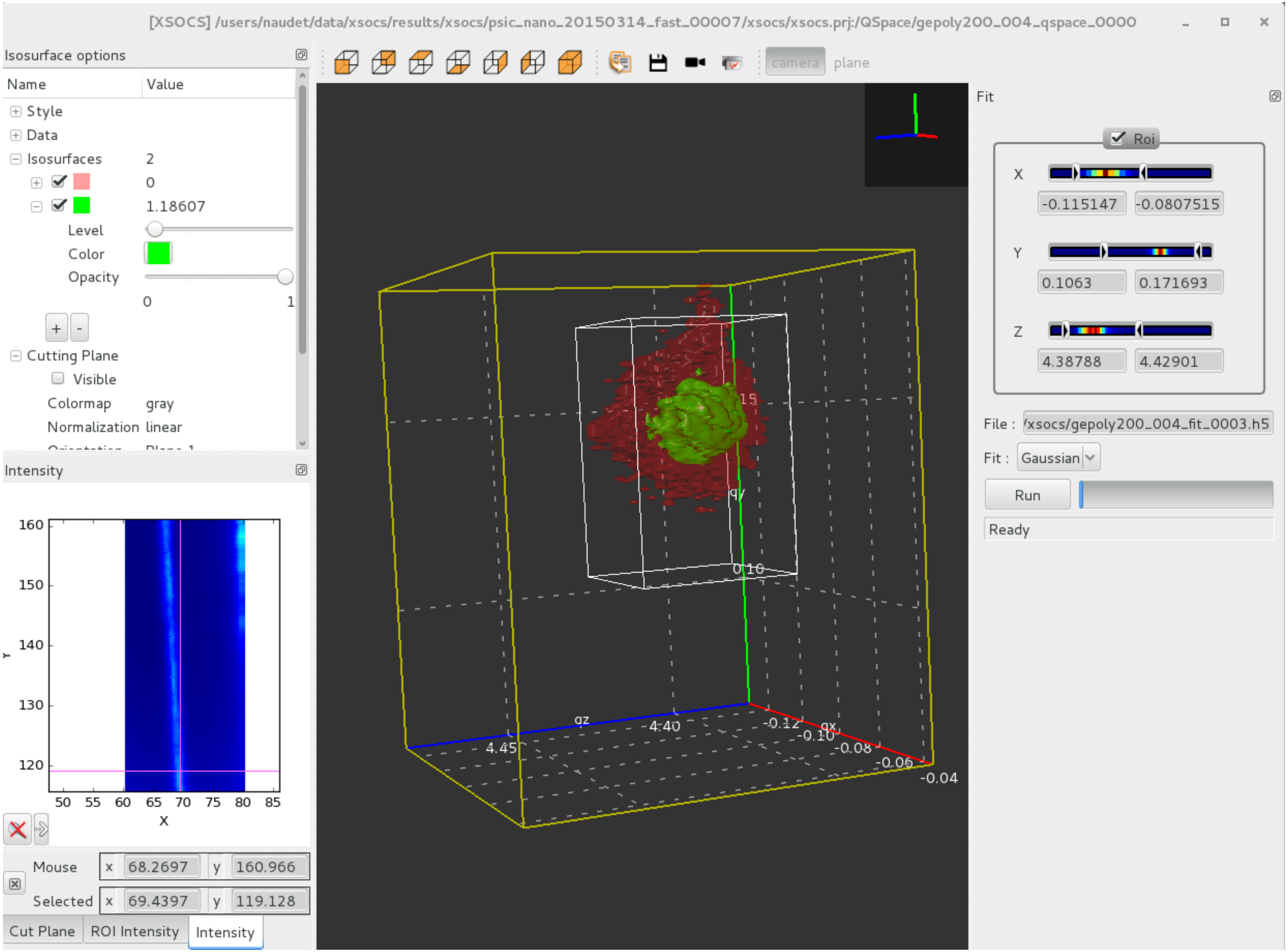
Element and Symmetry: Co, 2+, Oh

Experiment and Edge: XAS, L2,3 (2p)

Temperature: T (K) 0.100

```
Start of BlockOperatorPsiSerial
       <E>      <S^2>     <L^2>     <J^2>      <Sz>      <Lz>

   -3.6762    3.7470   11.8462   23.1483   -0.8306   -0.5766
   -3.6762    3.7470   11.8462   23.1483    0.8306    0.5766
   -3.6315    3.7466   11.8374   19.4098   -1.0679    0.4550
   -3.6315    3.7466   11.8374   19.4098   -0.4272   -0.0684
   -3.6315    3.7466   11.8374   19.4098    0.4272    0.0684
   -3.6315    3.7466   11.8374   19.4098    1.0679   -0.4550
```

Applications - XSOCS



[XSOCS] /users/naudet/data/xsocs/results/xsocs/psic_nano_20150314_fast_00007/xsocs/xsocs.prj:/QSpace/gepoly200_004_qspace_0000

Isosurface options

Name	Value
⊞ Style	
⊞ Data	
⊟ Isosurfaces	2
⊞ ☑ 🟥	0
⊟ ☑ 🟩	1.18607
Level	
Color	🟩
Opacity	0 – 1
⊟ Cutting Plane	
☐ Visible	
Colormap	gray
Normalization	linear
Orientation	Plane 1

Fit

☑ Roi

X: -0.115147 -0.0807515

Y: 0.1063 0.171693

Z: 4.38788 4.42901

File : /xsocs/gepoly200_004_fit_0003.h5

Fit : Gaussian

Run

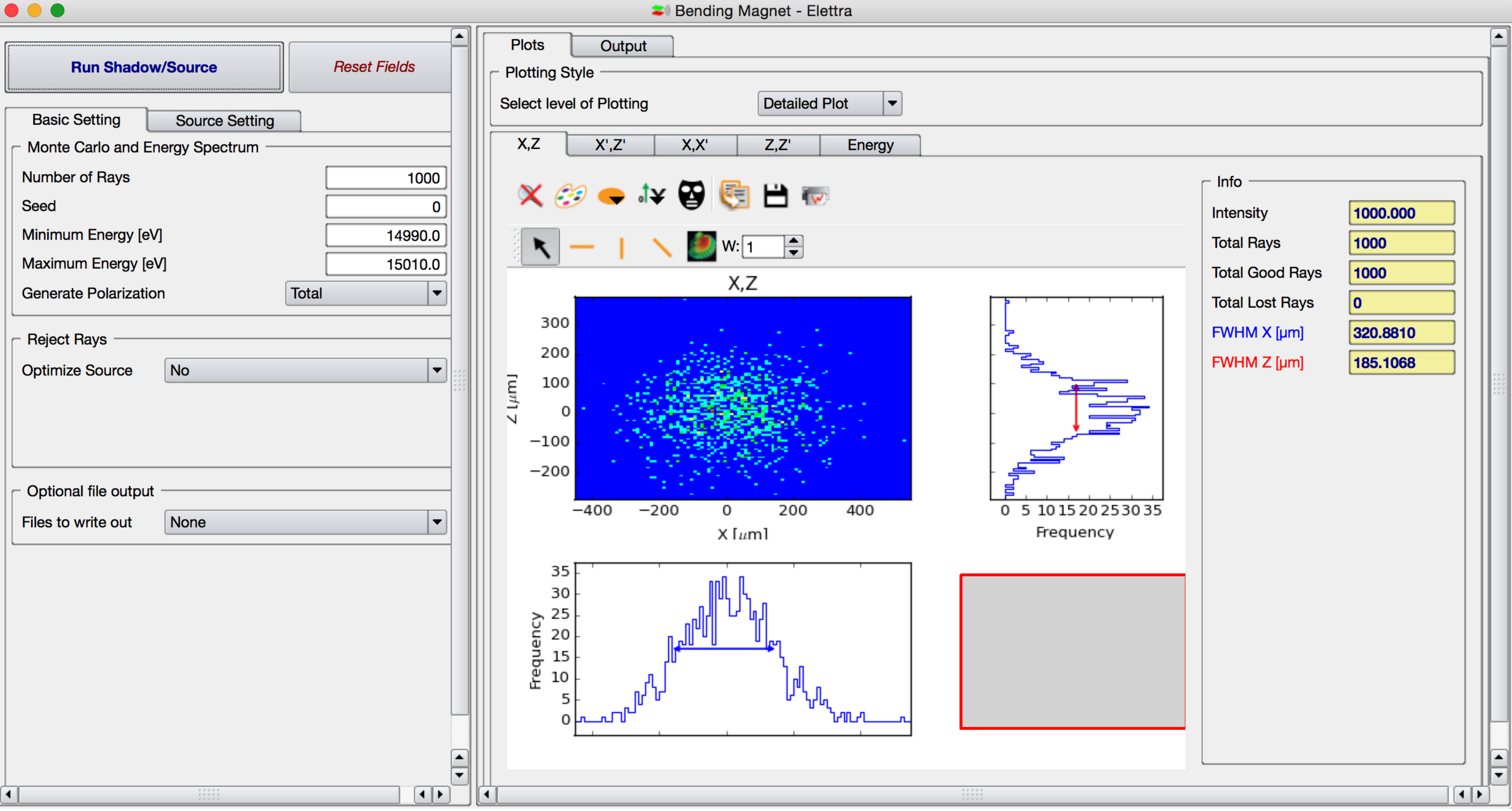
Ready

Intensity

Mouse x 68.2697 y 160.966

Selected x 69.4397 y 119.128

Applications - OASYS



Bending Magnet - Elettra

Monte Carlo and Energy Spectrum

Number of Rays	1000
Seed	0
Minimum Energy [eV]	14990.0
Maximum Energy [eV]	15010.0
Generate Polarization	Total

Reject Rays

Optimize Source: No

Optional file output

Files to write out: None

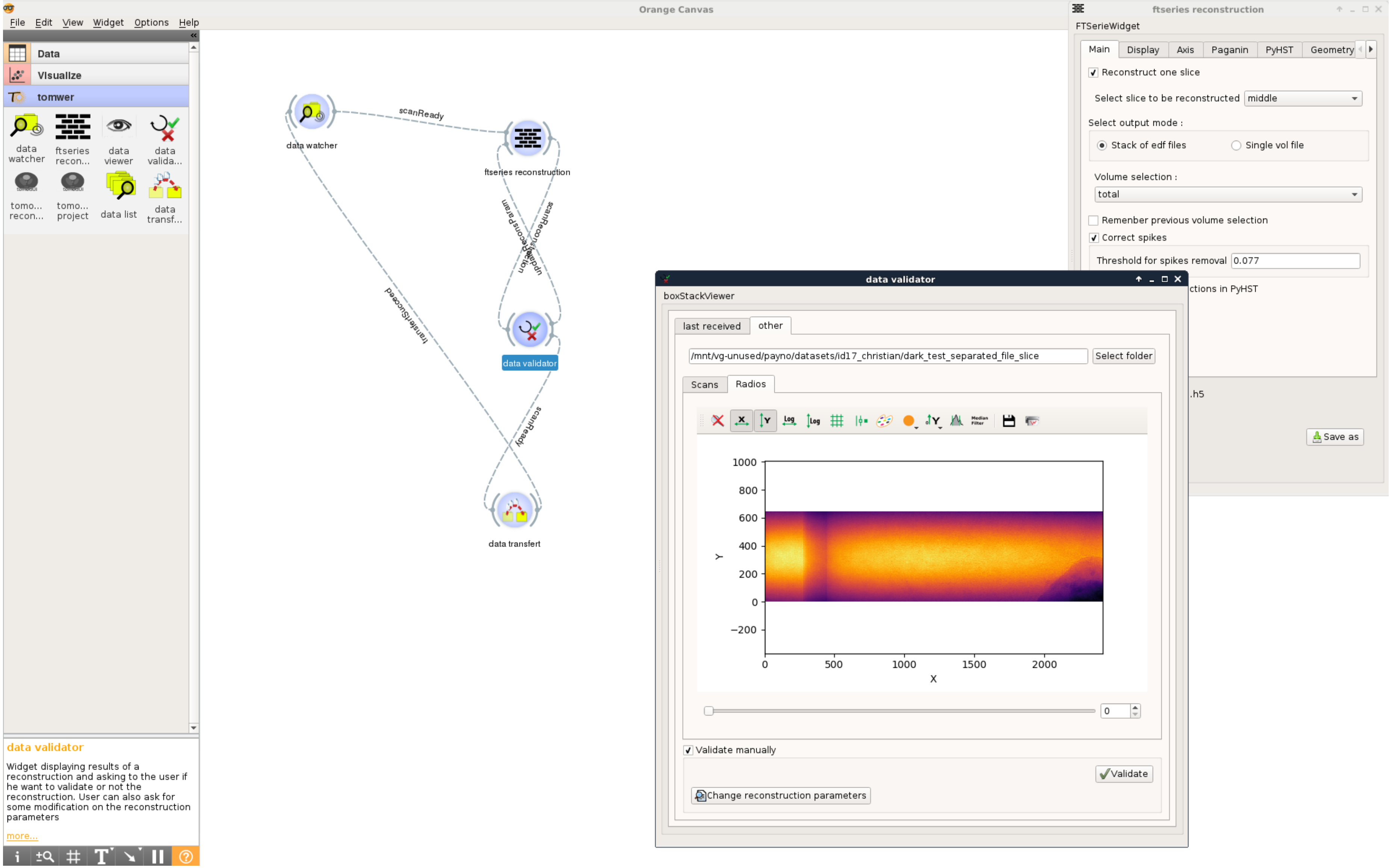
Plotting Style

Select level of Plotting: Detailed Plot

Info

Intensity	1000.000
Total Rays	1000
Total Good Rays	1000
Total Lost Rays	0
FWHM X [μm]	320.8810
FWHM Z [μm]	185.1068

Applications – Tomography Workflows



☑ Reconstruct one slice

Select slice to be reconstructed: middle

Select output mode :

● Stack of edf files ○ Single vol file

Volume selection : total

□ Remenber previous volume selection

☑ Correct spikes

Threshold for spikes removal 0.077

...ctions in PyHST

...h5

/mnt/vg-unused/payno/datasets/id17_christian/dark_test_separated_file_slice

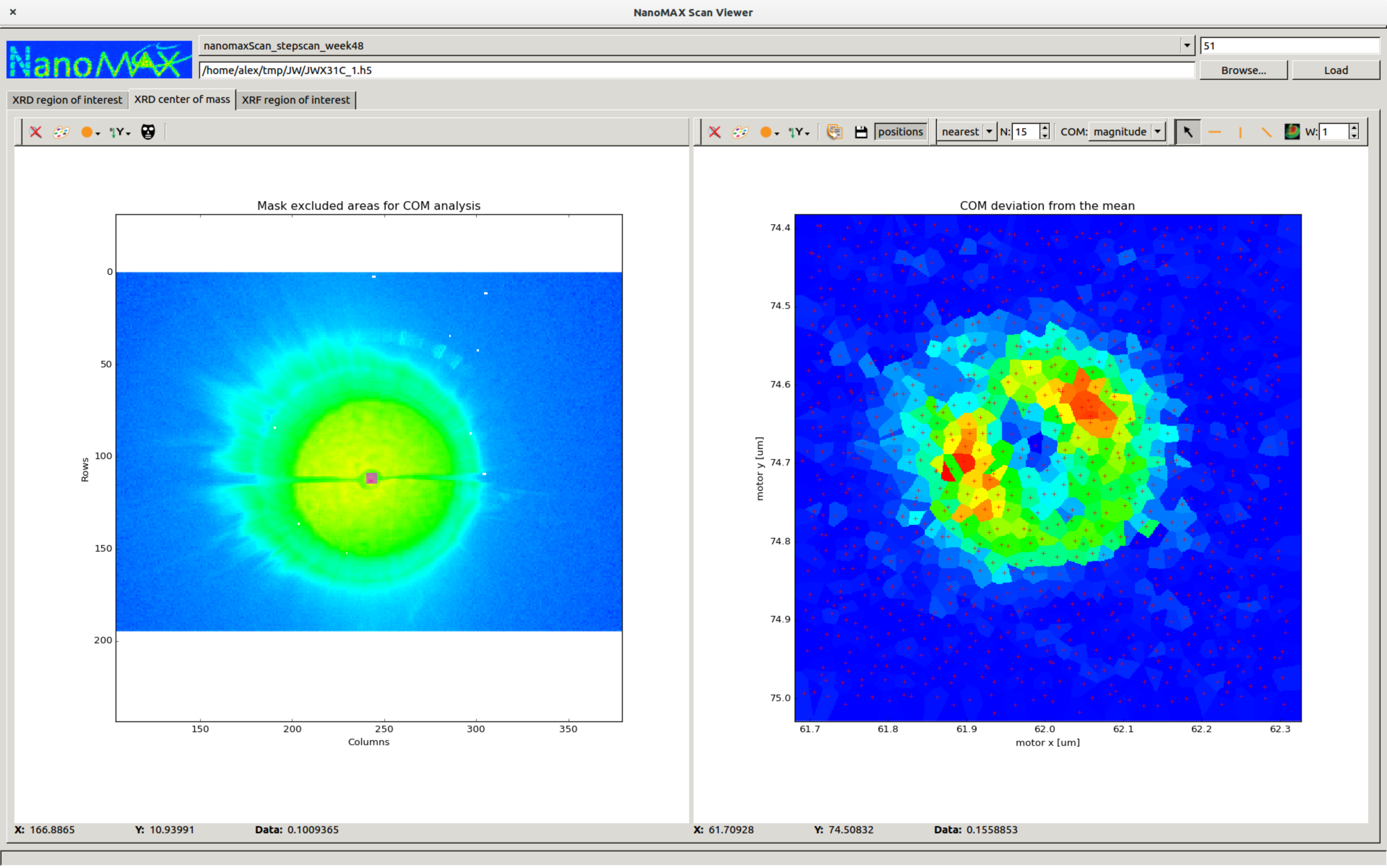
☑ Validate manually

data validator

Widget displaying results of a reconstruction and asking to the user if he want to validate or not the reconstruction. User can also ask for some modification on the reconstruction parameters

more...

Applications – Nanomax@Max IV



NanoMAX Scan Viewer

nanomaxScan_stepscan_week48

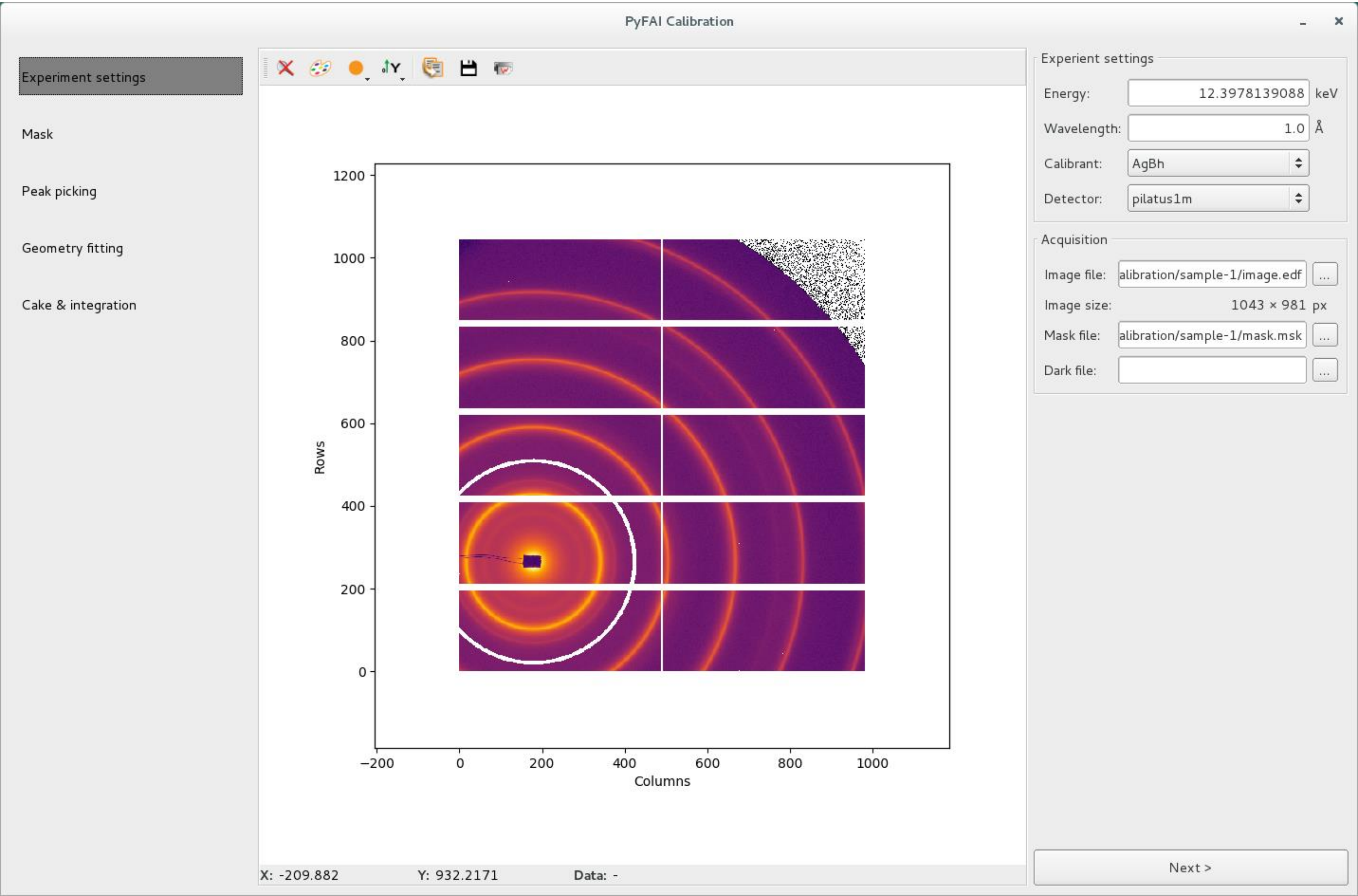
51

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

X: 166.8865 Y: 10.93991 Data: 0.1009365

X: 61.70928 Y: 74.50832 Data: 0.1558853

pyFAI Calibration - Settings



PyFAI Calibration

Experient settings

Energy: 12.3978139088 keV

Wavelength: 1.0 Å

Calibrant: AgBh

Detector: pilatus1m

Acquisition

Image file: alibration/sample-1/image.edf ...

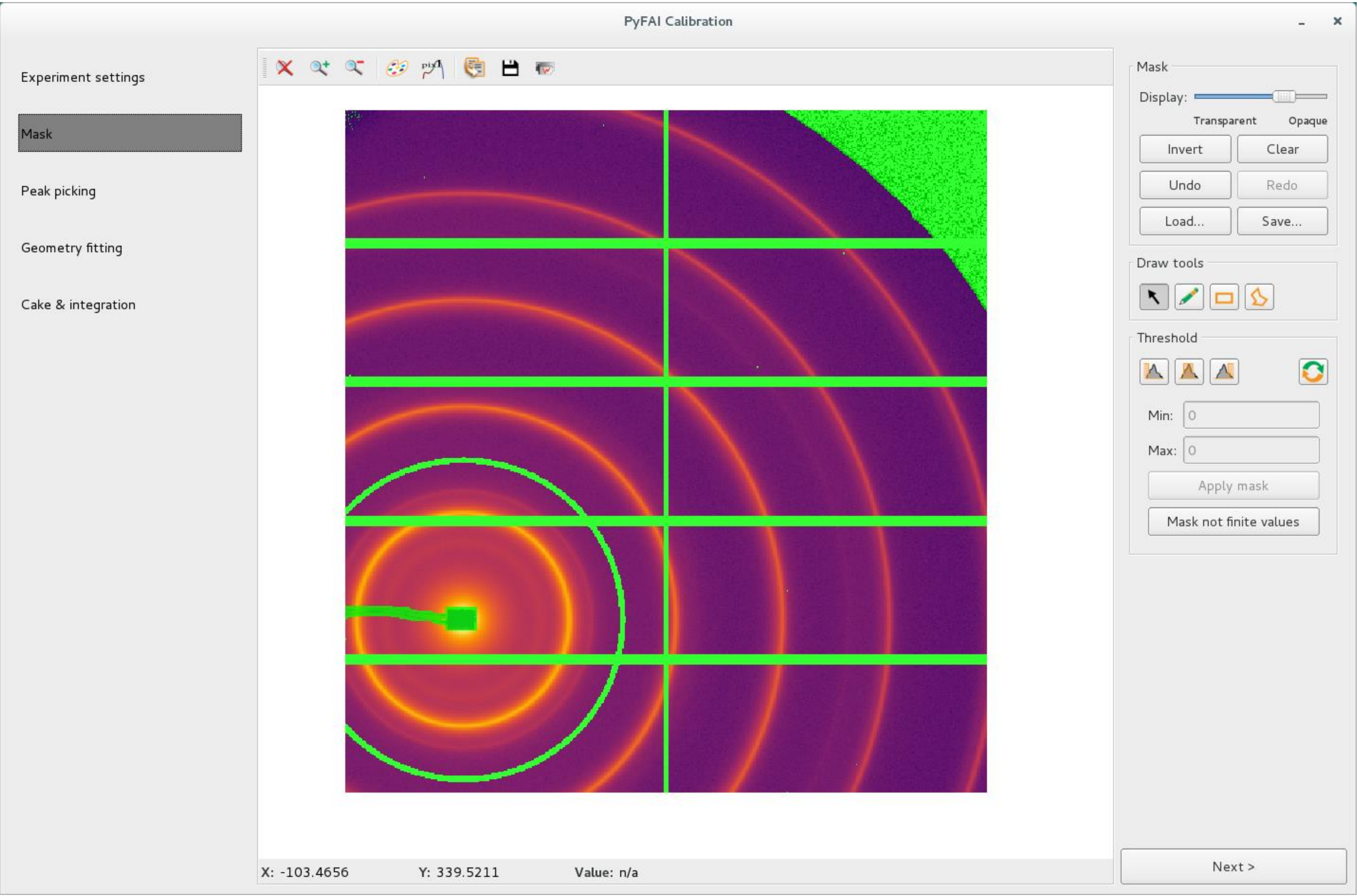
Image size: 1043 × 981 px

Mask file: alibration/sample-1/mask.msk ...

Dark file: ...

X: -209.882 Y: 932.2171 Data: -

pyFAI Calibration - Mask

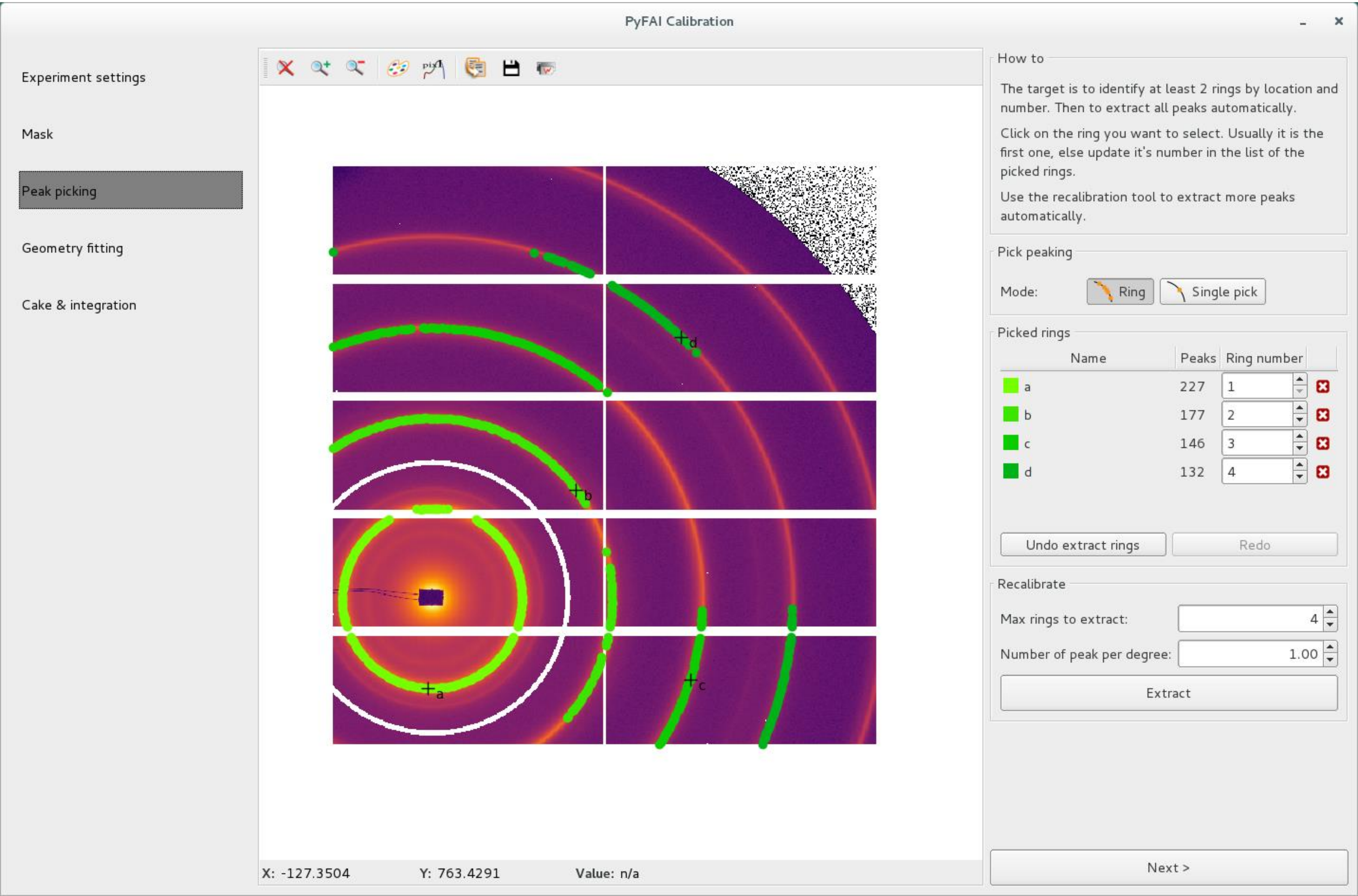


Min: 0

Max: 0

X: -103.4656 Y: 339.5211 Value: n/a

pyFAI Calibration – Peak Picking



How to

The target is to identify at least 2 rings by location and number. Then to extract all peaks automatically.

Click on the ring you want to select. Usually it is the first one, else update it's number in the list of the picked rings.

Use the recalibration tool to extract more peaks automatically.

Pick peaking

Mode: Ring | Single pick

Picked rings

	Name	Peaks	Ring number
🟩	a	227	1
🟩	b	177	2
🟩	c	146	3
🟩	d	132	4

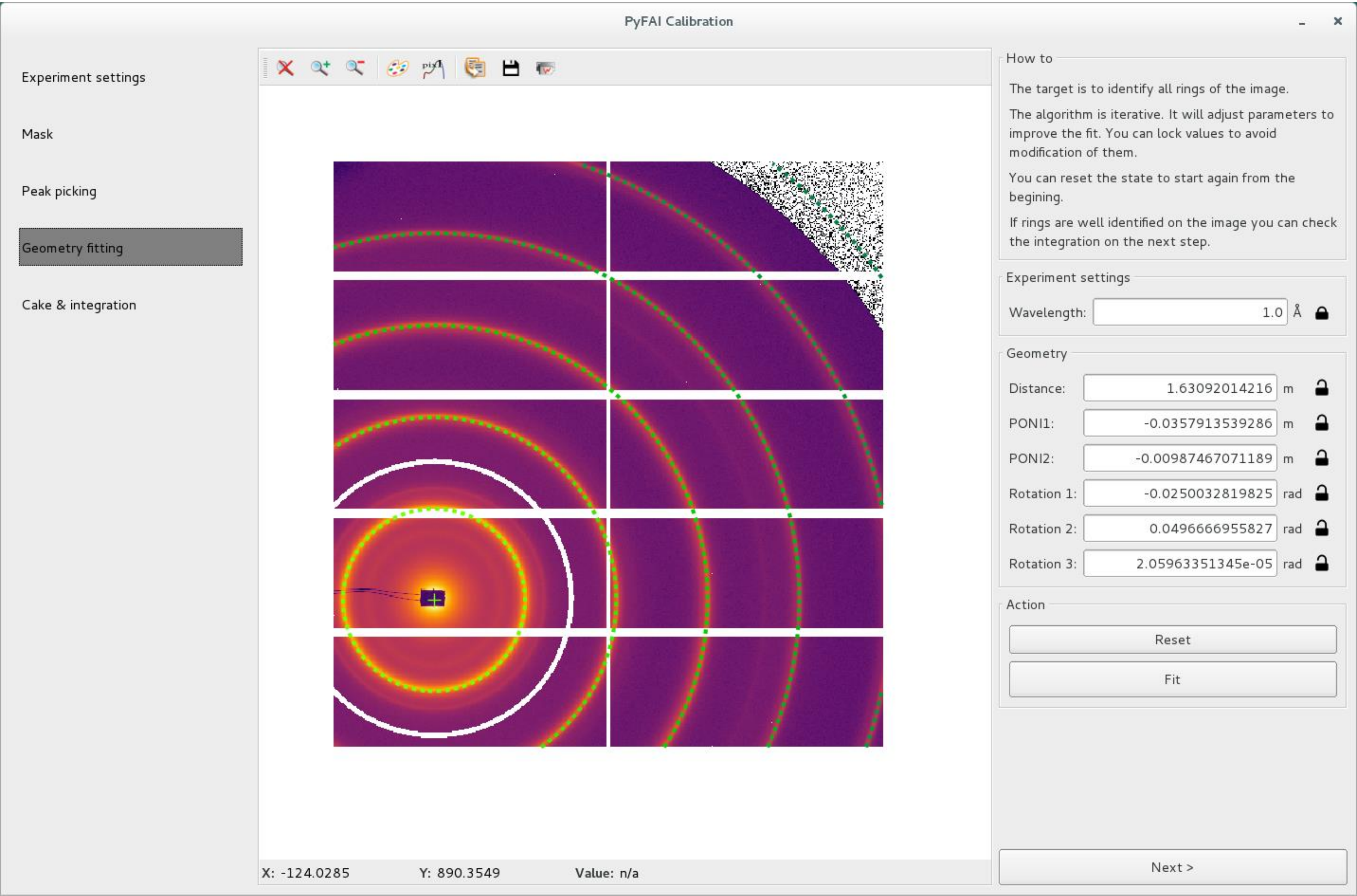
Recalibrate

Max rings to extract: 4

Number of peak per degree: 1.00

X: -127.3504 Y: 763.4291 Value: n/a

pyFAI Calibration – Geometry Fitting



PyFAI Calibration

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

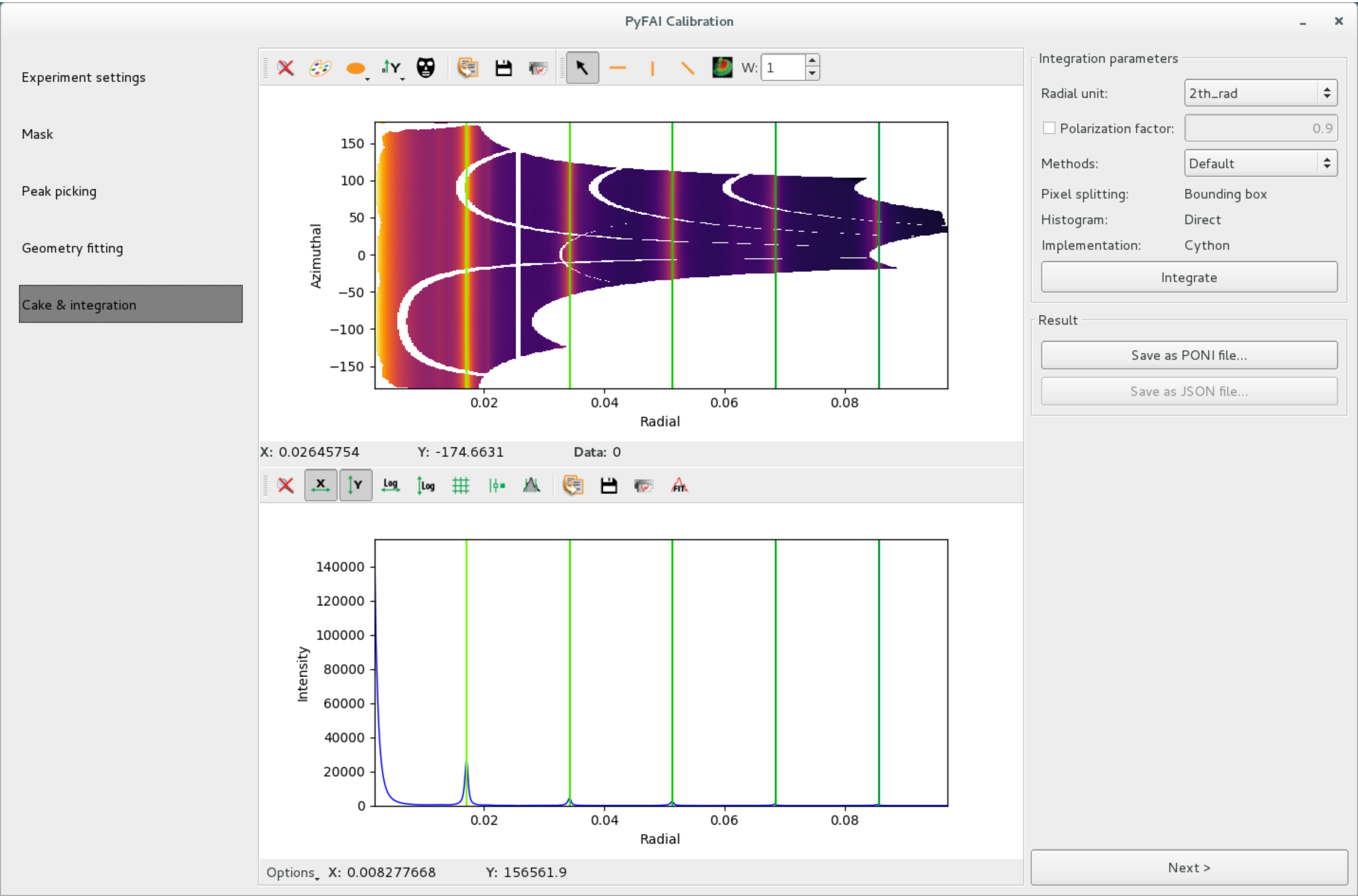
Rotation 2: 0.0496666955827 rad

Rotation 3: 2.05963351345e-05 rad

Action

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

pyFAI Calibration – Cake and Integrtion



PyFAI Calibration

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

X: 0.02645754 Y: -174.6631 Data: 0

Options X: 0.008277668 Y: 156561.9

Integration parameters

Radial unit: 2th_rad

□ Polarization factor: 0.9

Methods: Default

Pixel splitting: Bounding box

Histogram: Direct

Implementation: Cython

Integrate

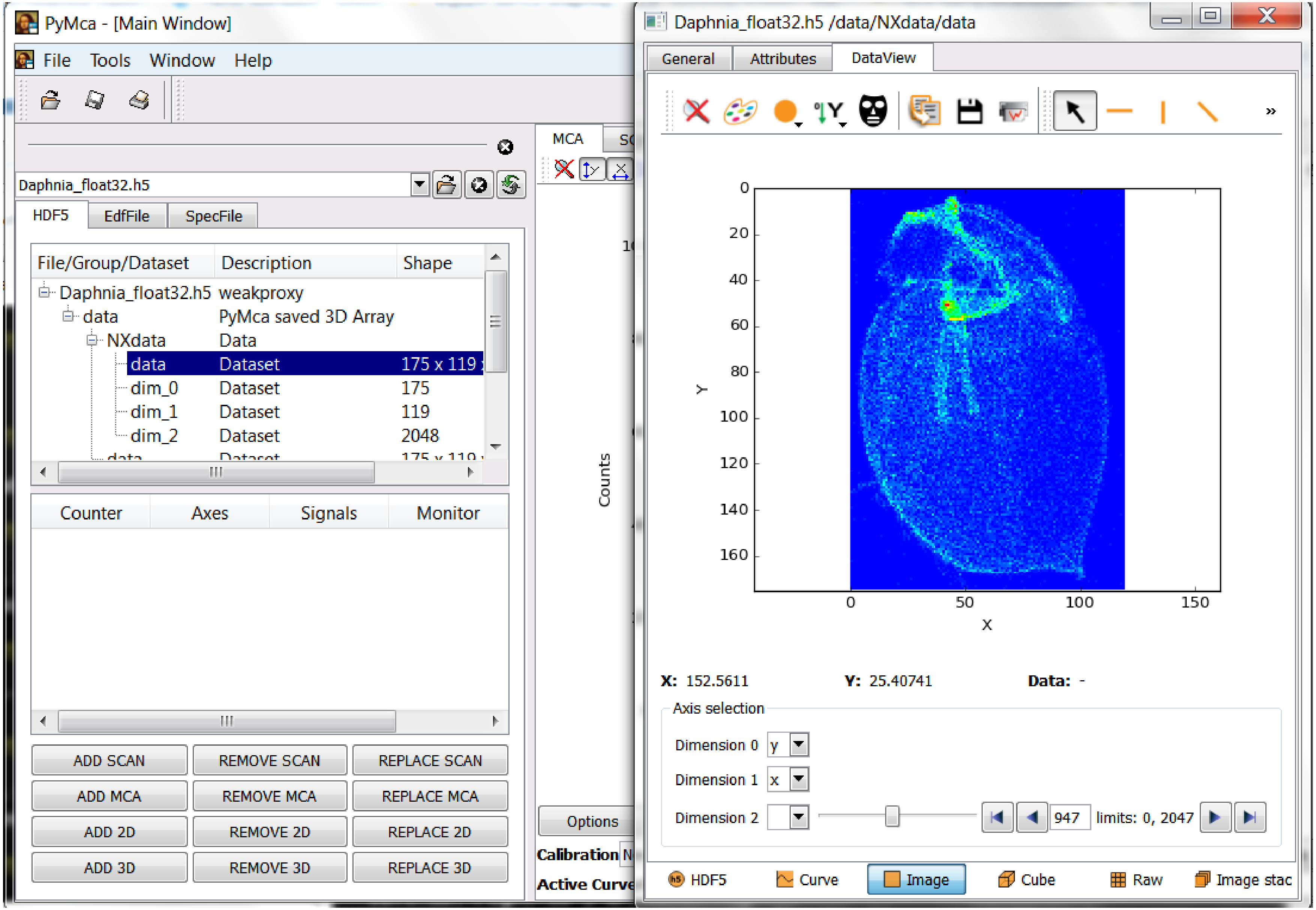
Result

Save as PONI file...

Save as JSON file...

Next >

PyMca - silx DataViewer replacing PyMca TableView



Daphnia_float32.h5

File/Group/Dataset	Description	Shape
Daphnia_float32.h5	weakproxy	
data	PyMca saved 3D Array	
NXdata	Data	
data	Dataset	175 x 119
dim_0	Dataset	175
dim_1	Dataset	119
dim_2	Dataset	2048
data	Dataset	175 x 119

X: 152.5611 **Y:** 25.40741 **Data:** -

Axis selection

Dimension 0 y

Dimension 1 x

Dimension 2 947 limits: 0, 2047

- This release
  - I/O dialogs, h5pyd support, data URLs
  - silx view full support of NXdata groups
  - silx convert as generic merge tool
  - Plot3D: SceneGraph and SceneItems.
  - OpenCL: image processing, byte offset…

- 2018
  - SceneGraph interaction
  - Statistics in Curves, Images, Volumes
  - 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 0.8.0 version
- For this code camp we’ll use 0.9.0a, 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/