4th Code Camp
May 2, 2017

THIS TALK

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

Plot: OpenGL Rendering

Add an OpenGL rendering backend to silx.gui.plot widgets:

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

- 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()
```

Plot: OpenGL Rendering

Pending improvements:

- Visual improvements:
  - Proper curve dash rendering
  - Text display on High DPI screen with PyQt5
  - Scatter plot points size in device independent units.
- Add support for Qt >=5.4 OpenGL widget API
- Optimizations
- Refactoring: Share more code with silx.gui.plot3d
- More (automated) testing and Continuous Integration

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.

- In v0.5.0 it will return 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.))
```

Plot: Object API

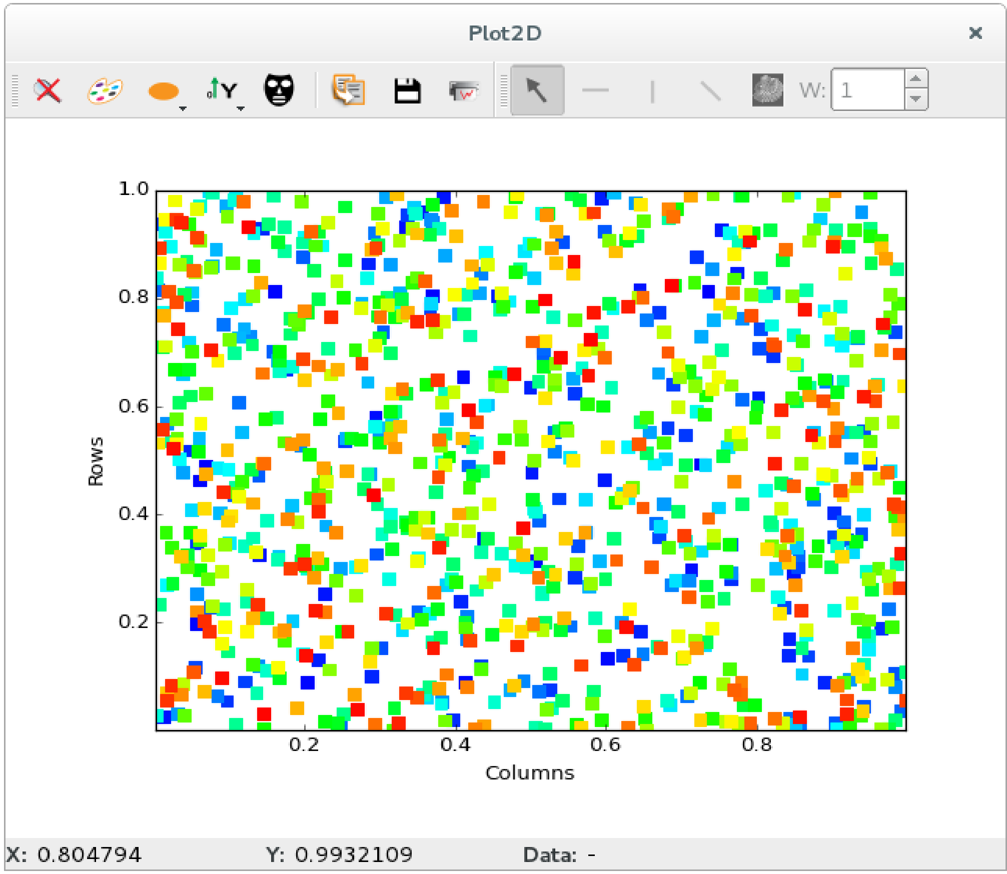
Pending improvements:

- Convert *dict* provided by Plot events to objects.
- Convert *dict* describing Plot colormap to objects.
- Add signals to Plot items objects.

Feedback on API welcome!

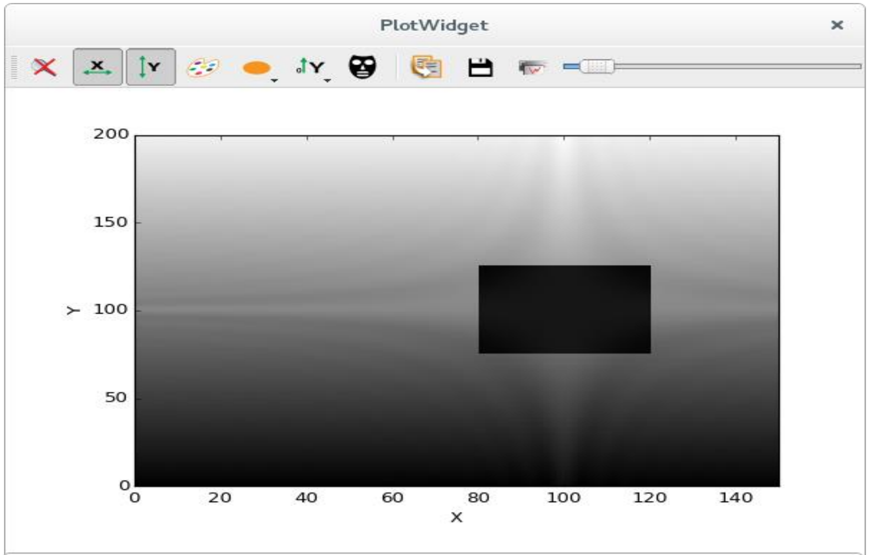
Scatter Objects

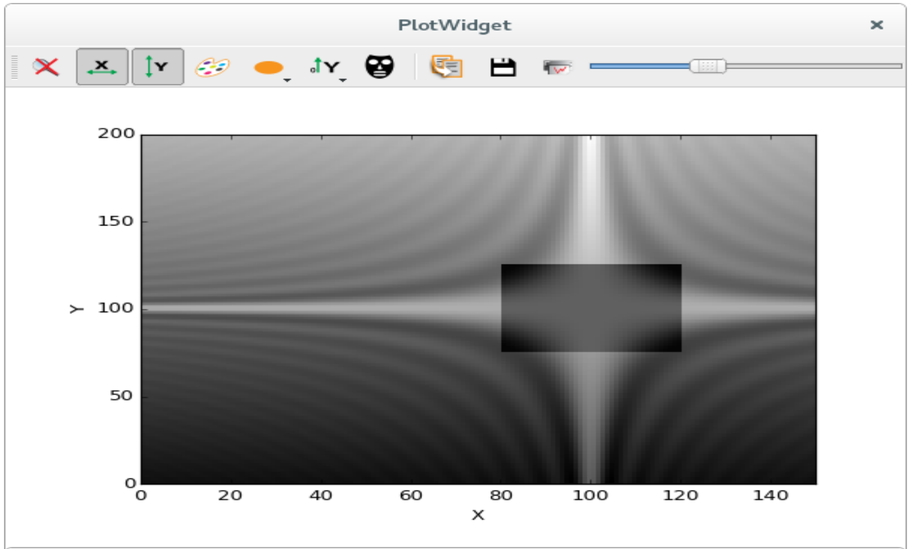
```
import numpy
import sys
from silx.gui import qt
from silx.gui.plot import Plot2D

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_())
```

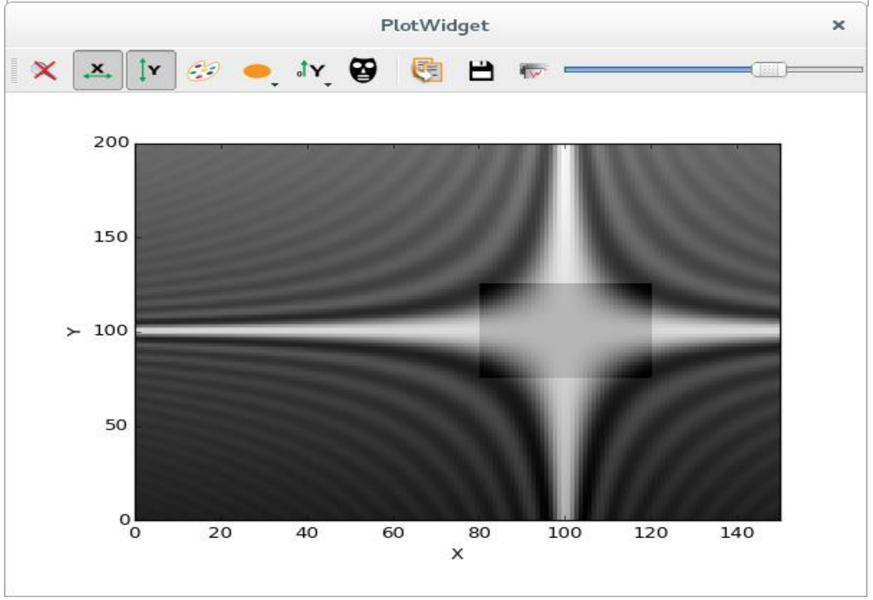


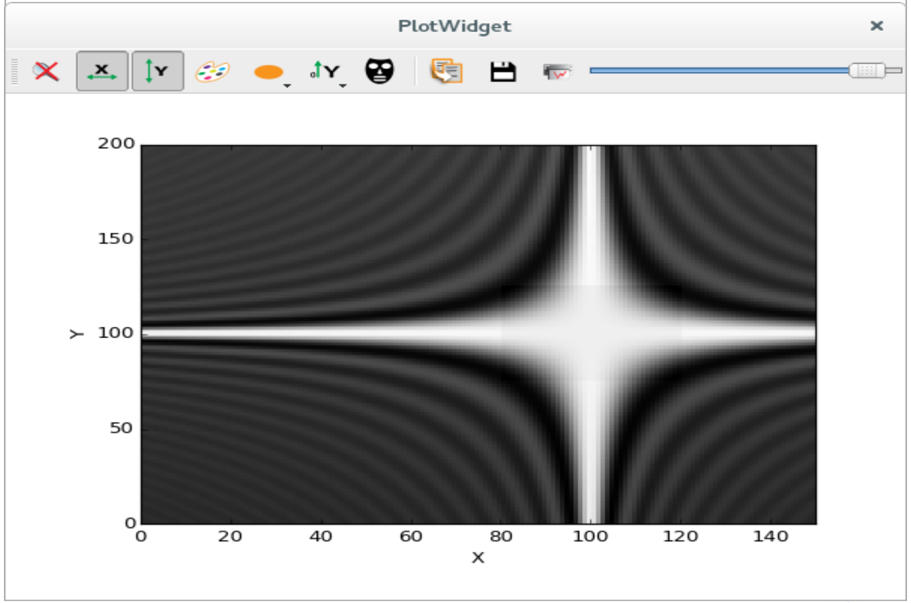
X: 0.804794 Y: 0.9932109 Data: -

Image/Scatter Transparency Slider









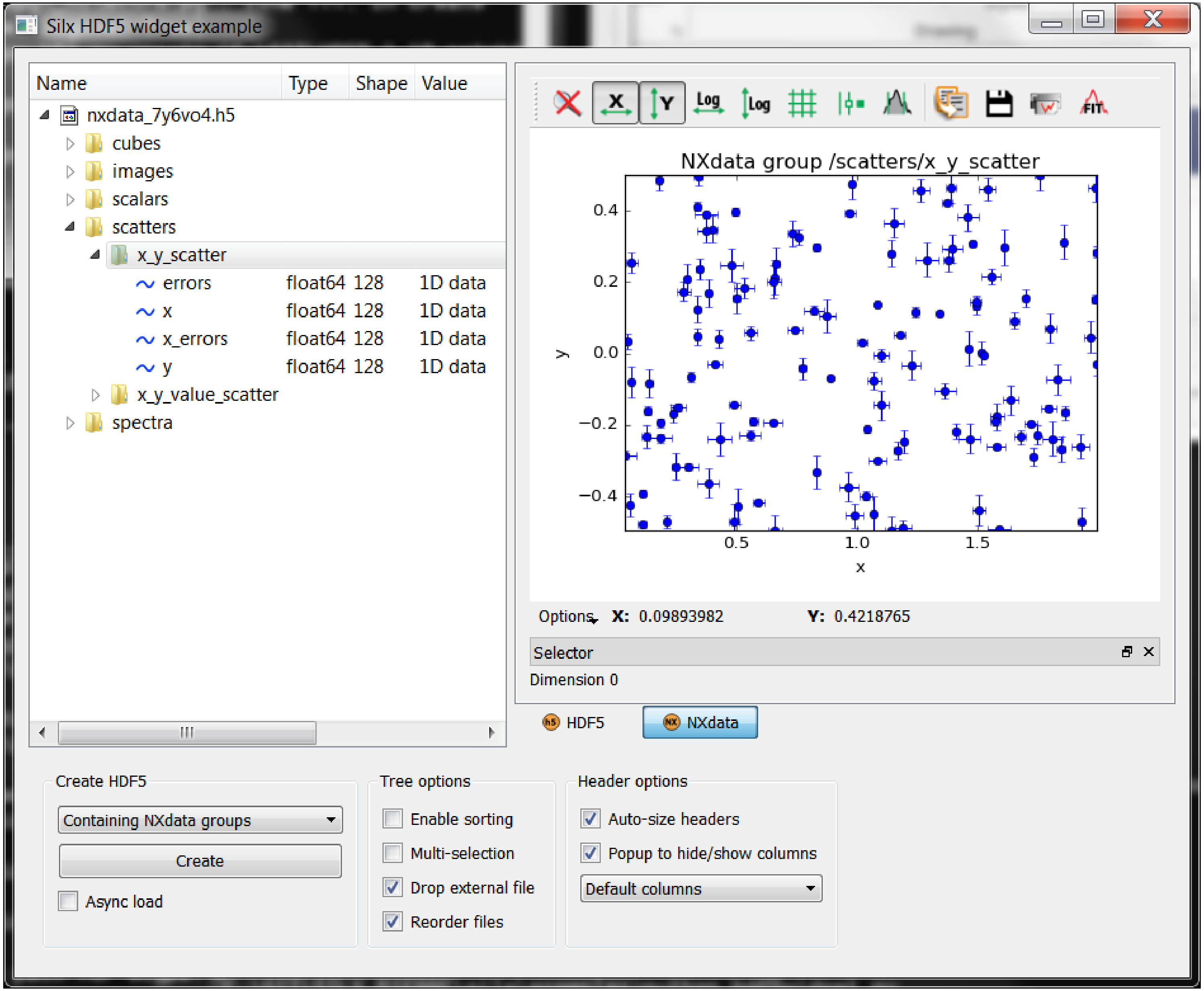
NXdataViewer

- 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)

NXdataViewer



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			

X: 0.09893982 Y: 0.4218765

Dimension 0

Create HDF5

Containing NXdata groups

☐ Async load

Tree options

☐ Enable sorting

☐ Multi-selection

☑ Drop external file

☑ Reorder files

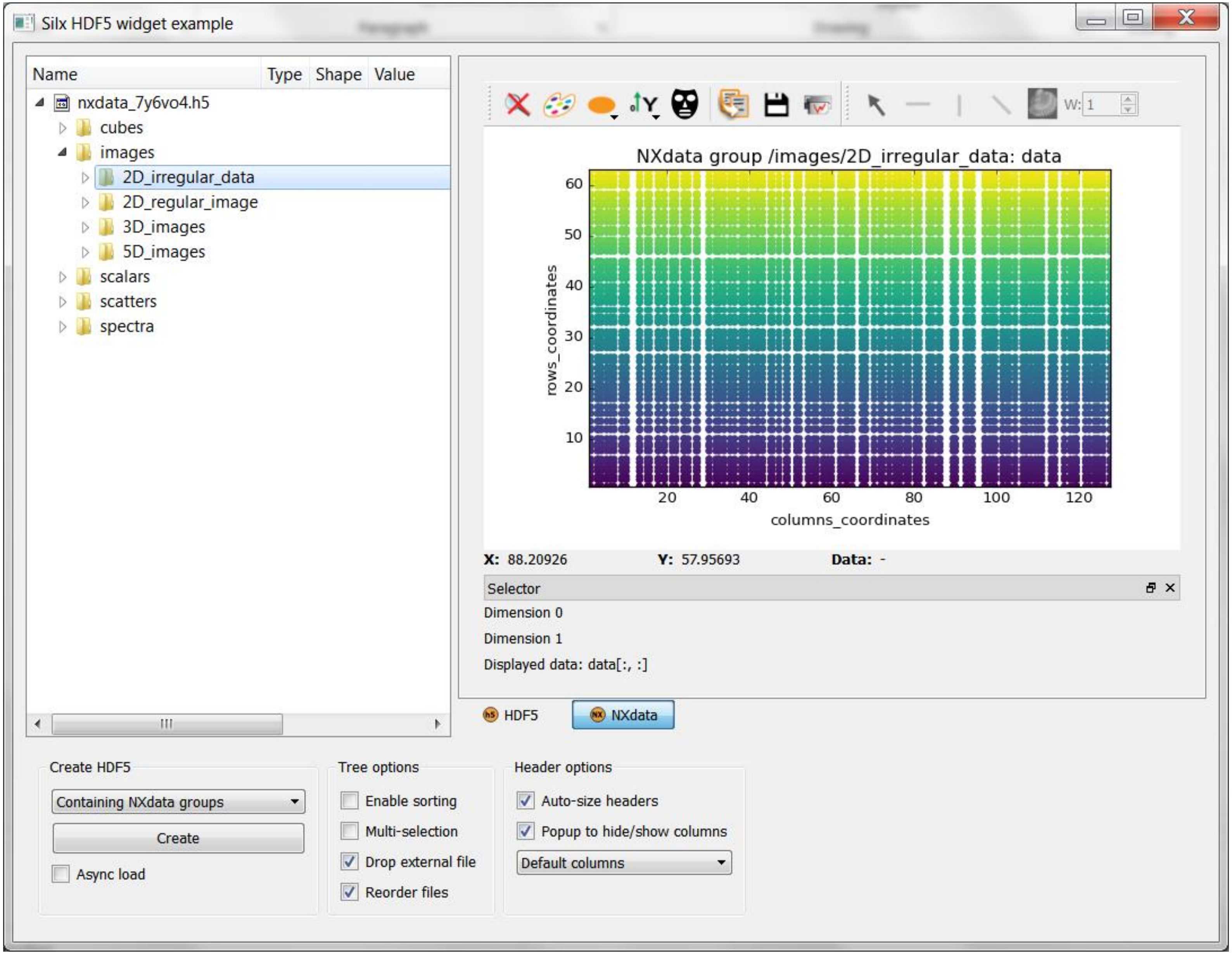
Header options

☑ Auto-size headers

☑ Popup to hide/show columns

Default columns

NXdataViewer



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

Selector

Dimension 0

Dimension 1

Displayed data: data[:, :]

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

Packaging

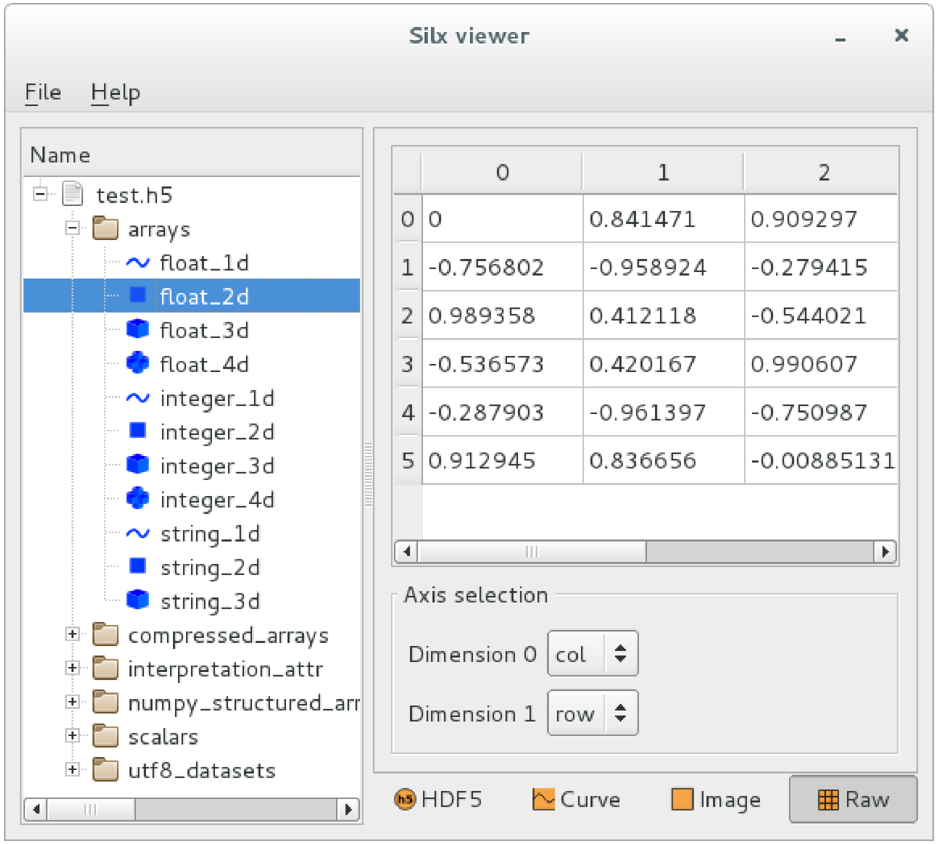
- Introduce a generic launcher
  - Linux / Mac / Windows
  - Can be run as silx (silx.exe) command line
  - Or as a python package (python -m silx)
- A single package for Debian 7
  - Containing Python 2 library and launcher
- A new package for Debian 8
  - silx package containing the launcher (Python 3)
- Packaging for Debian 9
  - silx, python-silx, python3-silx...

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*



	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

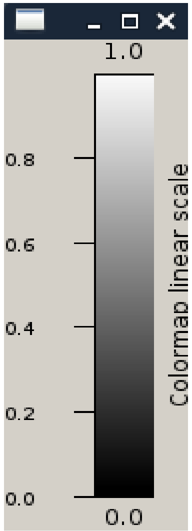
Dimension 1: row

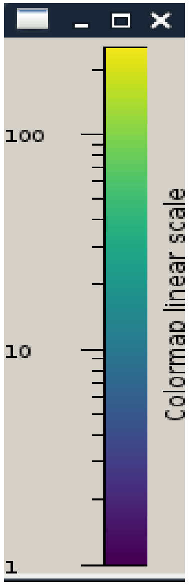
Color Bar

silx.gui.plot.ColorBar

Show colormap information (log scale, min, max …)
On mouse move display values associated to the color

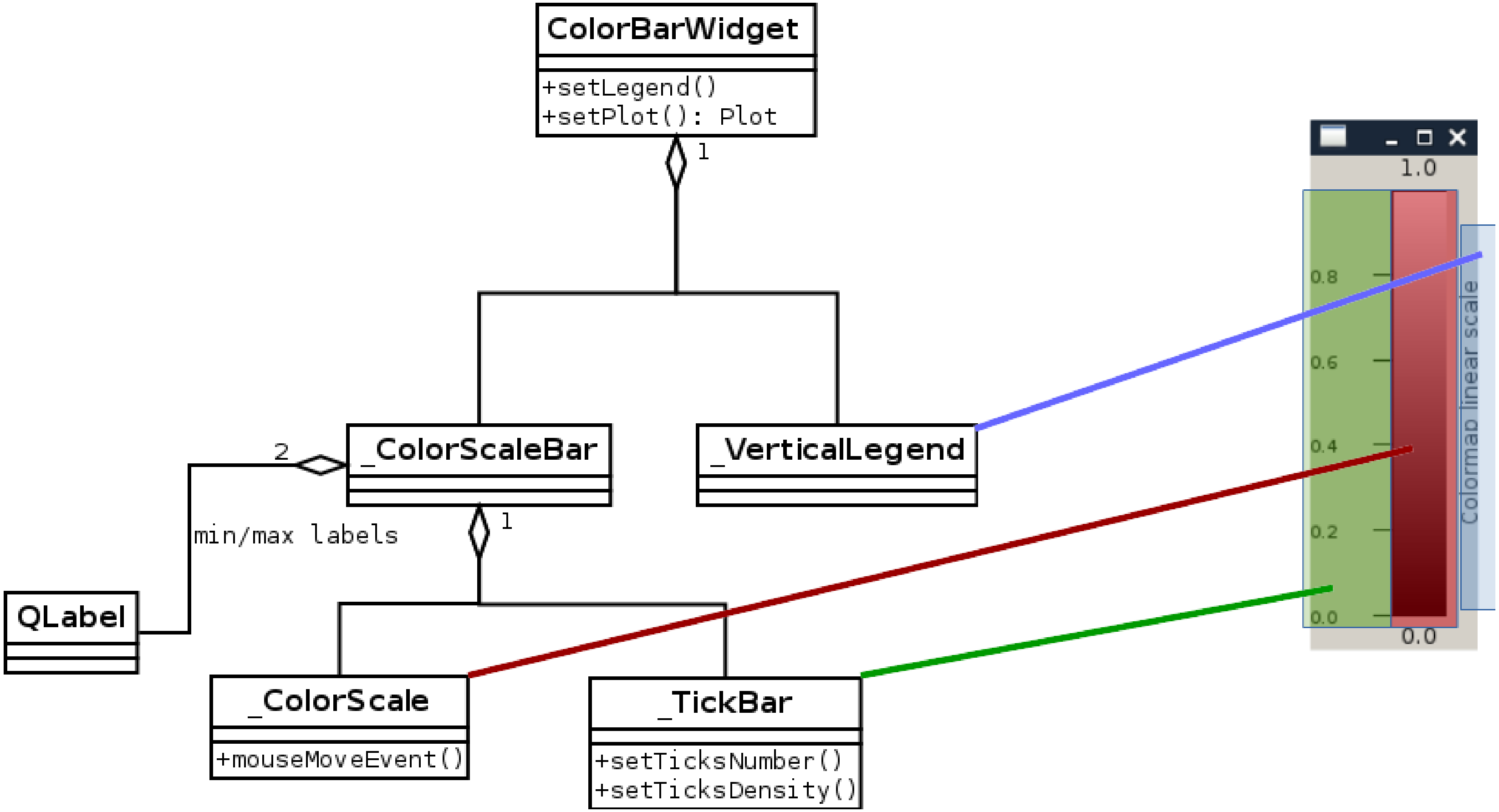
```
import numpy
import sys
from silx.gui import qt
from silx.gui.plot.PlotWindow import Plot2D
from silx.gui.plot.ColorBar import ColorBarWidget

image = numpy.arange(100).reshape(10, 10)/99
app = qt.QApplication([])
plot = Plot2D()

plot.addImage(image)
colorbar = ColorBarWidget(parent=None, plot=plot)
colorbar.setLegend('Colormap linear scale')
colorbar.show()


image = numpy.arange(200).reshape(10, 20)
colorbar.getColorScaleBar().setMinMaxVisible(False)
clm = plot.getDefaultColormap()
clm['normalization'] = 'log'
clm['name'] = 'viridis'
plot.addImage(data=image, colormap=clm, legend='toto')
plot.setActiveImage('toto')

sys.exit(app.exec_())
```





Color Bar

silx.gui.plot.ColorBar



ColorBarWidget

+setLegend()
+setPlot(): Plot

1

_ColorScaleBar

_VerticalLegend

2

min/max labels

1

QLabel

_ColorScale

+mouseMoveEvent()

_TickBar

+setTicksNumber()
+setTicksDensity()

Median Filter (C++)

silx.math.medianfilter

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

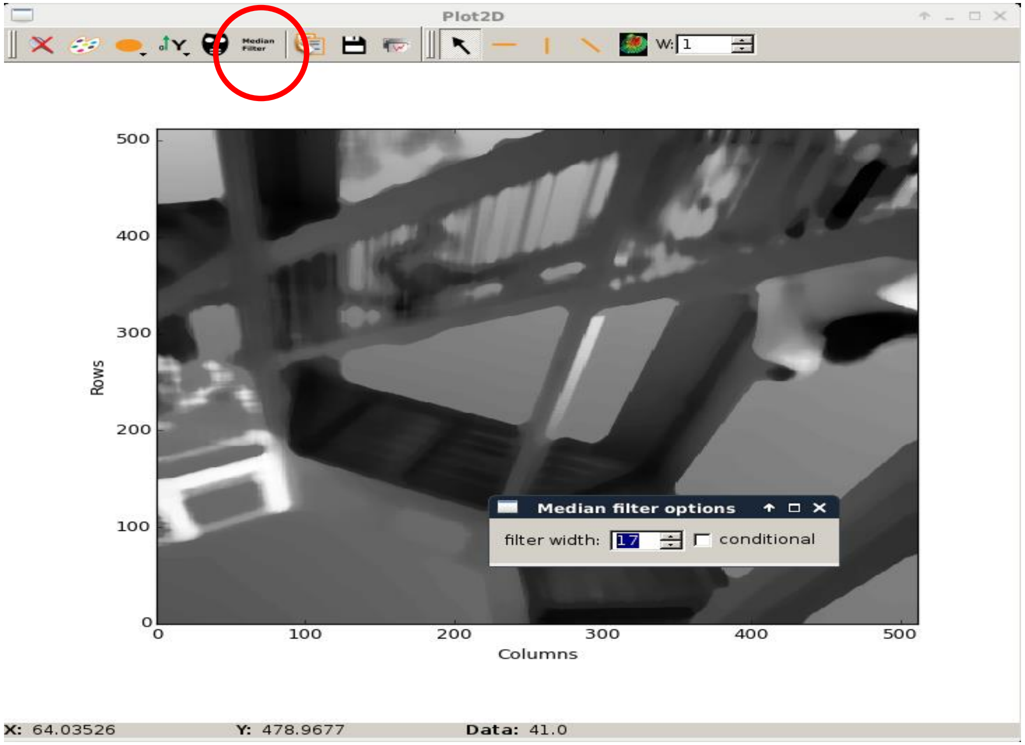
- 1D-2D median filter
  - data: 1D or 2D numpy array
  (specialized functions medfilt1d and medfilt2d available)
  - kernel_size int or tuple
  - Conditional if True apply conditional median filtering
   (apply only if pixel value is window minimum or maximum)

- Example:

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

Median Filter Option

silx.math.medianfilter



filter width: 17 □ conditional

X: 64.03526 Y: 478.9677 Data: 41.0

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

import scipy.misc
app=qt.QApplication([])
image=scipy.misc.ascent().astype('float32')

plotImage=Plot2D()
plotImage.addImage(image)
plotImage.getMedianFilter2DAction().setVisible(True)
plotImage.show()

sys.exit(app.exec_())
```

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)) → 5.8s

import silx.image
%timeit silx.image.medfilt2d(img, (55,55)) → 8.6s          (issue #773)

from silx.opencl import medifilt
%timeit medfilt.medfilt2d(img, (55,55)) → 2.4s
```

External Resources Manager

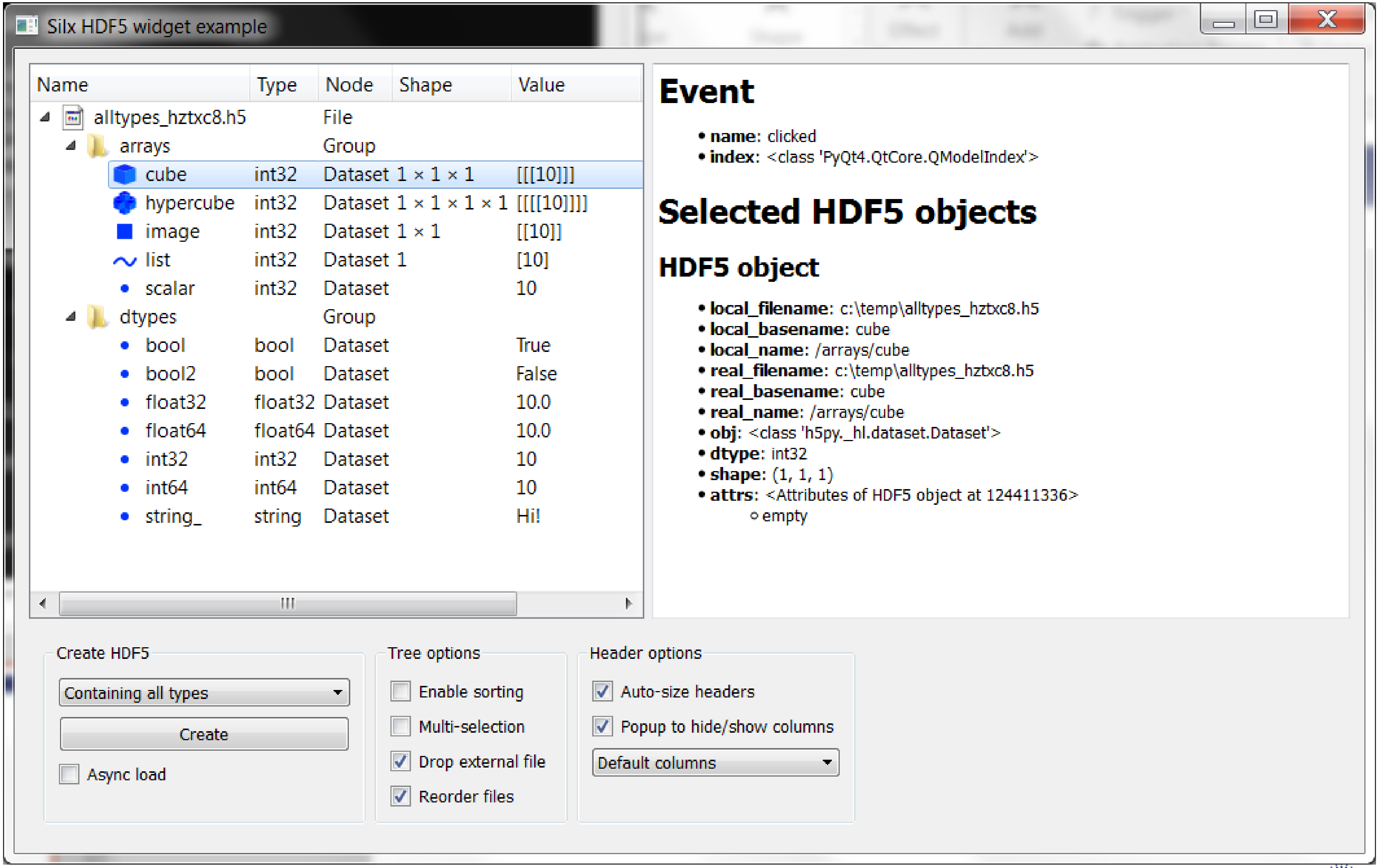
- Used in tests to download data as needed
- Used to create a temporary work-directory
- Can be re-used directly by other projects.

```
import silx.test.utils
print(silx.test.utils.utilstest.getfile("lena.png"))
        /tmp/silx_testdata_kieffer/lena.png
print(silx.test.utils.utilstest.tempdir)
        /tmp/silx_BHynBI_kieffer

import silx.resources as sr
erm = sr.ExternalResources("toto",“http://www.silx.org/pub/pyFAI/testimages/")
print(erm.getfile("Pilatus1M.edf"))
        /tmp/toto_testdata_kieffer/Pilatus1M.edf
```

CURRENT STATUS (0.5.0A)

silx.io Input/Output

- Read ALL files using an API similar to the h5py one
- Convert SPEC files to ESRF HDF5 NeXus implementation
- Dump dictionaries to files in HDF5, json or ini format
- Use FabIO for image formats other than TIFF
- Unified widget to deal with all data formats
- Generic data viewer (*silx view*)



Silx HDF5 widget example

Name	Type	Node	Shape	Value
alltypes_hztxc8.h5		File		
arrays		Group		
cube	int32	Dataset	1 × 1 × 1	[[[10]]]
hypercube	int32	Dataset	1 × 1 × 1 × 1	[[[[10]]]]
image	int32	Dataset	1 × 1	[[10]]
list	int32	Dataset	1	[10]
scalar	int32	Dataset		10
dtypes		Group		
bool	bool	Dataset		True
bool2	bool	Dataset		False
float32	float32	Dataset		10.0
float64	float64	Dataset		10.0
int32	int32	Dataset		10
int64	int64	Dataset		10
string_	string	Dataset		Hi!

Event

- **name**: clicked
- **index**: <class 'PyQt4.QtCore.QModelIndex'>

Selected HDF5 objects

HDF5 object

- **local_filename**: c:\temp\alltypes_hztxc8.h5
- **local_basename**: cube
- **local_name**: /arrays/cube
- **real_filename**: c:\temp\alltypes_hztxc8.h5
- **real_basename**: cube
- **real_name**: /arrays/cube
- **obj**: <class 'h5py._hl.dataset.Dataset'>
- **dtype**: int32
- **shape**: (1, 1, 1)
- **attrs**: <Attributes of HDF5 object at 124411336>
  - empty

Create HDF5

Containing all types

□ Async load

Tree options

□ Enable sorting

□ Multi-selection

☑ Drop external file

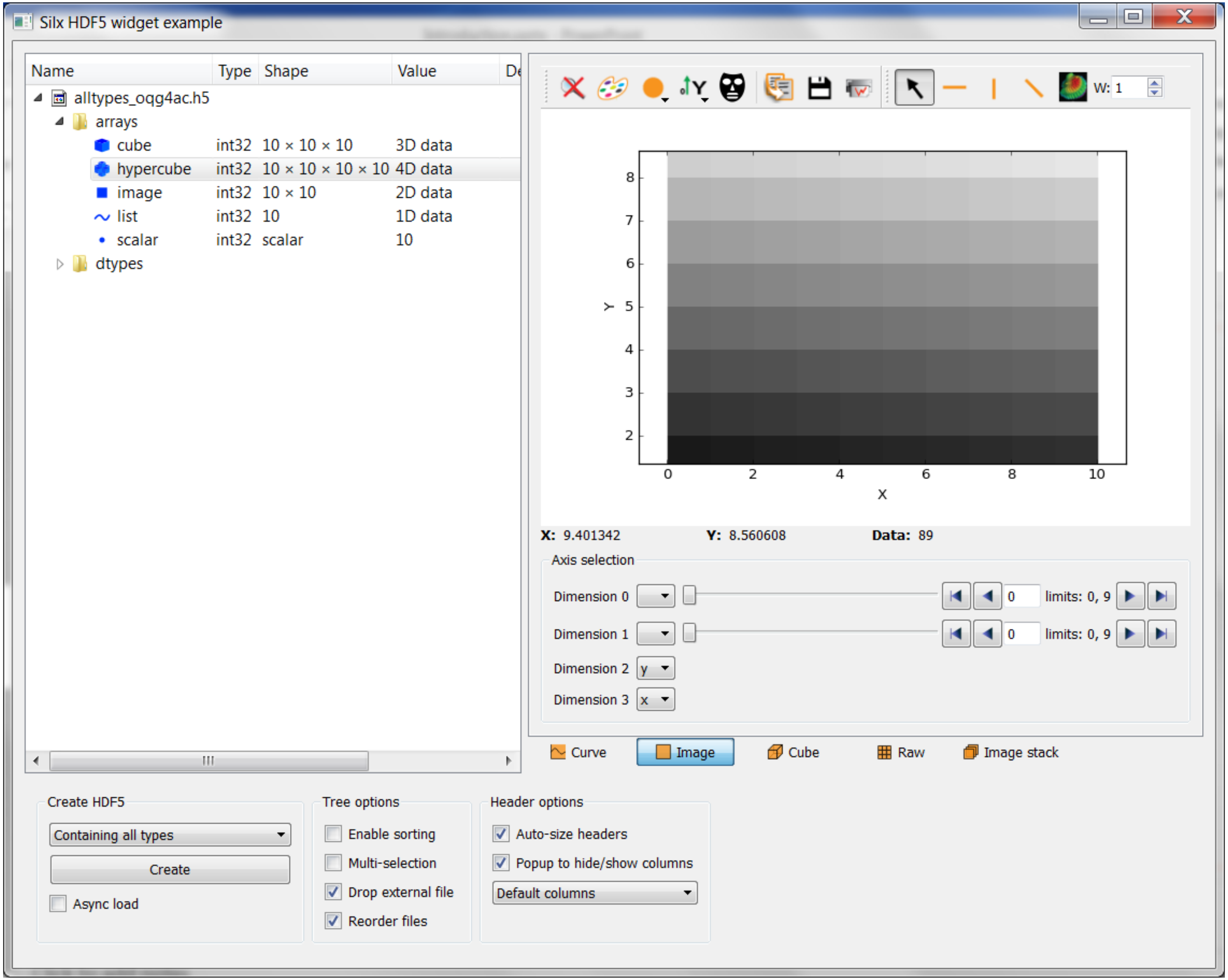
☑ Reorder files

Header options

☑ Auto-size headers

☑ Popup to hide/show columns

Default columns



Name	Type	Shape	Value	De
alltypes_oqg4ac.h5				
arrays				
cube	int32	10 × 10 × 10	3D data	
hypercube	int32	10 × 10 × 10 × 10	4D data	
image	int32	10 × 10	2D data	
list	int32	10	1D data	
scalar	int32	scalar	10	
dtypes				

X: 9.401342 **Y:** 8.560608 **Data:** 89

Axis selection

Dimension 0 0 limits: 0, 9

Dimension 1 0 limits: 0, 9

Dimension 2 y

Dimension 3 x

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

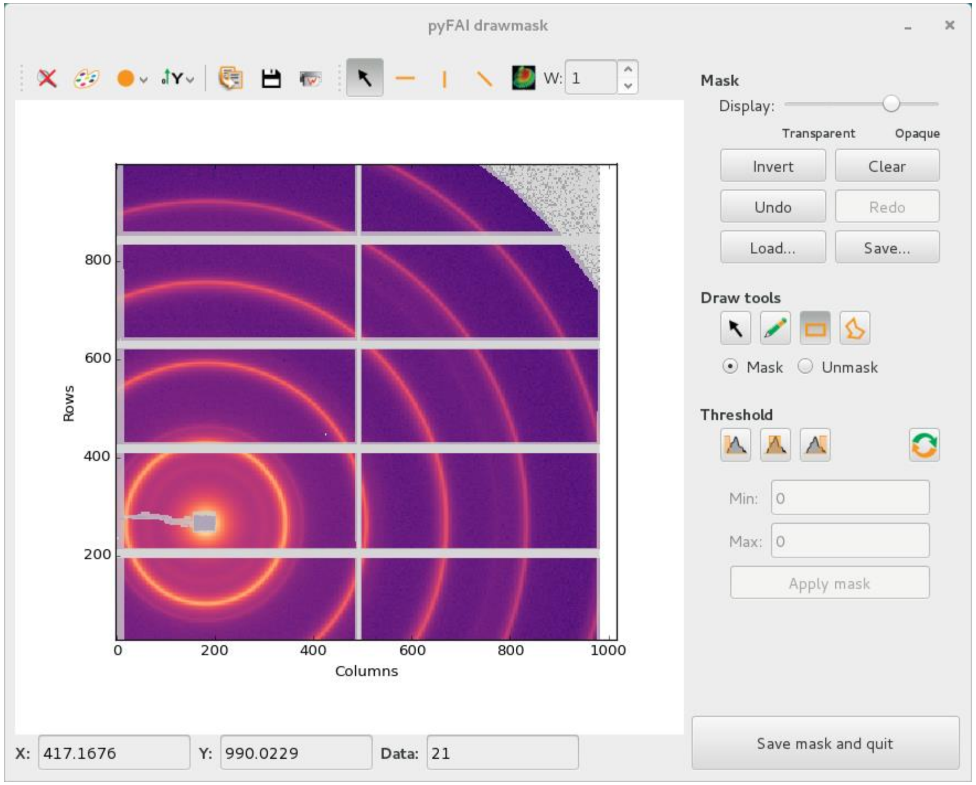
Default columns

silx.math

- Weighted n-dimensional histograms
- Fast histogramming using look up tables
- Non-linear least squares fits with constraints
- 1D peak search
- 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



W: 1

● Mask ○ Unmask

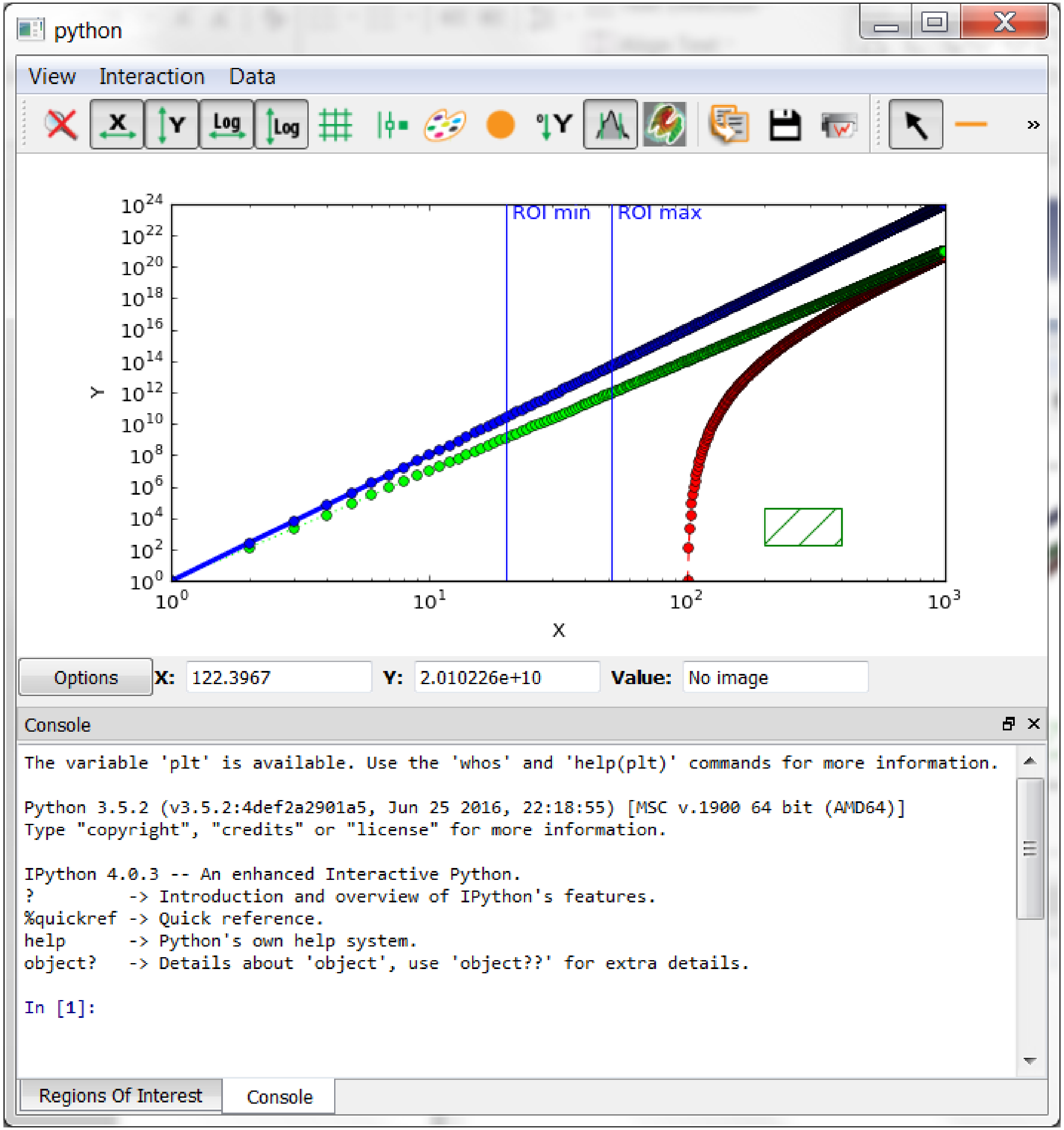
Min: 0

Max: 0

X: 417.1676 Y: 990.0229 Data: 21

silx.gui: Plot 1D

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



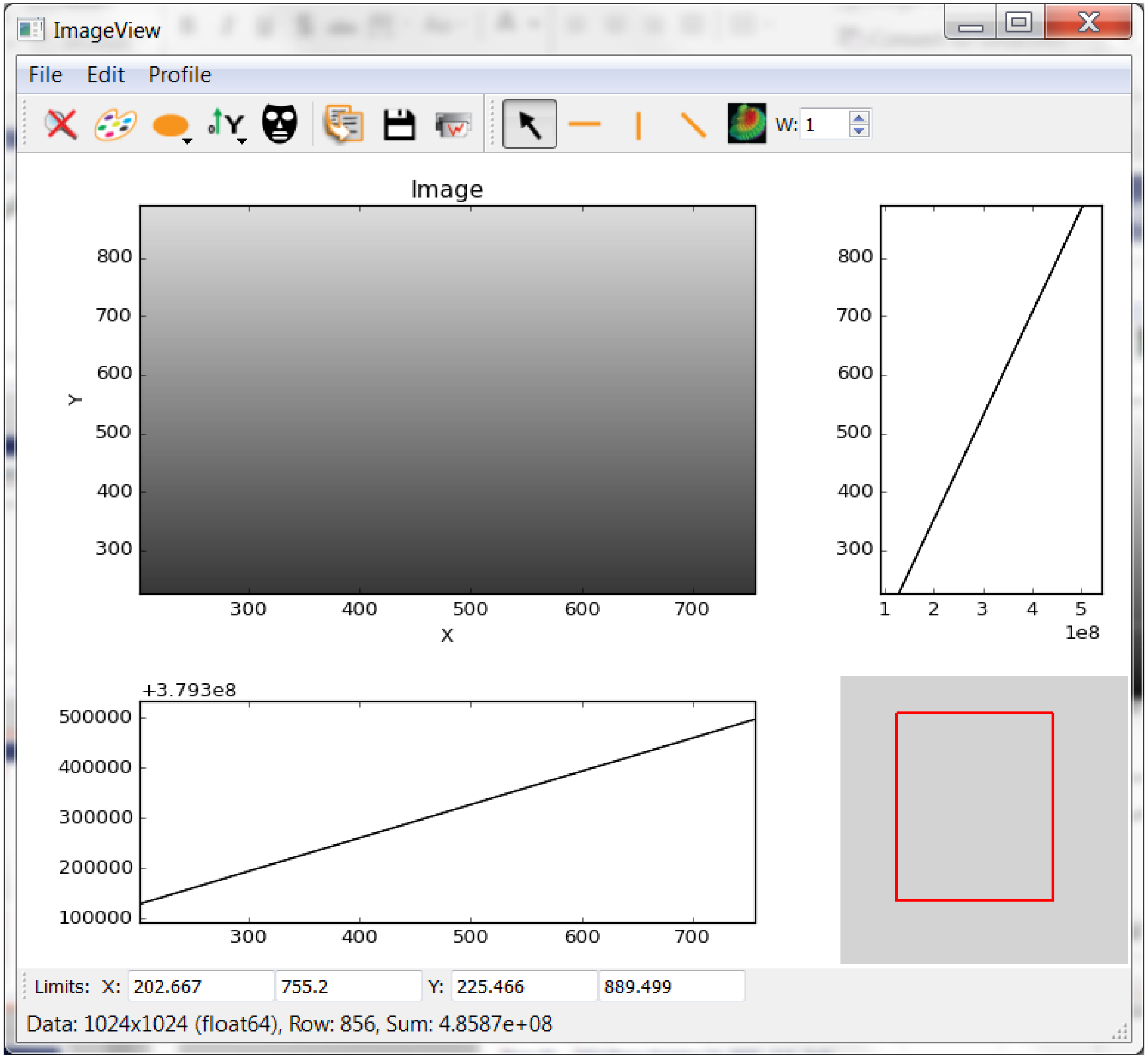
X: 122.3967 Y: 2.010226e+10 Value: No image

```
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]:
```

silx.gui: Plot

- 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)

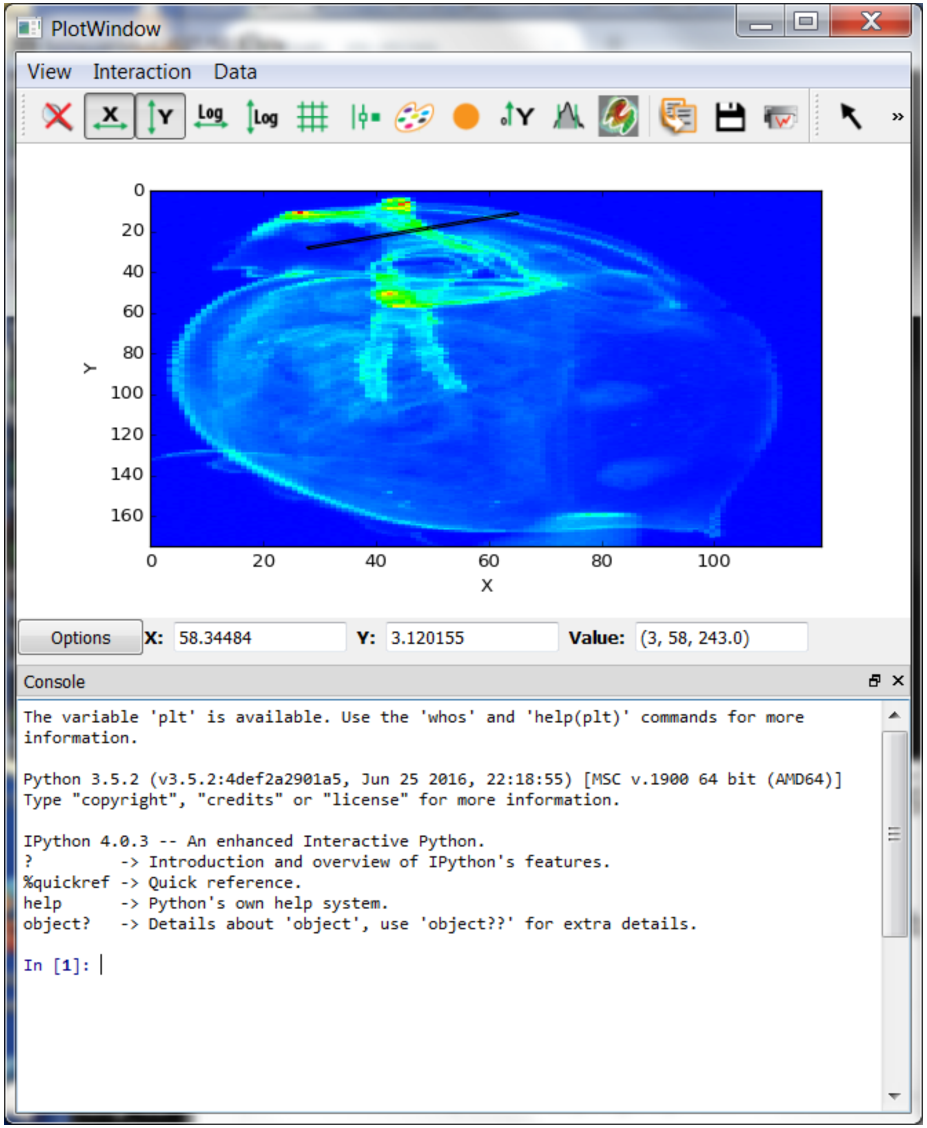
Full-featured Widgets



Limits: X: 202.667 755.2 Y: 225.466 889.499

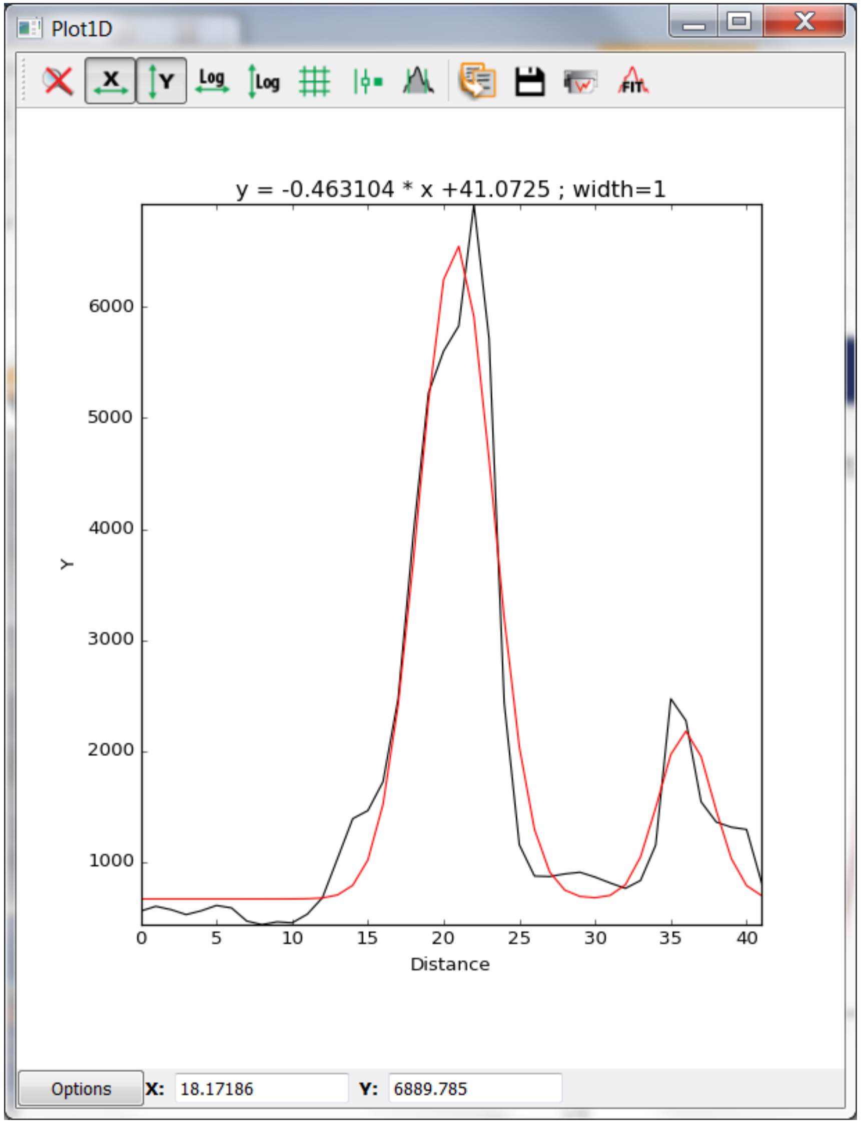
Data: 1024x1024 (float64), Row: 856, Sum: 4.8587e+08

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]: |
```

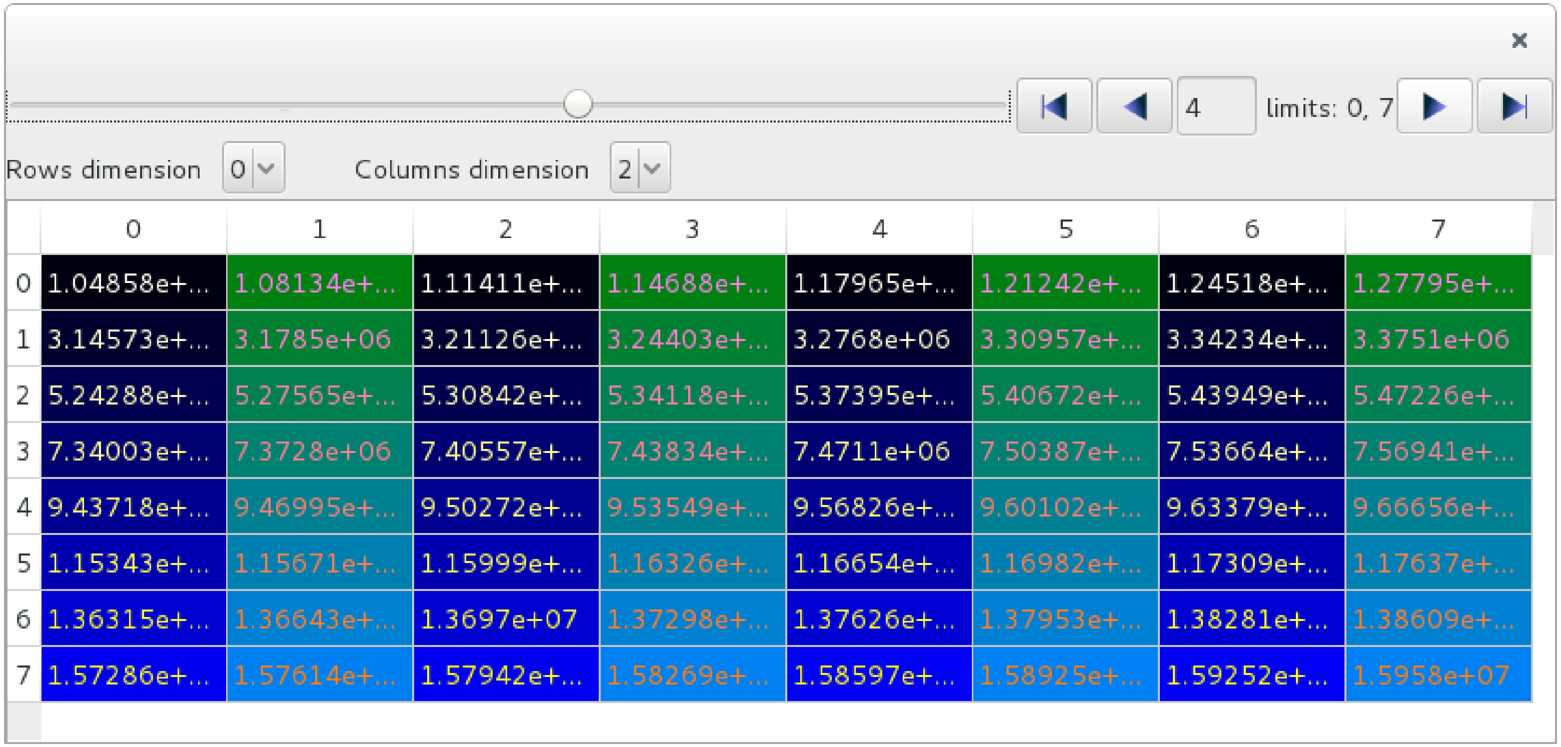


X: 18.17186 Y: 6889.785

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

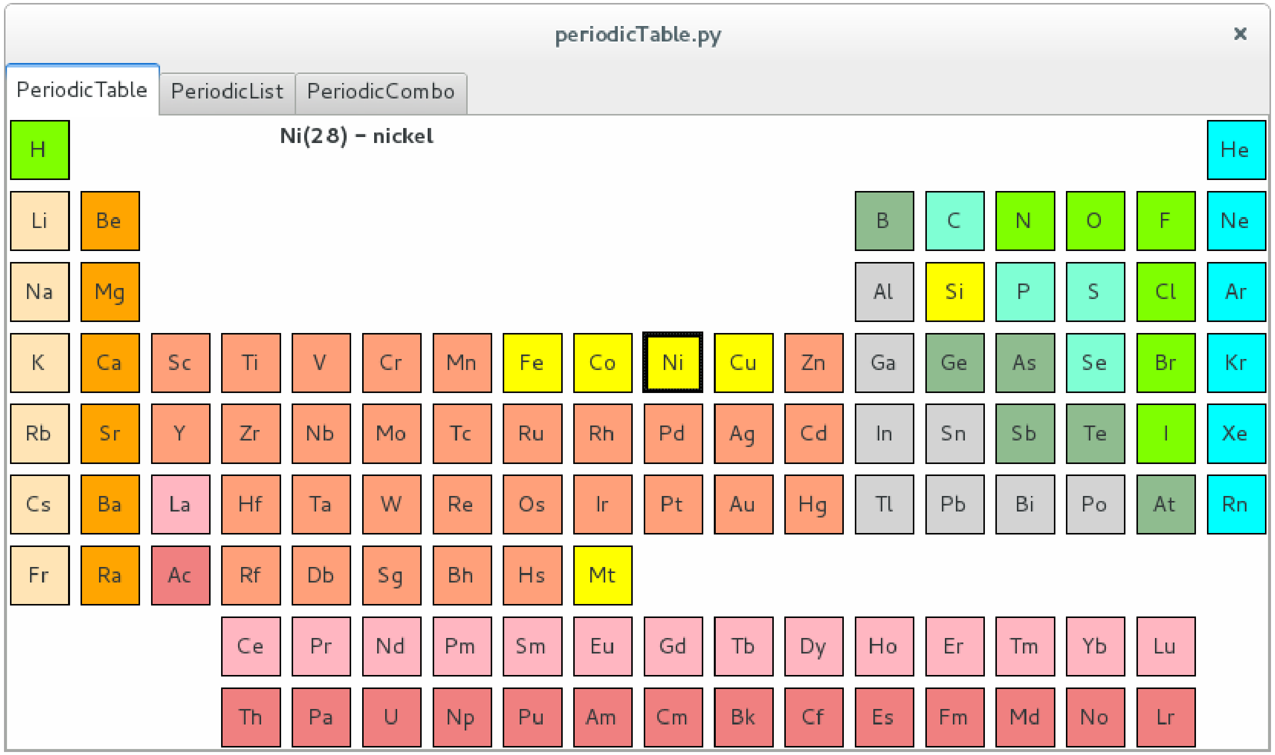


	0	1	2	3	4	5	6	7
0	1.04858e+...	1.08134e+...	1.11411e+...	1.14688e+...	1.17965e+...	1.21242e+...	1.24518e+...	1.27795e+...
1	3.14573e+...	3.1785e+06	3.21126e+...	3.24403e+...	3.2768e+06	3.30957e+...	3.34234e+...	3.3751e+06
2	5.24288e+...	5.27565e+...	5.30842e+...	5.34118e+...	5.37395e+...	5.40672e+...	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

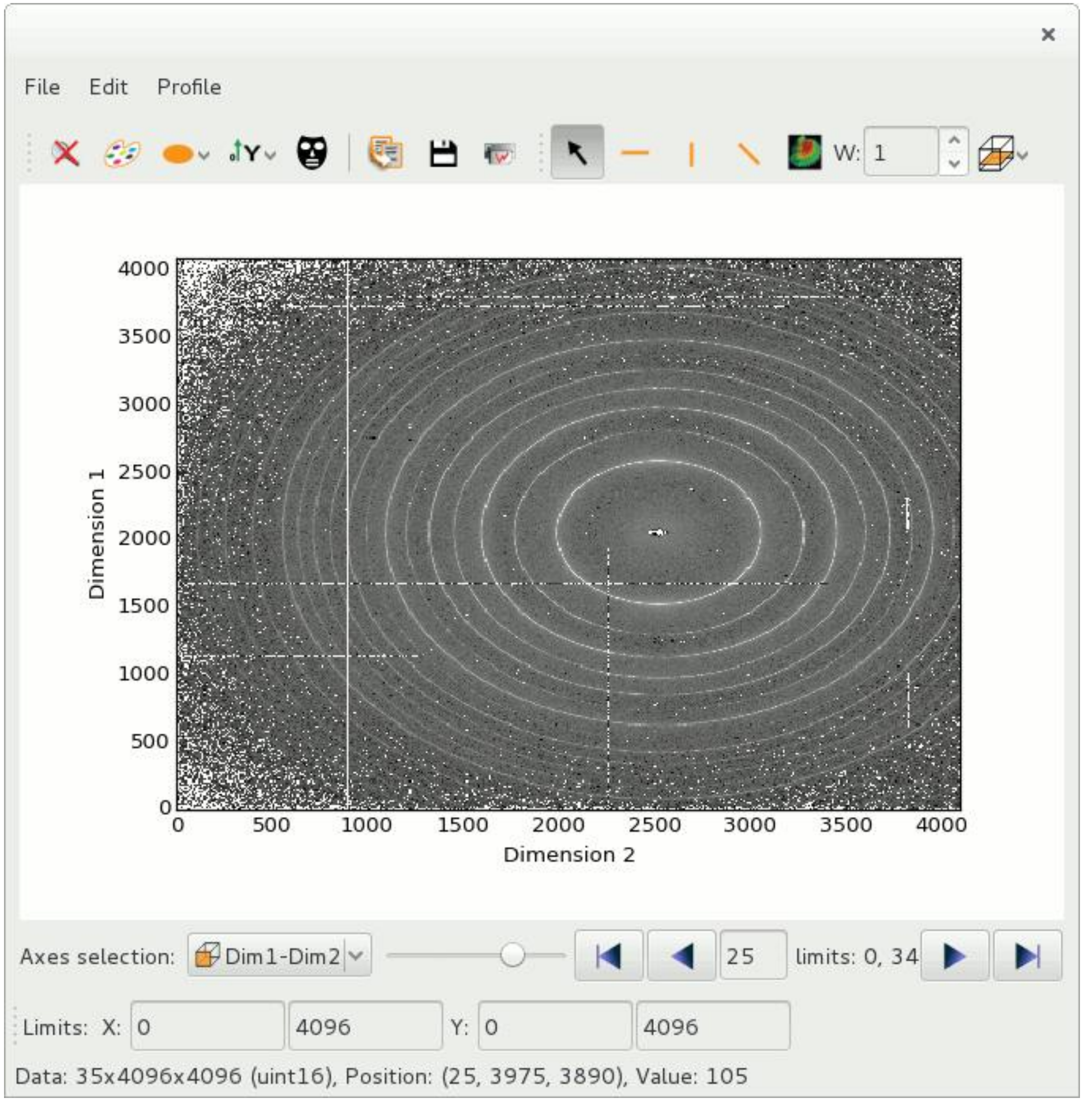


Ni(28) – nickel

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)



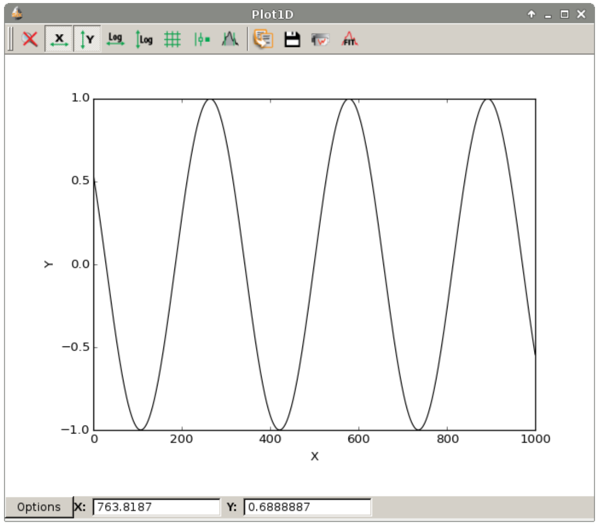
Data: 35x4096x4096 (uint16), Position: (25, 3975, 3890), Value: 105

silx.sx: a module to simplify interactive use

pylab like module on steroids

- 1D plotting: ROI, fitting & printing

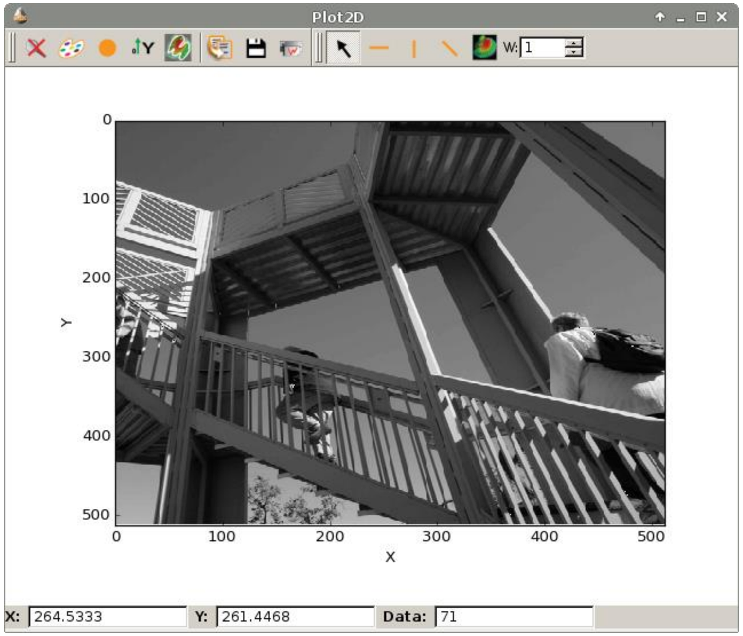
```
>>> from silx import sx
>>> from numpy import sin, linspace
>>> sx.plot(sin(linspace(-10, 10, 1000)))
```



X: 763.8187 Y: 0.6888887

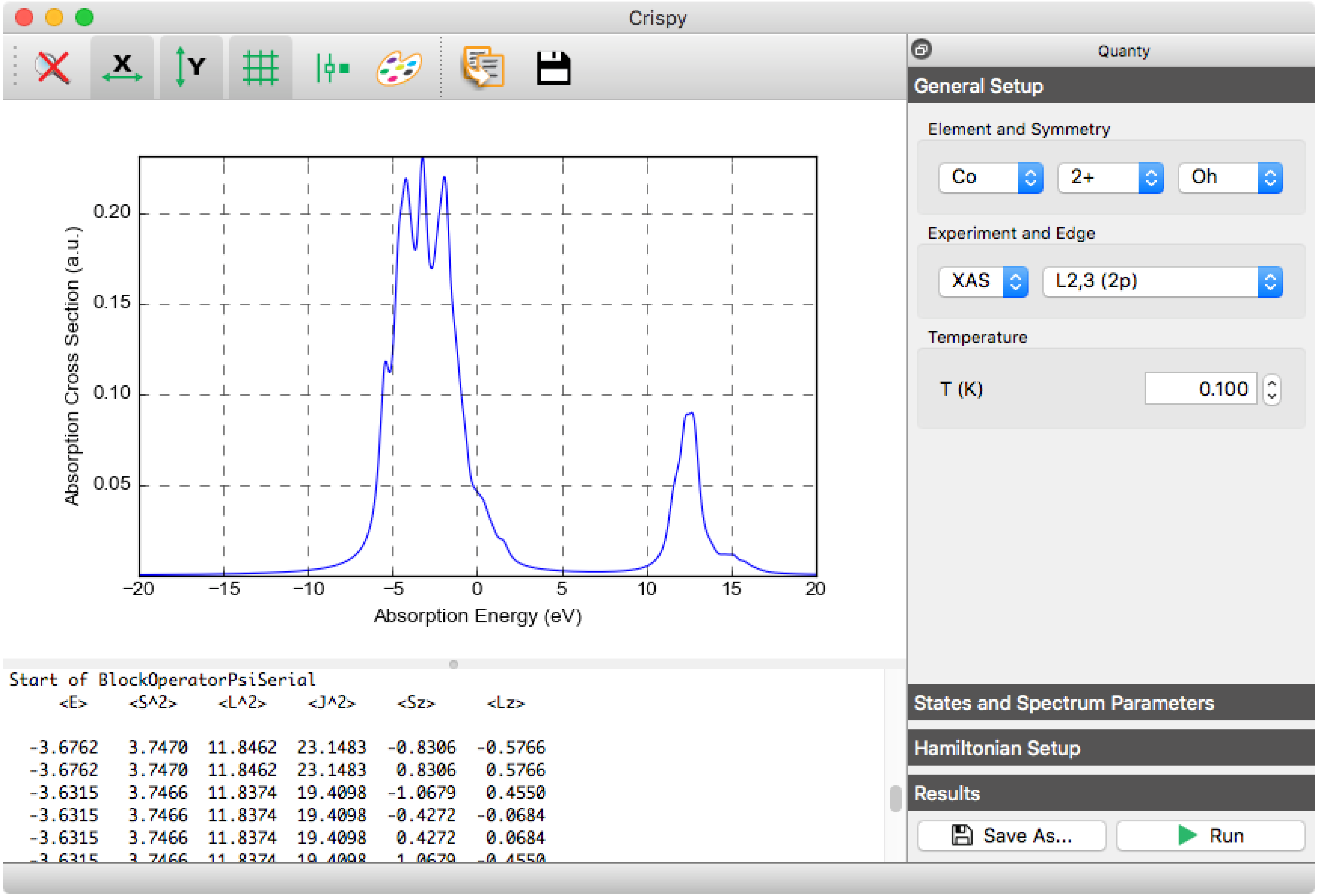
- 2D display: intensity, mask, profile

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



X: 264.5333 Y: 261.4468 Data: 71

Applications - Crispy



Element and Symmetry

Co 2+ Oh

Experiment and Edge

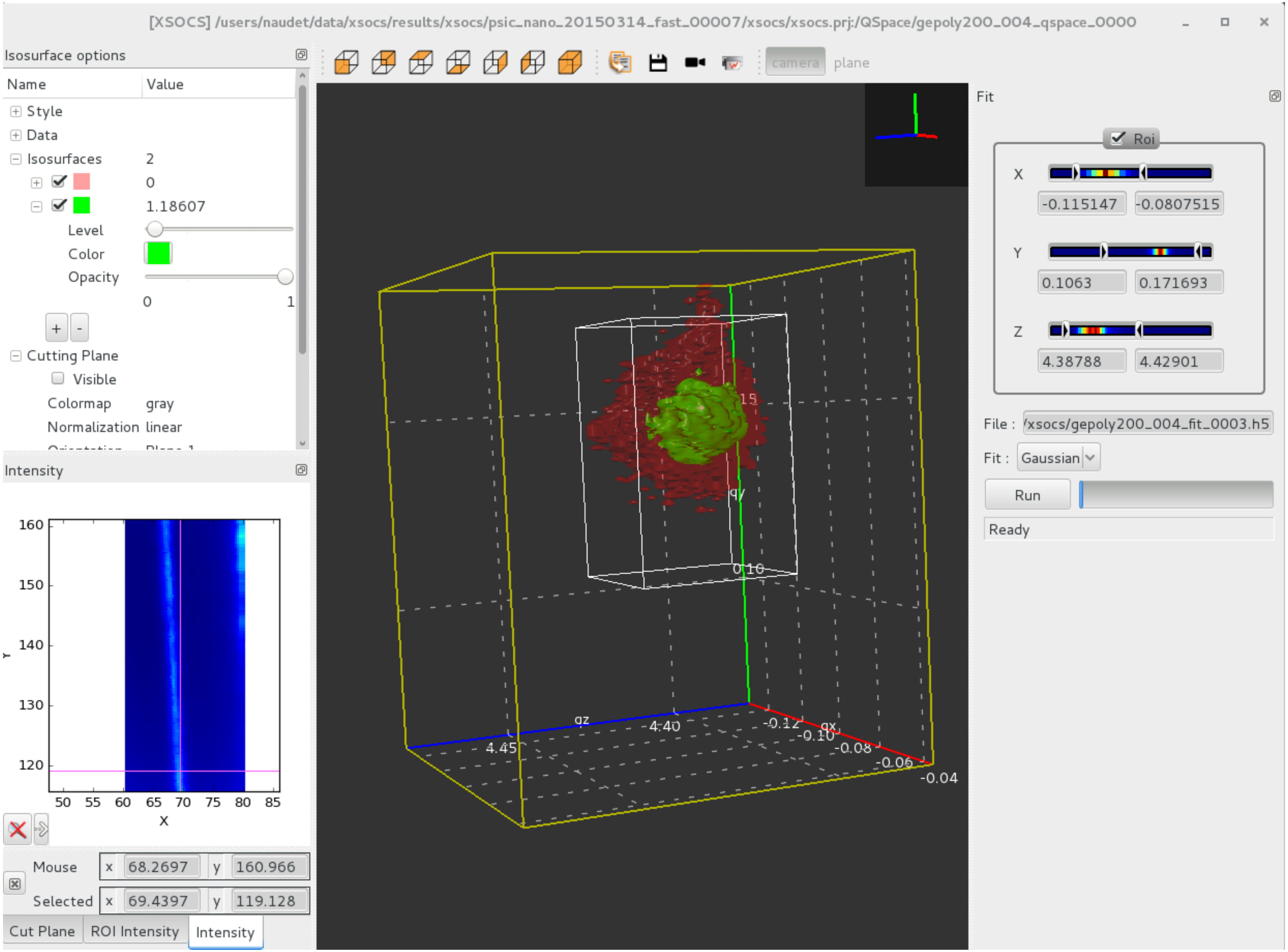
XAS L2,3 (2p)

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

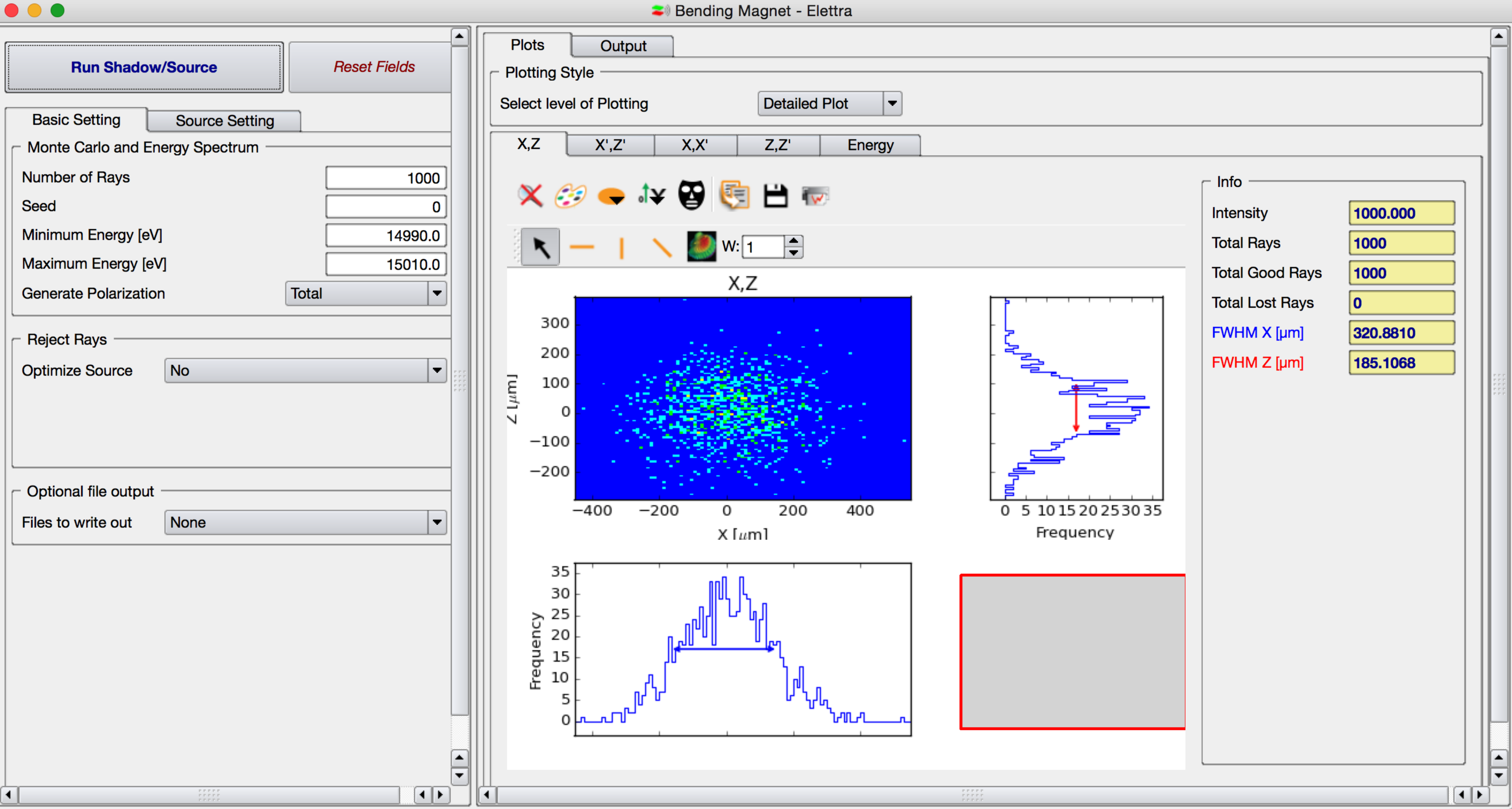
Ready

Intensity

Mouse x 68.2697 y 160.966

Selected x 69.4397 y 119.128

Applications - OASYS



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

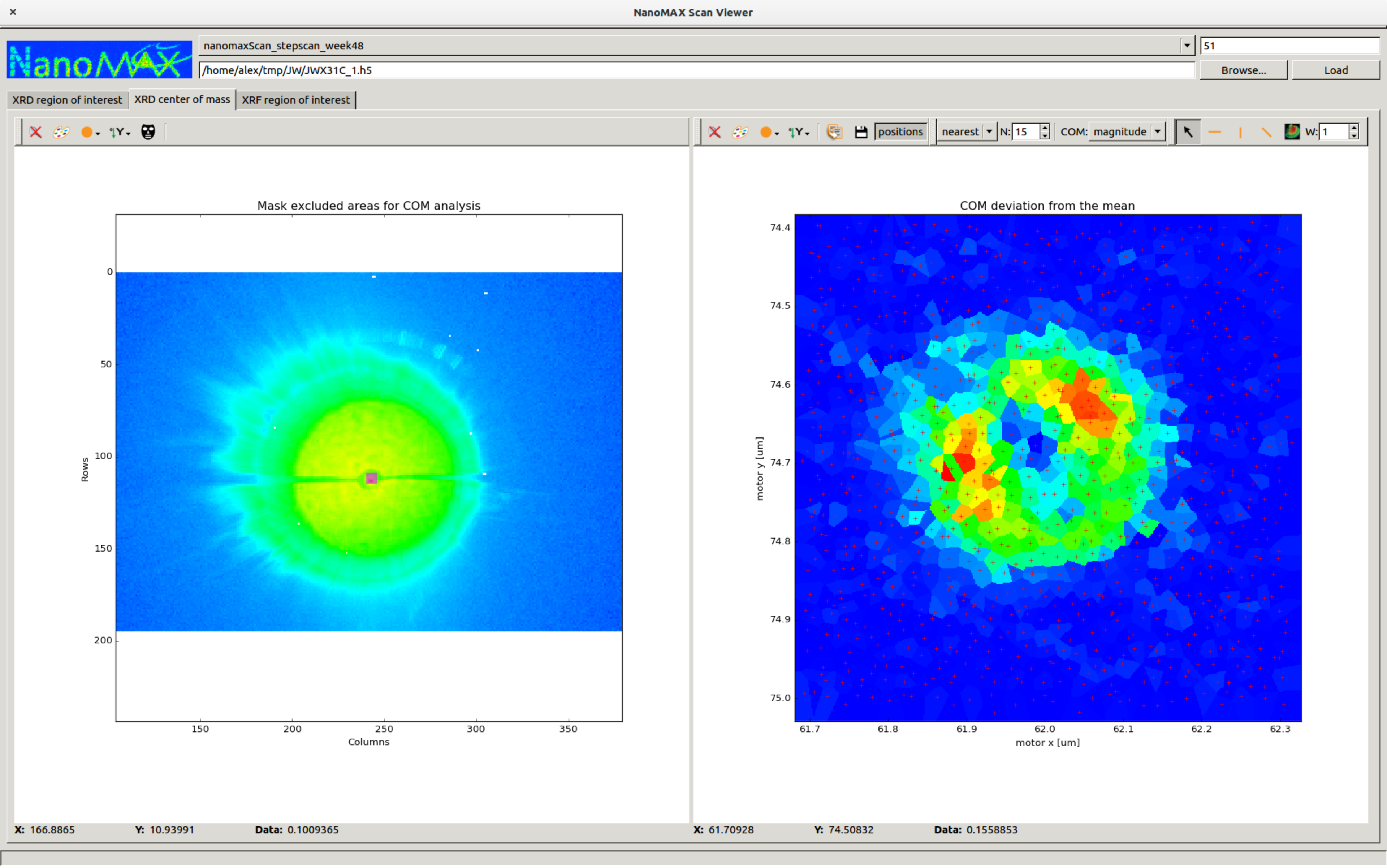
Plotting Style

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 – Nanomax@Max IV



NanoMAX Scan Viewer

nanomaxScan_stepscan_week48

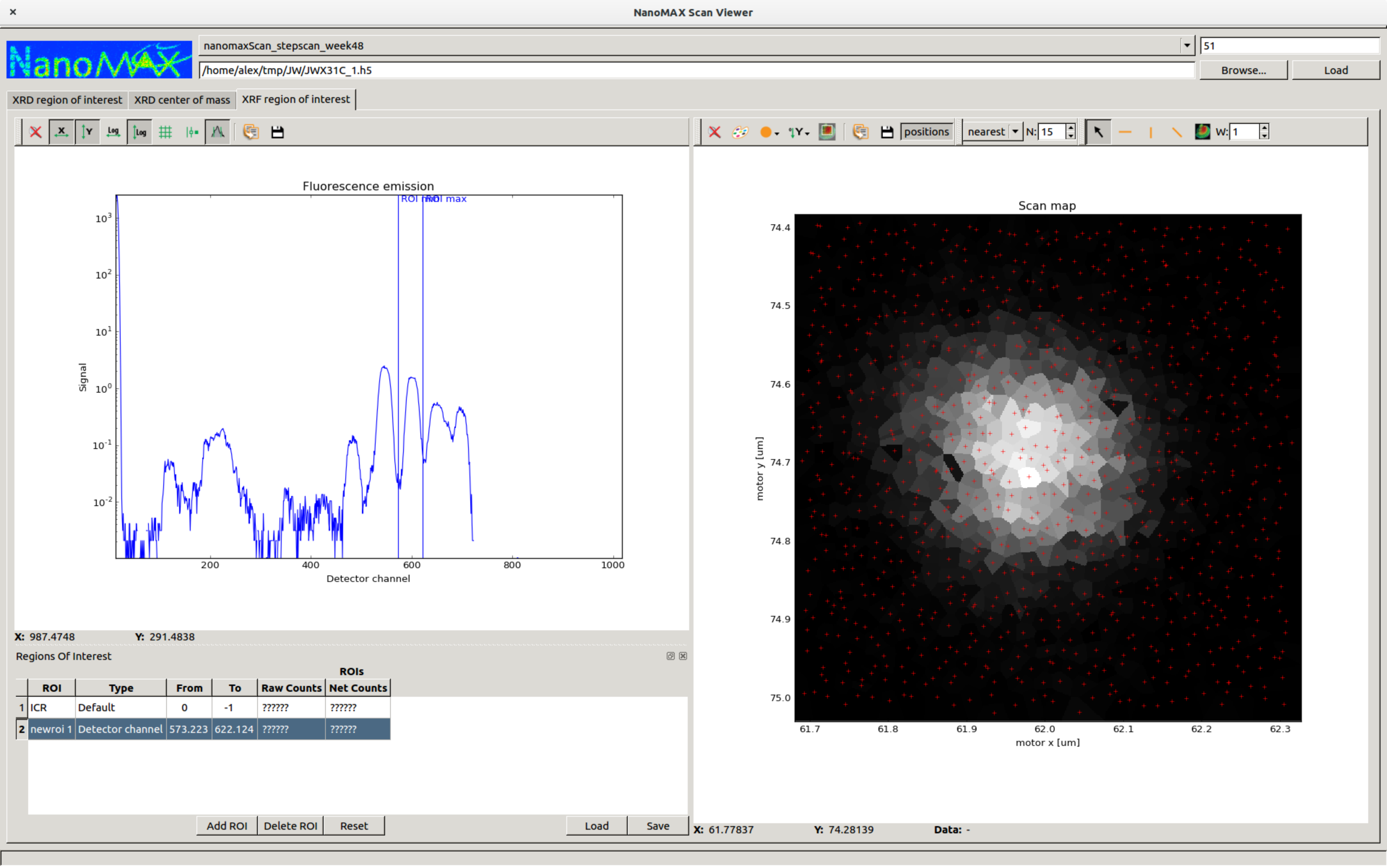
51

/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

Applications – Nanomax@Max IV



NanoMAX Scan Viewer

nanomaxScan_stepscan_week48

51

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

X: 987.4748 Y: 291.4838

Regions Of Interest

ROIs

	ROI	Type	From	To	Raw Counts	Net Counts
1	ICR	Default	0	-1	??????	??????
2	newroi 1	Detector channel	573.223	622.124	??????	??????

X: 61.77837 Y: 74.28139 Data: -

Roadmap

- This release
  - Object Oriented Plot API
  - OpenGL Plot Backend
  - NXdata Viewer
- Late 2017
  - 3D SceneGraph
  - Print Preview
  - pyFAI Calibration GUI
  - PyMca using silx Plot
- 2018
  - pyFAI 0.14 release with pyFAI GUI
- 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 0.4.0 version
- For this code camp we’ll use 0.5.0a, 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/