



9th Silx Code Camp February 11, 2019



This talk

- Introduction
 - Novelties (version 0.10.0)
- Status of silx (version 0.9.0)
- Goals of the code camp
 - For users
 - For core developers
- Hands on!



silx.math.fft: silx FFT

- A new module for Fast Fourier Transform: `silx.math.fft`
- One unique interface with 4 backends:
 - numpy, fftw, OpenCL, CUDA.
- 1D, 2D, 3D ; possibly batched
- R2C, C2C
- For GPU transforms, input and/or output can be device arrays.
- Currently not supported :
 - In-place transforms
 - Hermitian transforms
 - Automatic zero-padding (ex. `fft(data, size=2048)`)



Simple FFT with numpy

```
import numpy as np
from scipy.misc import ascent
from silx.math.fft import FFT
img = ascent().astype(np.float32)

F = FFT(data=img, backend="numpy") # automatically chooses R2C transform
img_f = F.fft(img)
```

Using FFTW

```
F = FFT(data=img, backend="fftw", num_threads=4)
img_f = F.fft(img)
# do some operation of img_f ...
F.ifft(img_f)
```



silx.math.fft: silx FFT

Using OpenCL

```
import pyopencl.array as parray
F = FFT(data=img, backend="opencl")
# All the Host <-> Device copies are handled under the hood
img_f = F.fft(img) # by default, result is a numpy array
# Input and/or output can be device array as well
d_in = parray.to_device(F.queue, img)
d_out = parray.zeros(F.queue, F.shape_out, dtype=F.dtype_out)
F.fft(d_in, output=d_out)
```

Using CUDA

```
import pycuda.autoinit
import pycuda.gpuarray as gpuarray
F = FFT(data=img, backend="cuda")
d_in = gpuarray.to_gpu(img)
d_out = gpuarray.zeros(F.shape_out, F.dtype_out)
F.fft(d_in, output=d_out) # CUFFT is twice faster than clfft for R2C transforms
```



silx.openc1.backprojection: silx backprojector

- New features in *silx.openc1.backprojection*:
- Filtering is done with *silx.math.fft* (possibly on GPU)
- User can choose other built-in and custom filters
- Input and/or output of backproj/FBP can be numpy and pyopenc1 arrays



silx.openc1.backprojection: silx backprojector

```
import numpy as np
import pyopenc1.array as parray
from silx.openc1.backprojection import Backprojection
from silx.test.utils import utilstest
from silx.gui import qt
from silx.gui.plot.CompareImages import CompareImages

sino = np.load(utilstest.getfile("sino500.npz"))["data"]

B1 = Backprojection(sino.shape)
rec1 = B1(sino)

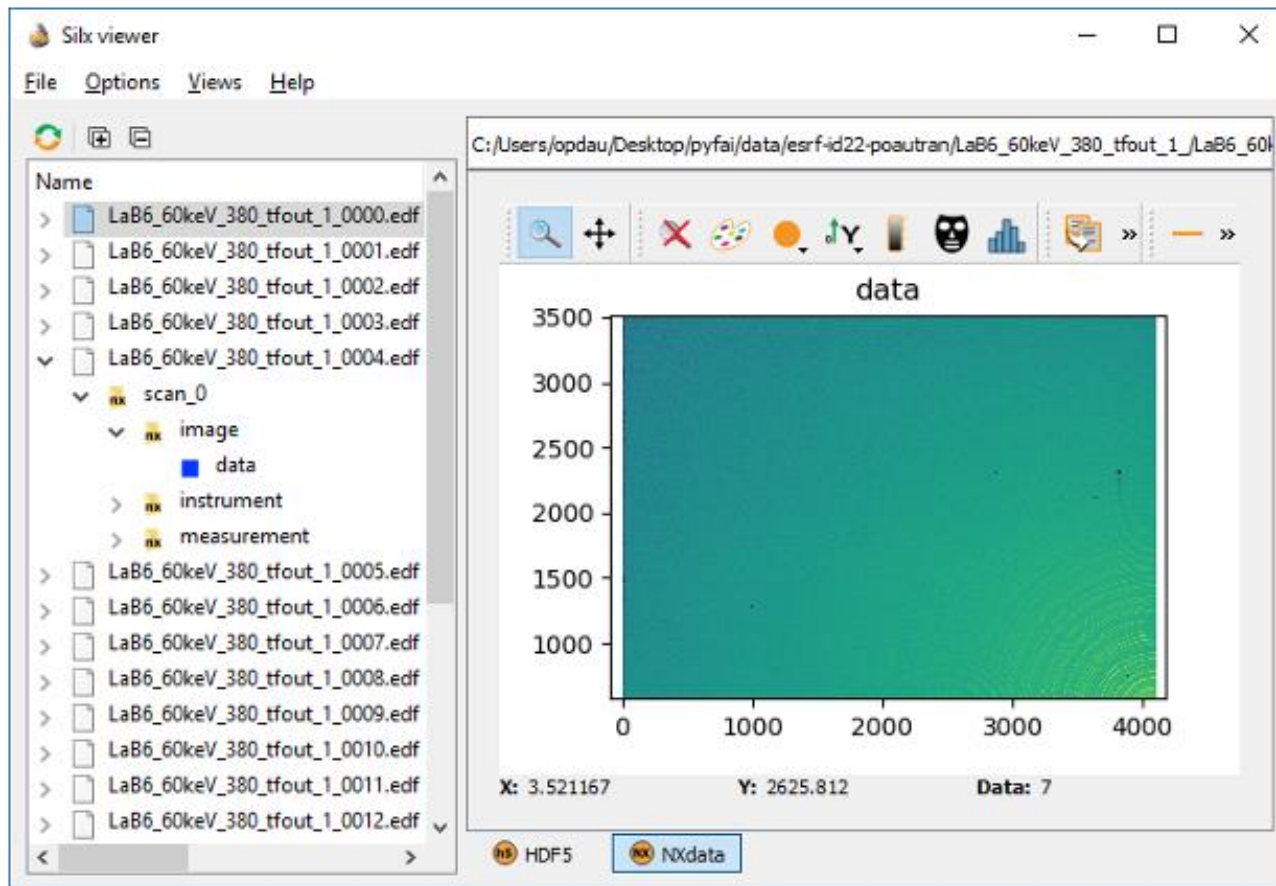
B2 = Backprojection(
    sino.shape,
    filter_name="hamming",
    extra_options={"cutoff": 0.7}
)
d_sino = parray.to_device(B2.queue, sino)
d_rec2 = parray.zeros(B2.queue, B2.slice_shape, "f")
B2(d_sino, output=d_rec2)

app = qt.QApplication([])
C = CompareImages()
C.setData(rec1, d_rec2.get())
C.show()
app.exec_()
```



Silx view

- Image displayed without browsing it
 - A default NXdata is generated





Silx view

- Drag & drop data as silx URL
- AS text/plain and application/x-silx-uri

The screenshot displays the Silx viewer application window. On the left, a file icon labeled 'hc3532_lzf_pyFAI.h5' is shown with a pink arrow pointing to the file tree in the main window. The file tree lists the following structure:

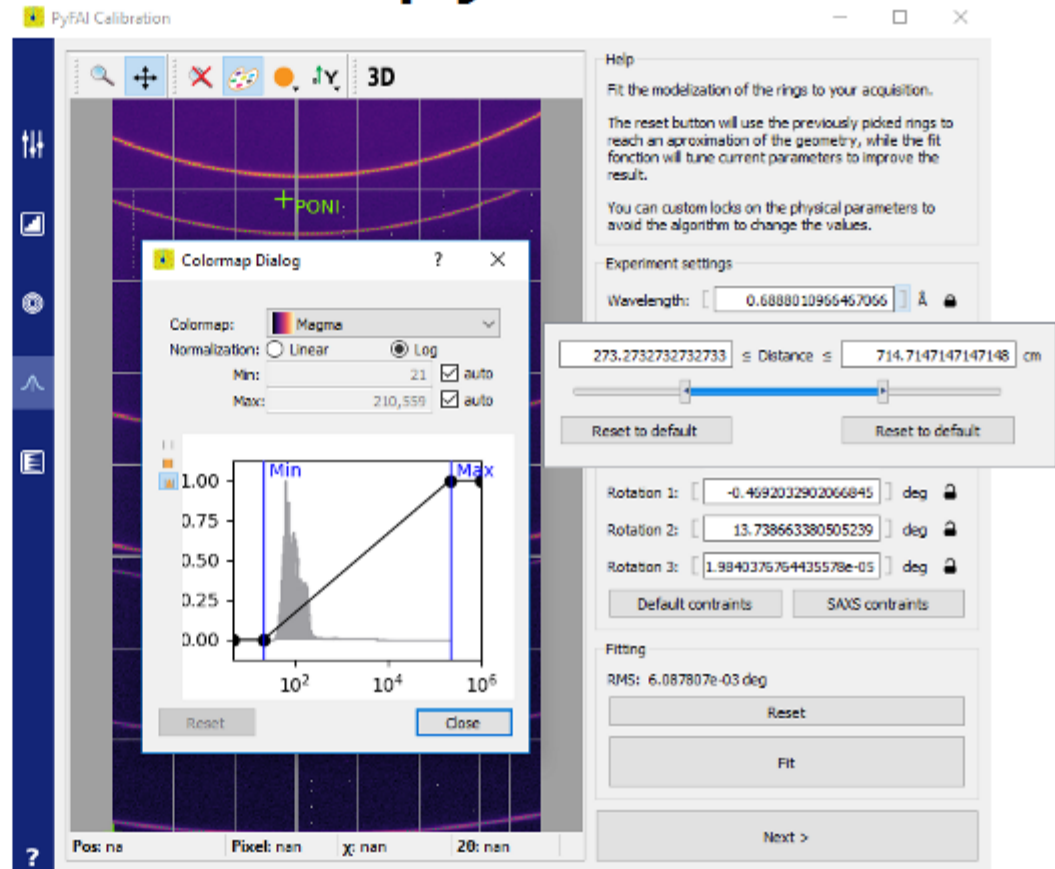
Name	Type	Shape
hc3532_lzf_pyFAI.h5	NXentry	
data	NXentry	
end_time	string	scalar
integrate	NXprocess	
config		
end_time	string	scalar
program	string	scalar
results	NXdata	
data	float32	152 x 1000
radial	float32	100
start_time	string	scalar
version	string	scalar
start_time	string	scalar
title	string	scalar

The main plot area shows a line graph titled 'data' with a y-axis from 0 to 6000 and an x-axis from 0 to 50. The plot shows a sharp peak at x=0 followed by a decay. Below the plot, the 'Options' section shows X: 5.696135 and Y: 6173.398. A pink arrow points from the 'data' entry in the file tree to the plot. Another pink arrow points from the 'data' entry to a terminal window at the bottom, which displays the URL: `file:///.../hc3532_lzf_pyFAI.h5?/integrate/results/data`. A third pink arrow points from the 'data' entry to a heatmap plot at the bottom right, which has a y-axis from 0 to 100 and an x-axis from 0 to 1000. A 'Drop something here' label is visible below the heatmap.



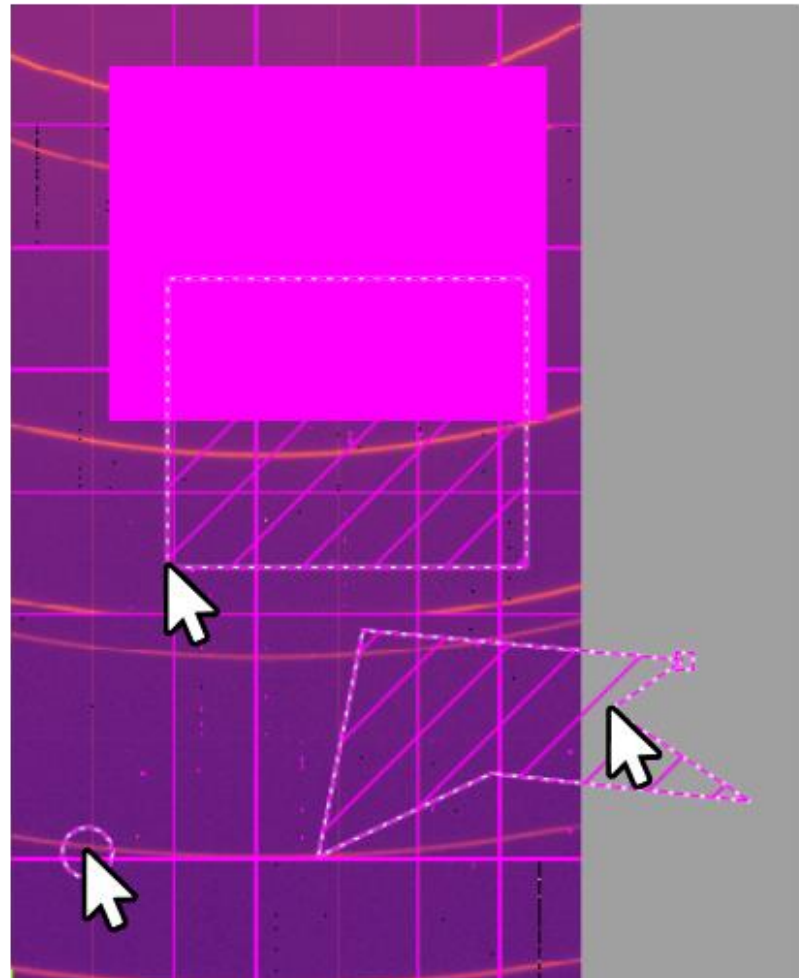
Improvements for pyFAI

- Improve of `silx.gui.widgets.RangeSlider`
- Colormap histogram in log
- Plot background



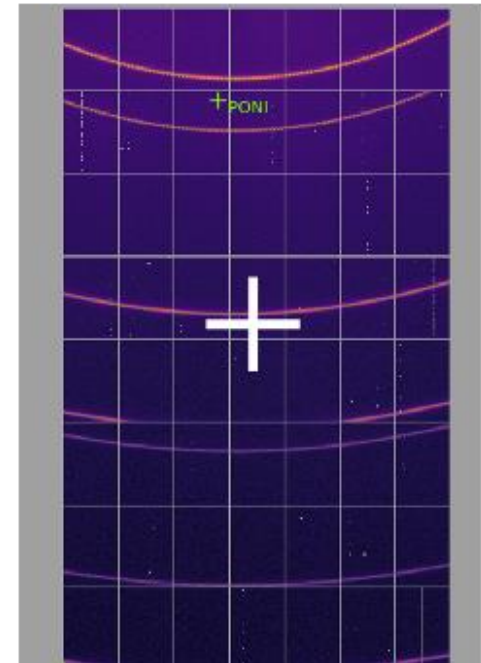
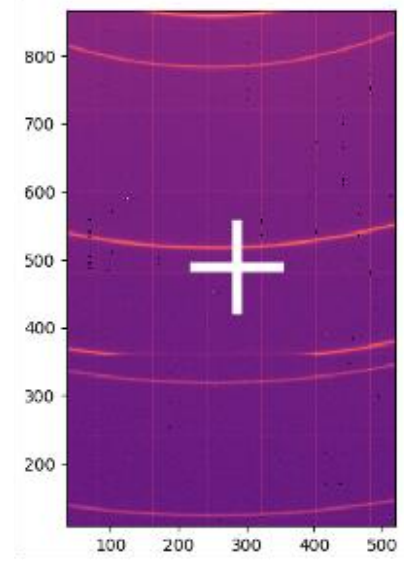
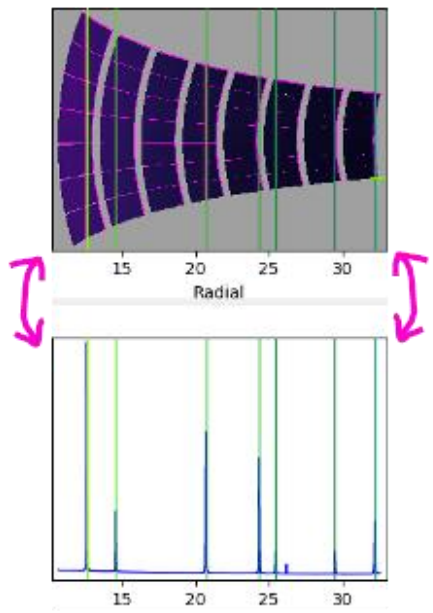
Improvements for pyFAI

- Bicolor strokes
- Mask tools now always visible



Improvements for pyFAI

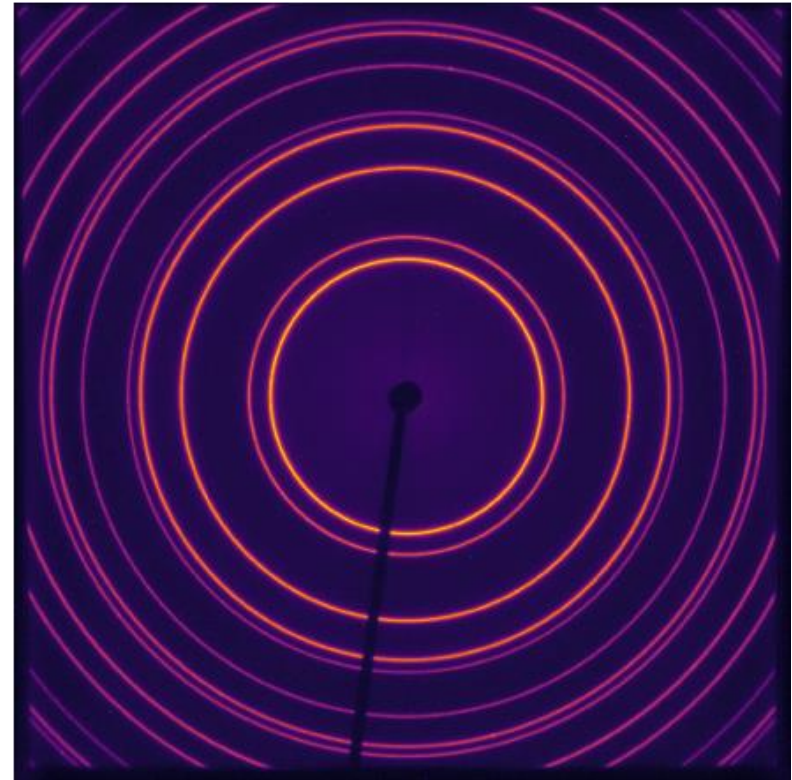
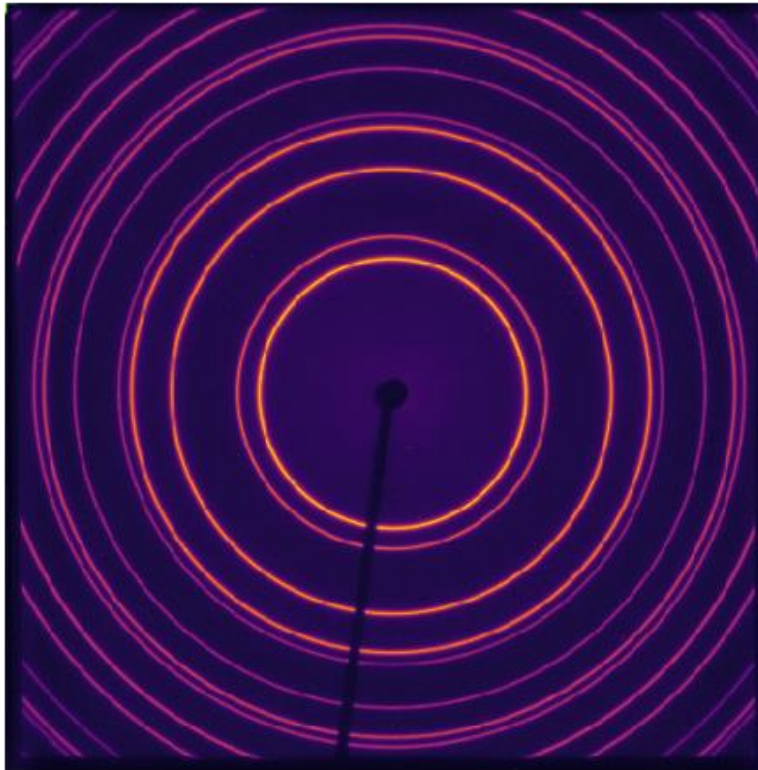
- `silx.gui.plot.utils.axis.SyncAxes`
 - Previously: Sync min/max of axes
 - In addition now: Sync center and pixel size





silx.gui.plot: Improvements

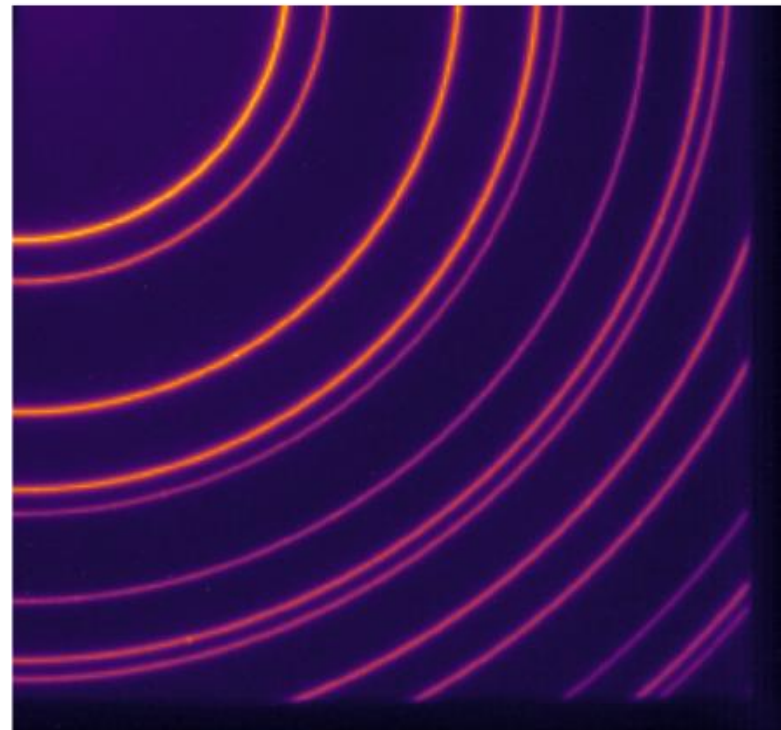
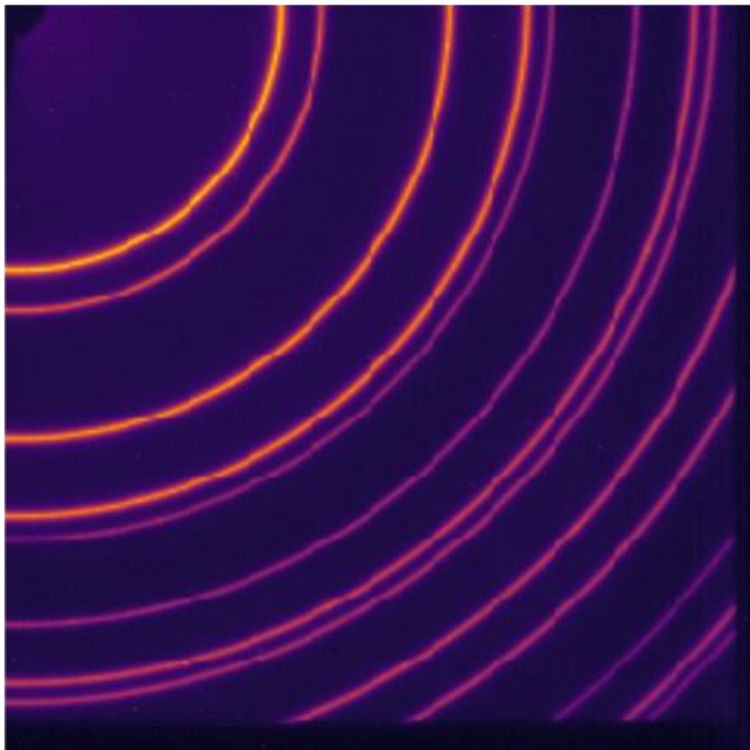
- Fix image rendering glitches
 - Thanks Jonathan Wright





silx.gui.plot: Improvements

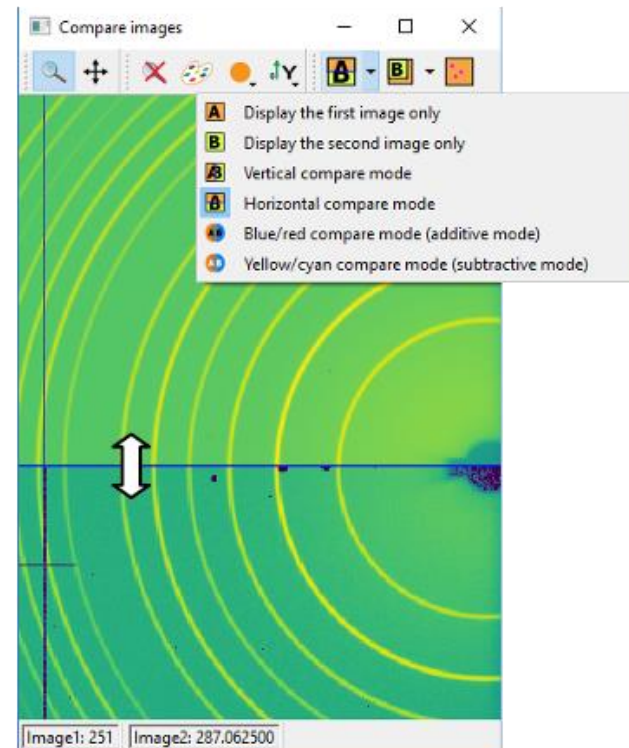
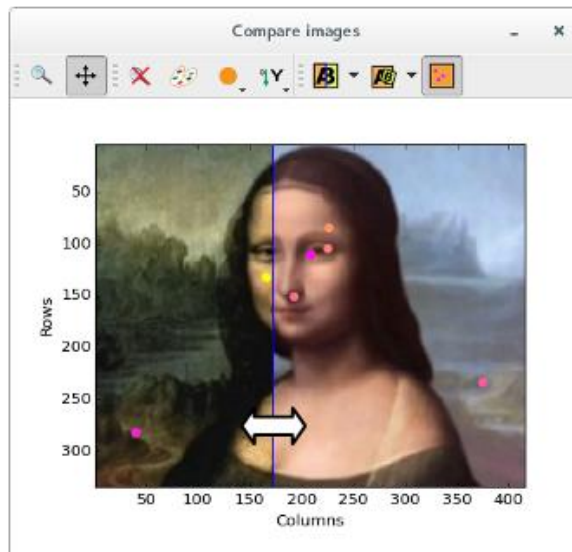
- Fix image rendering glitches
 - Thanks Jonathan Wright





silx.gui.plot.CompareImages

- As widget: `silx.gui.plot.CompareImages`
 - `python -m silx.examples.compareImages`





silx.gui.widgets: Url Selection Table

- Select two url from a list
- Extend the compareImages.py examples if more than two files in input

Terminal window showing the command:

```
(venv)linazimov:silx/paynoSilx/silx % python examples/compareImages.py /nobackup/linazimov/payno/datasets/id19/D2_H2_T2_h_old/*.edf
```

GUI window showing a table of URLs and a comparison image viewer.

url	img A	img B
D2_H2_T2_h_0797.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0798.edf	<input checked="" type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0799.edf	<input type="radio"/>	<input checked="" type="radio"/>
D2_H2_T2_h_0800.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0801.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0802.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0803.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0804.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0805.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0806.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0807.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0808.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0809.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0810.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0811.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0812.edf	<input type="radio"/>	<input type="radio"/>
D2_H2_T2_h_0813.edf	<input type="radio"/>	<input type="radio"/>

Image viewer showing two images side-by-side. The status bar at the bottom indicates: Image1: No data Image2: No data



silx.io.utils: HDF5 External storage helpers

- Two helper functions to create an external dataset:

- From a .vol file (pyhst):

→ `vol_to_h5_external_dataset(vol_file, output_url,
info_file=None,
vol_dtype=numpy.float32,
overwrite=False)`

- From a binary file:

→ `rawfile_to_h5_external_dataset(bin_file,
output_url,
shape,
dtype, overwrite=False)`

- Need `h5py >= 2.9`



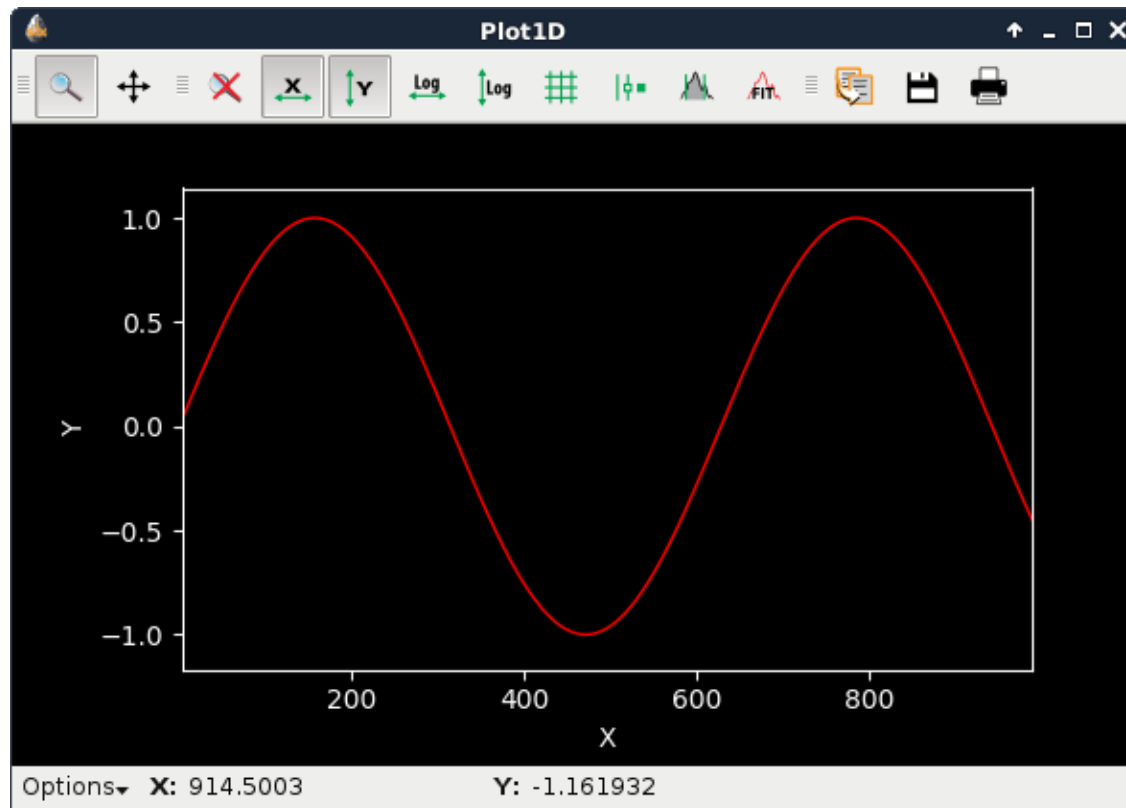
silx.opencl:

- Stats calculated with OpenCL (mean, STD)



silx.gui.plot: PlotWidget Updates

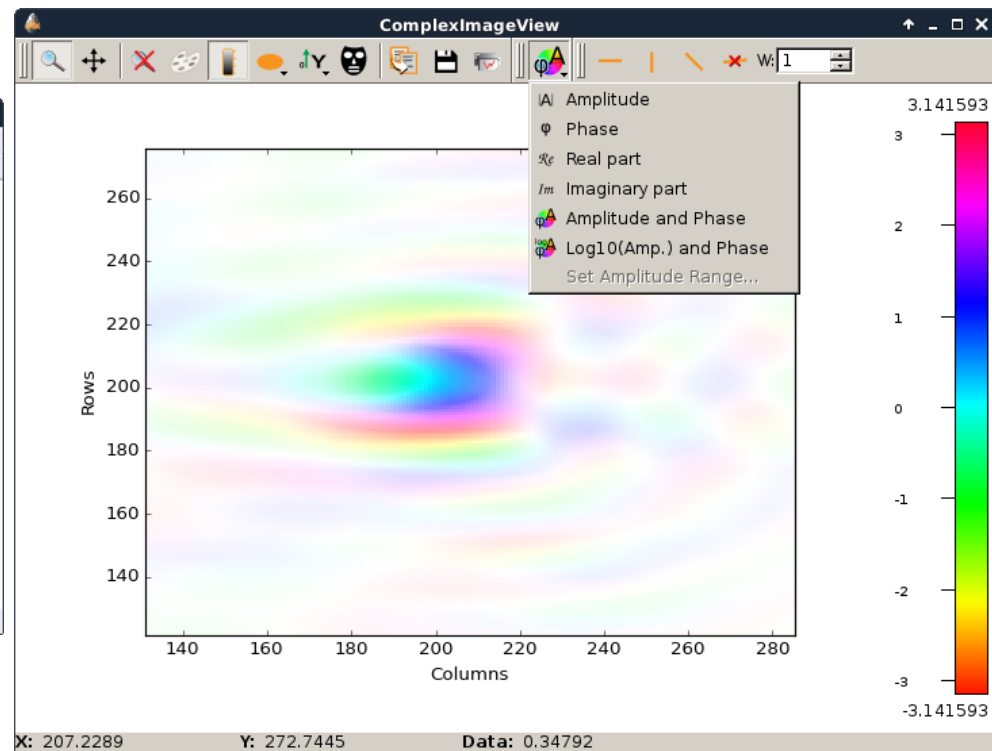
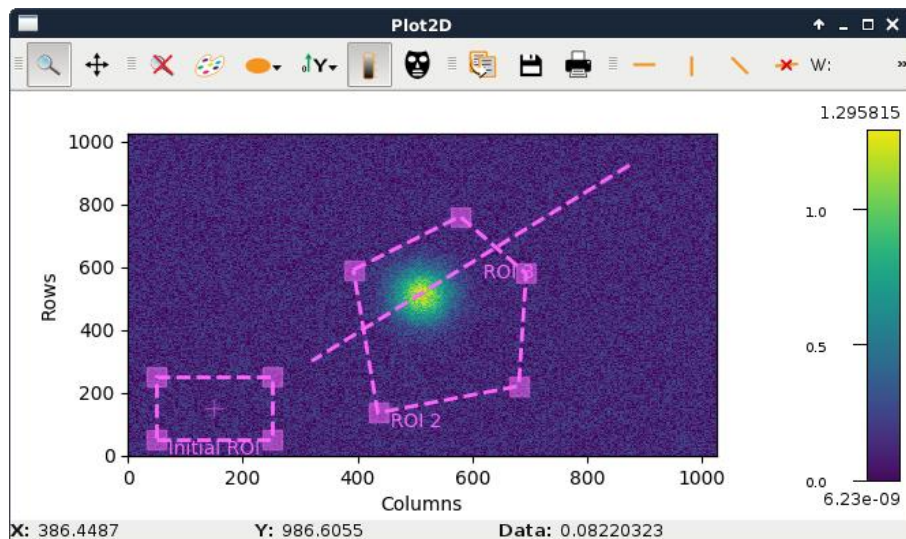
- Remove more than one year old deprecated methods
- Make background/foreground settable





silx.gui.plot: Updates

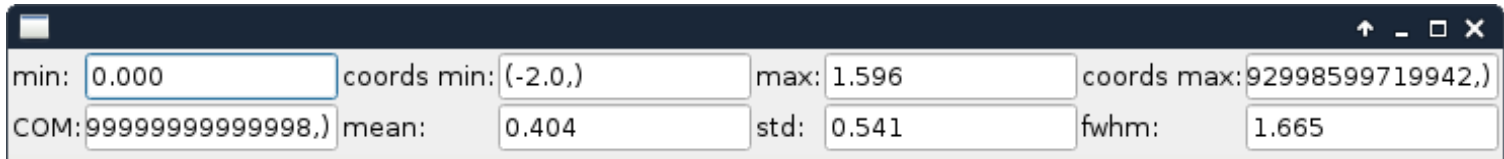
- `silx.gui.plot.tools.roi`: Added line and symbol style settings
- `silx.gui.plot.ComplexImageView`: Allow to edit phase colormap





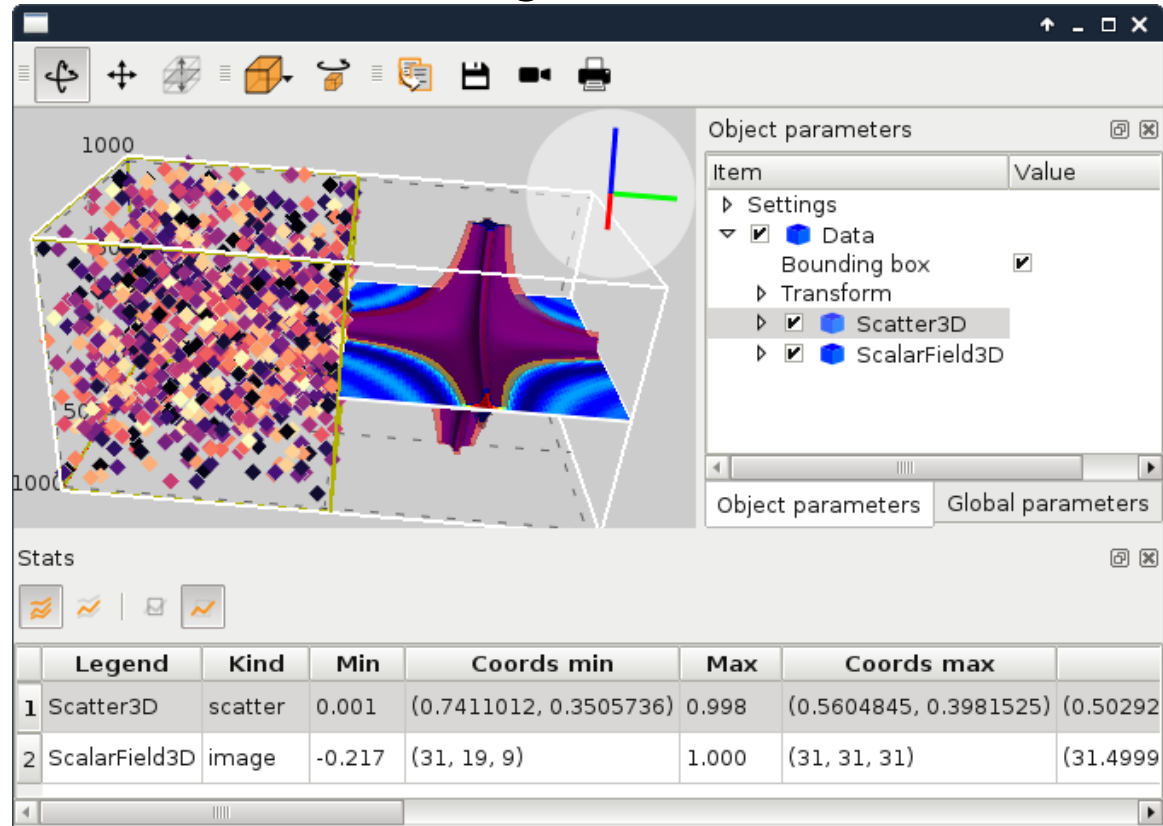
silx.gui.plot.StatsWidget

- Add *BasicGridStatsWidget*



- Add support of *silx.gui.plot3d* widgets

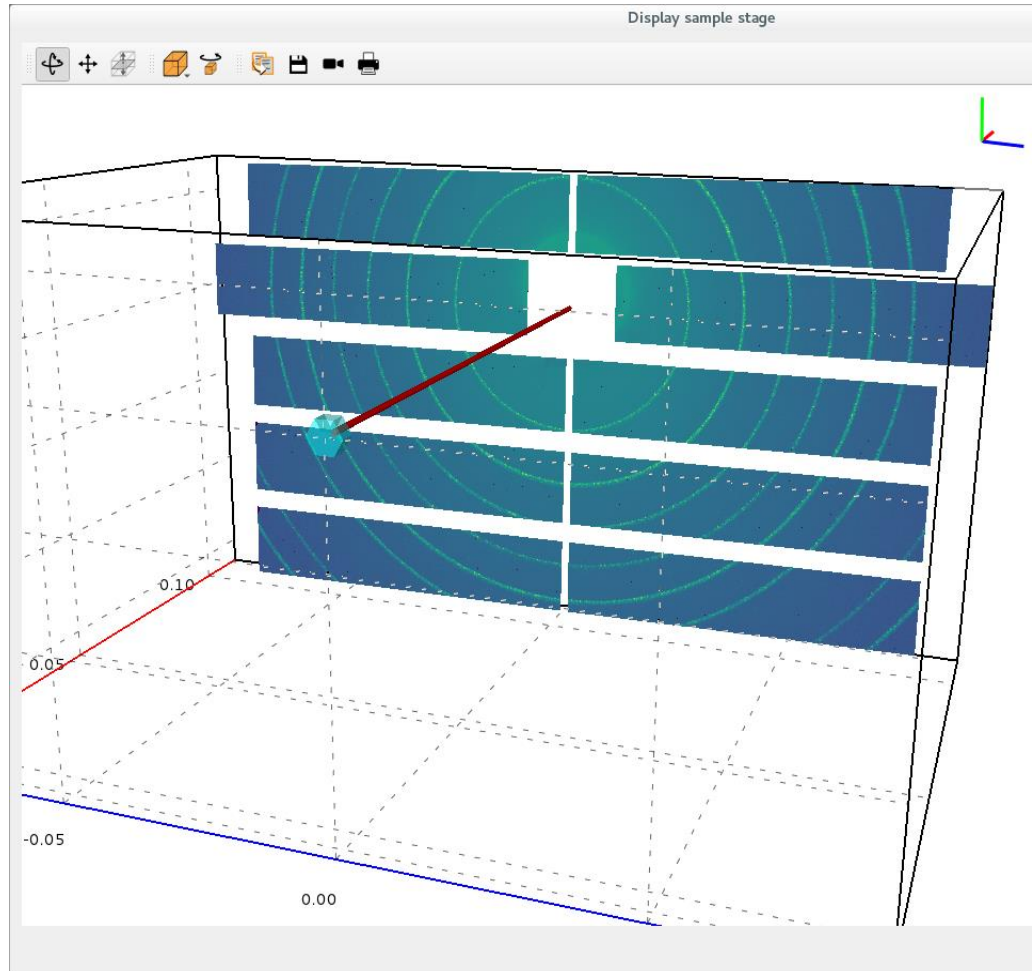
- Improvements
 - On-demand mode
 - Optimizations





silx.gui.plot.plot3d: ColormapMesh

- Add *ColormapMesh* item to the *SceneWidget*



- @alexmarie78:
 - Code clean-up
 - Remove build warnings
 - Clean-up optional imports of h5py and Fabio
- @titusjan:
 - PlotWidget foreground color settable



Dependency Changes

- *PySide*: End of support (use *PyQt5*)
- *numpy*:
 - Last version of silx supporting v1.8.2 (i.e., Debian8)
 - Next version will require v1.12.1 (i.e., Debian9 or Debian8 backports).
- *Python2*: Dropped by the end of 2019
(as Python does: <https://pythonclock.org/>)



This talk

- Introduction
 - Novelties (version 0.9.0)
- **Status of silx (version 0.9.0)**
- 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 / Output
 - Support for SPEC, HDF5 and image formats
- math:
 - least squares fit with constraints, isosurface calculations, histograms, fft,...
- opencl: Optimize the use of GPU (FBP, registration, median filter, ...)
- third-party: External utilities
- utils: Internal utilities
- sx: Convenience module for interactive use



- Easier installation of all dependencies:

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

- Windows standalone application



Container of icons, openccl 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")

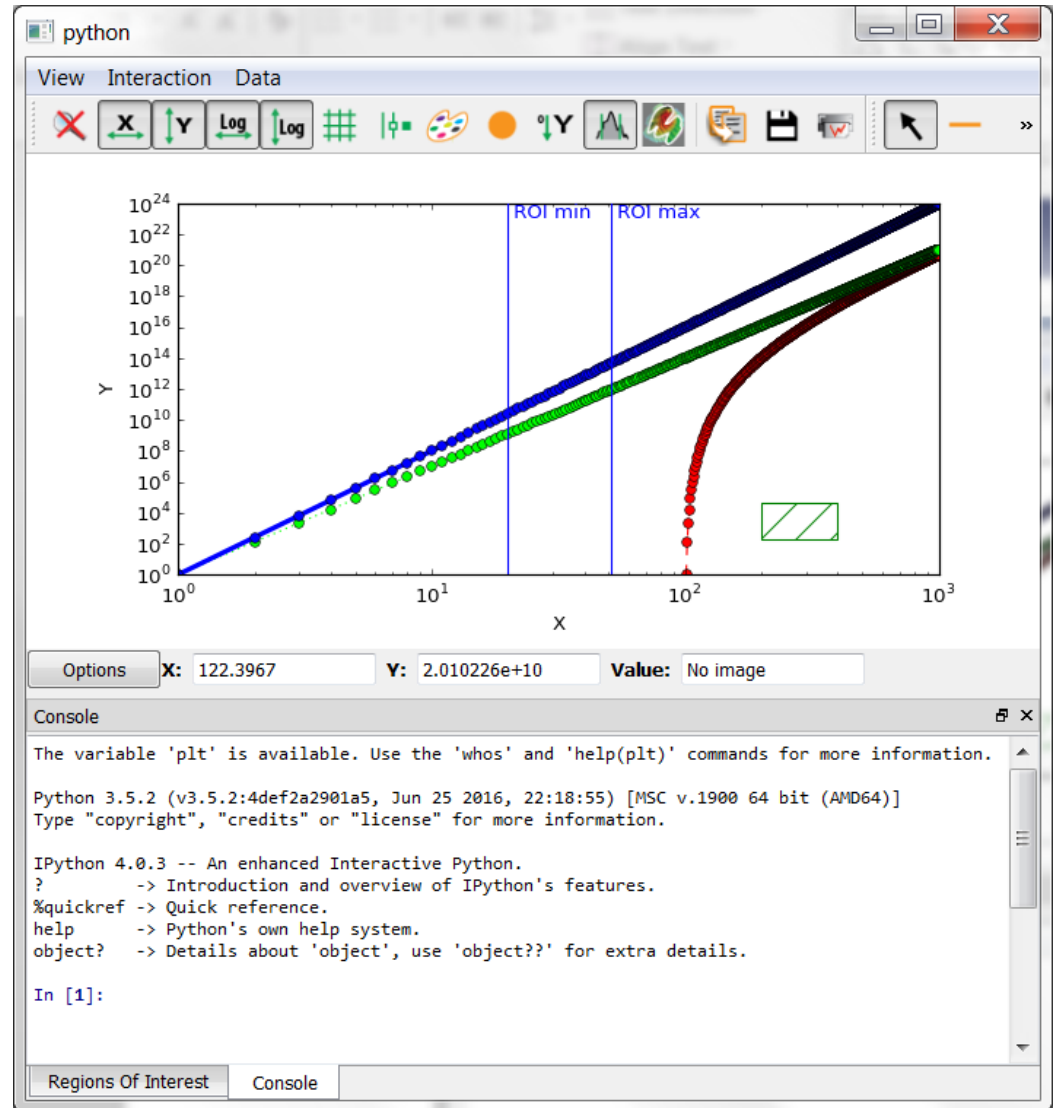
import silx.openccl.utils
filename = silx.openccl.utils.get_cl_file("pyfai:openccl/integrate")

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



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





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.))
```




- Add signals on *PlotWidget* items (i.e. curves, images, markers,...) notifying updates: *sigItemChanged*
- Get all items in the plot: *getItems*
- Follow plot content update through signals: *sigItemAdded* and *sigItemAboutToBeRemoved*



silx.gui: Plot 1D

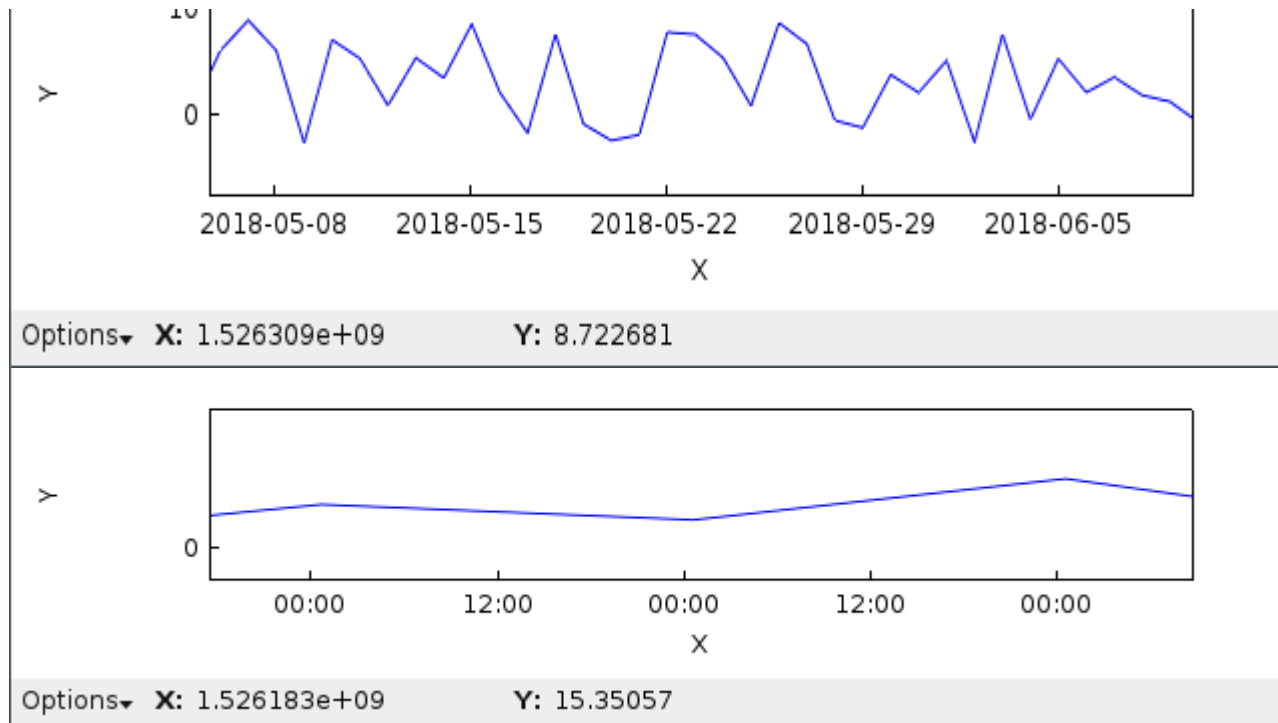
- Visualize 1D data
- Apply ROIs on them
- Control the plot via an interactive console
- Fitting capabilities
- Object oriented API



silx.gui.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>





PlotWidget axis

- Provide a plot axis API

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

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

- Constraints on axes

```
axis.setLimitsConstraints(minPos, maxPos)
```

```
axis.setRangeConstraints(minRange, maxRange)
```

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

- Helper to synchronize axes

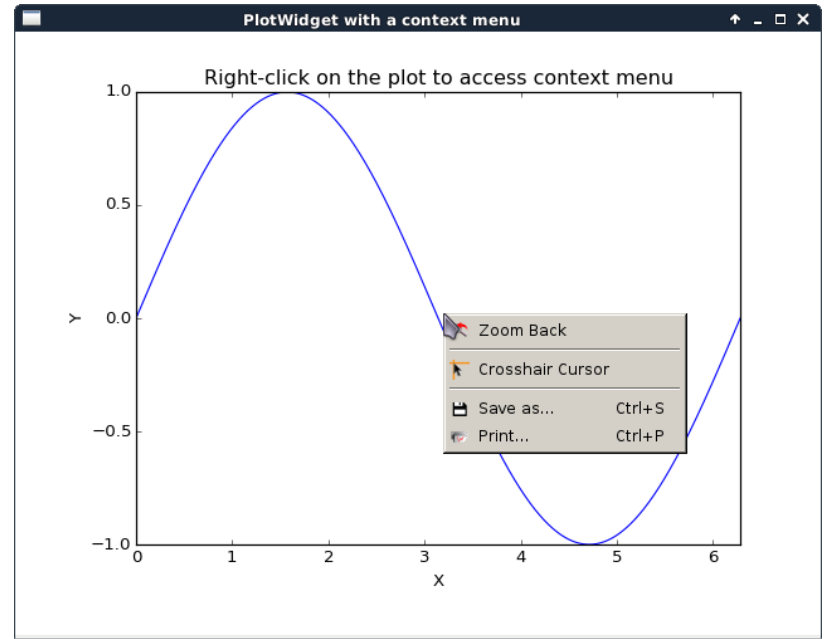
```
from silx.gui.plot.utils.axis import SyncAxes
```

```
sync = SyncAxes([plot1.getXAxis(),  
                 plot2.getXAxis(),  
                 plot3.getXAxis()])
```

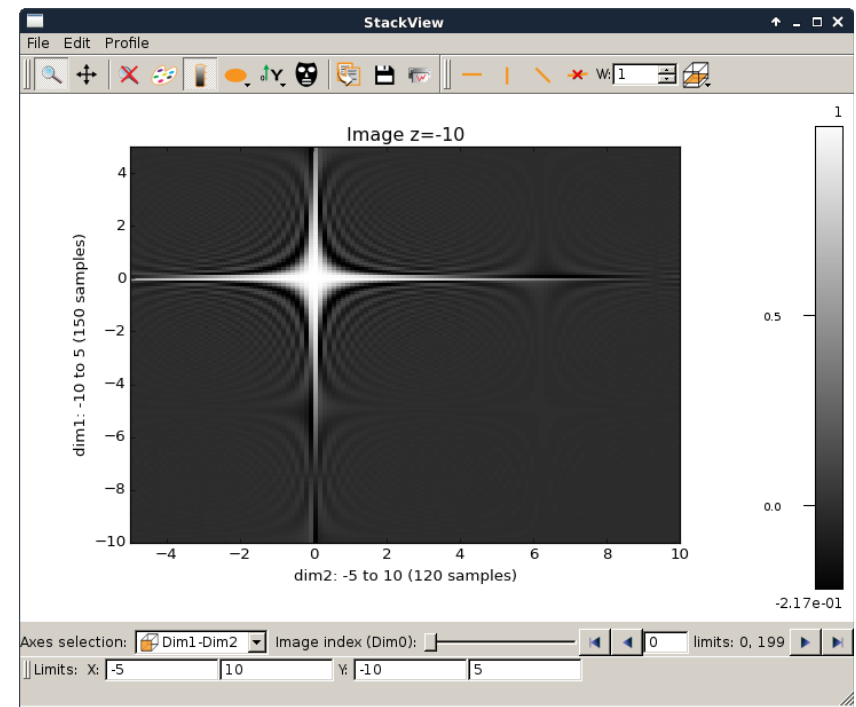
- A demo is available at *examples/syncaxis.py*



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



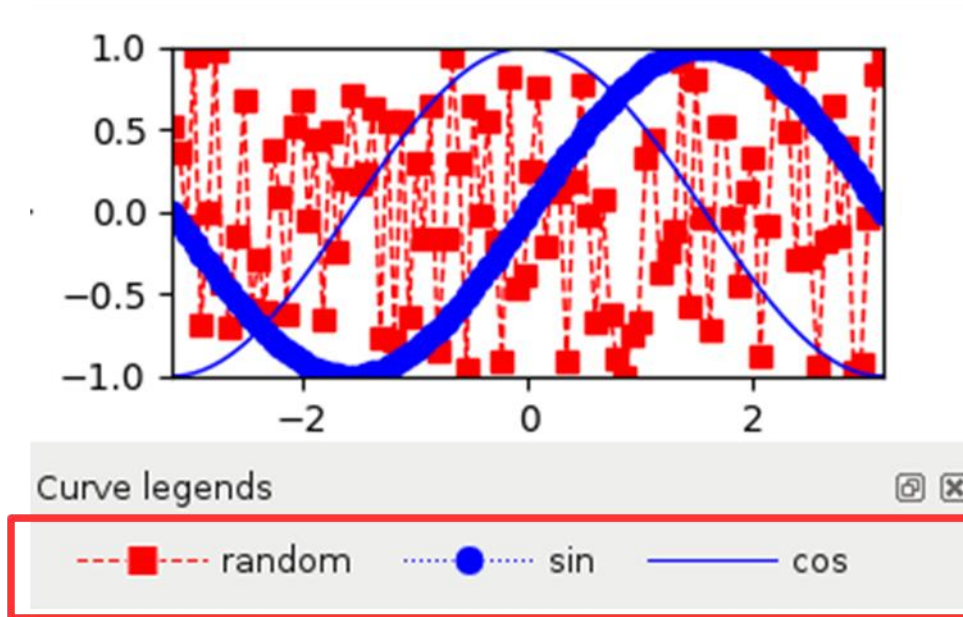
- PlotWindow, Plot2D
- Add colorbar





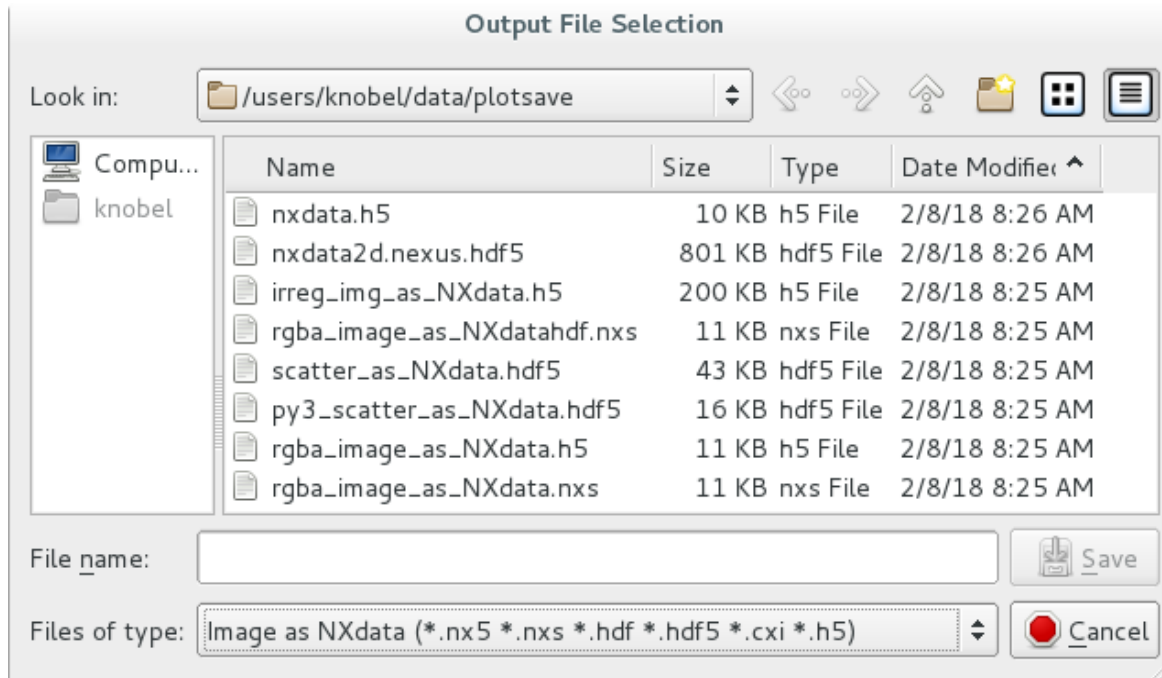
silx.gui.plot: CurveLegendWidget

`silx.gui.plot.CurveLegendWidget:`



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

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



- 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



- 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)

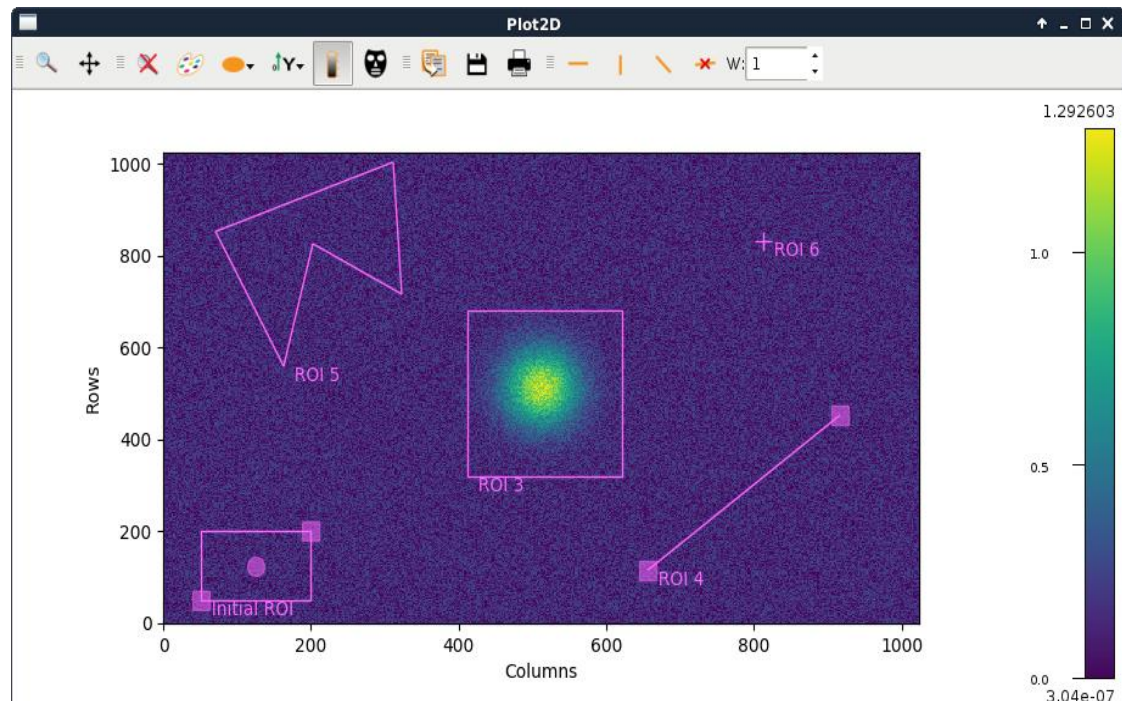


Interactive Regions of Interest

- `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`





- **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()
```



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.



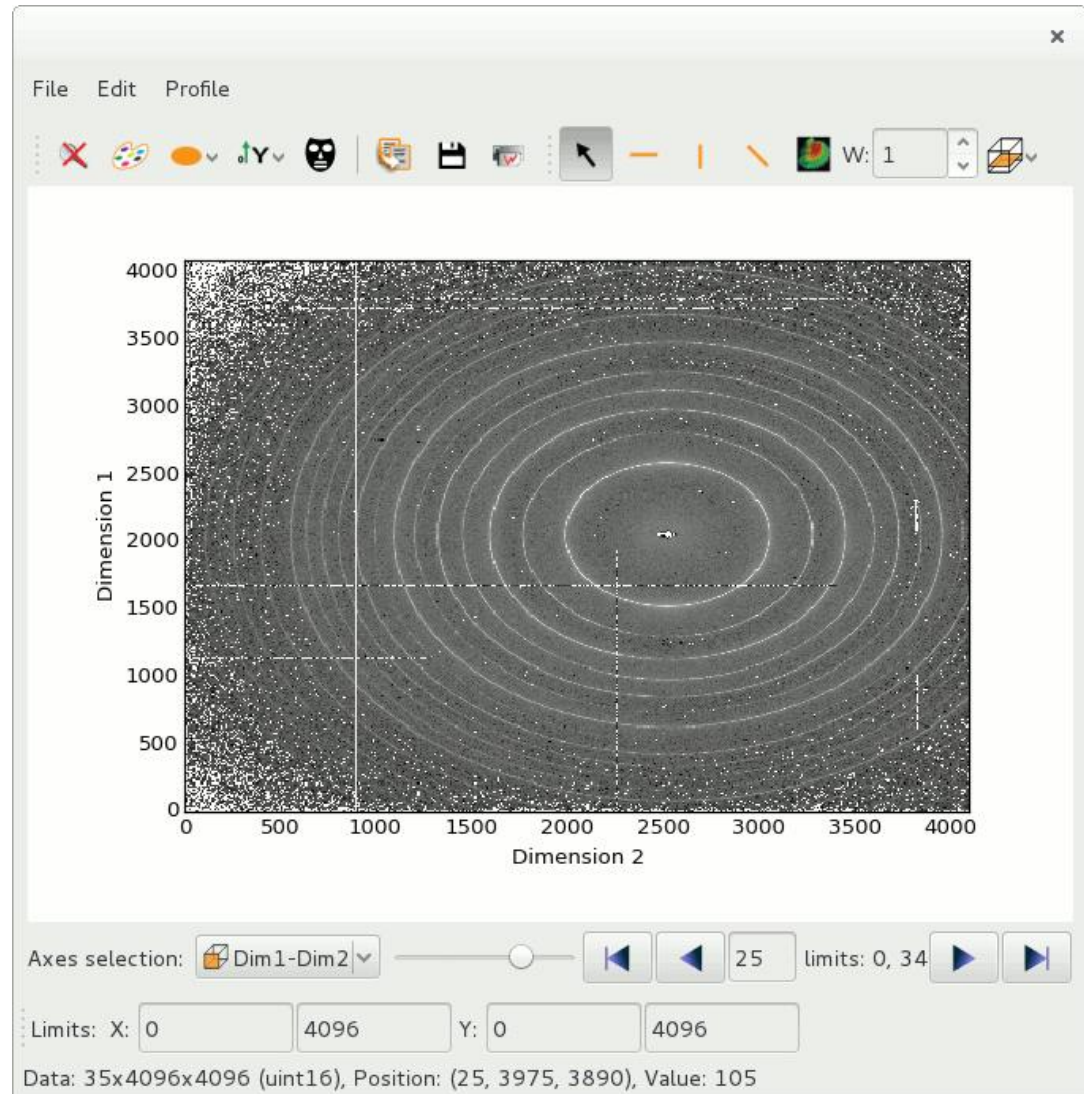


silx.gui: Qt / numpy image conversion

`silx.gui.utils:`

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

- 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

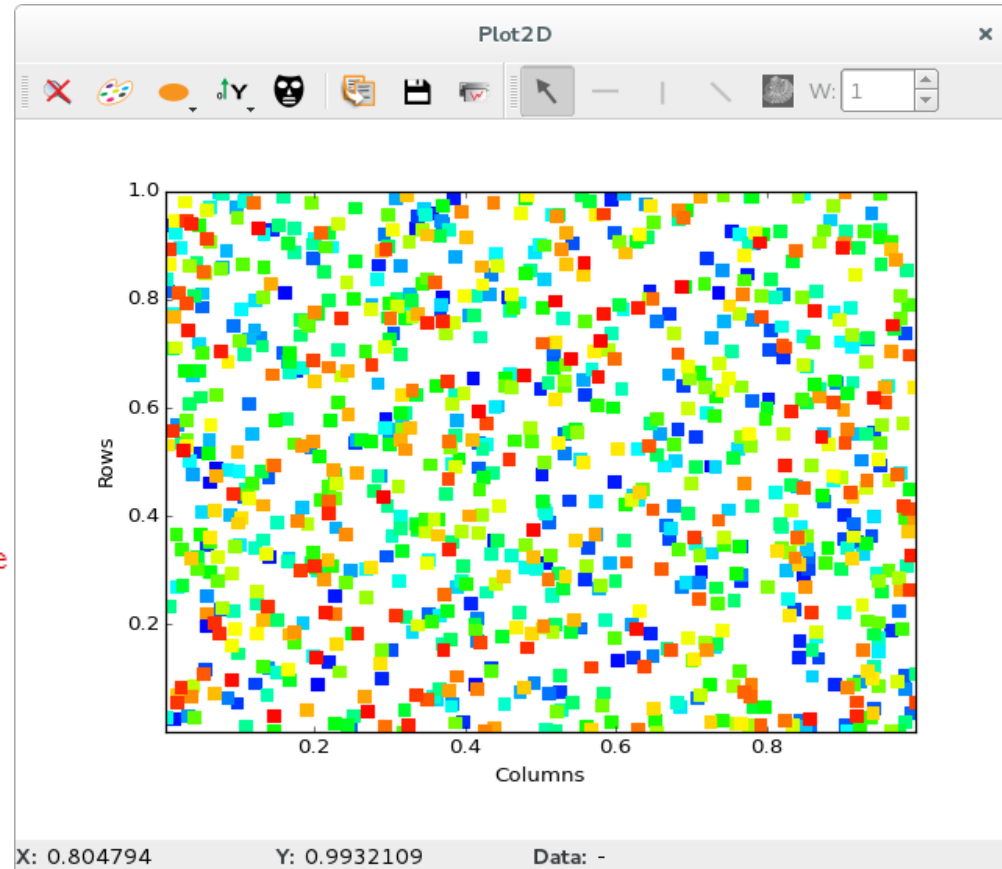
```
import numpy
import sys
from silx.gui import qt
from silx.gui.plot import Plot2D

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,})

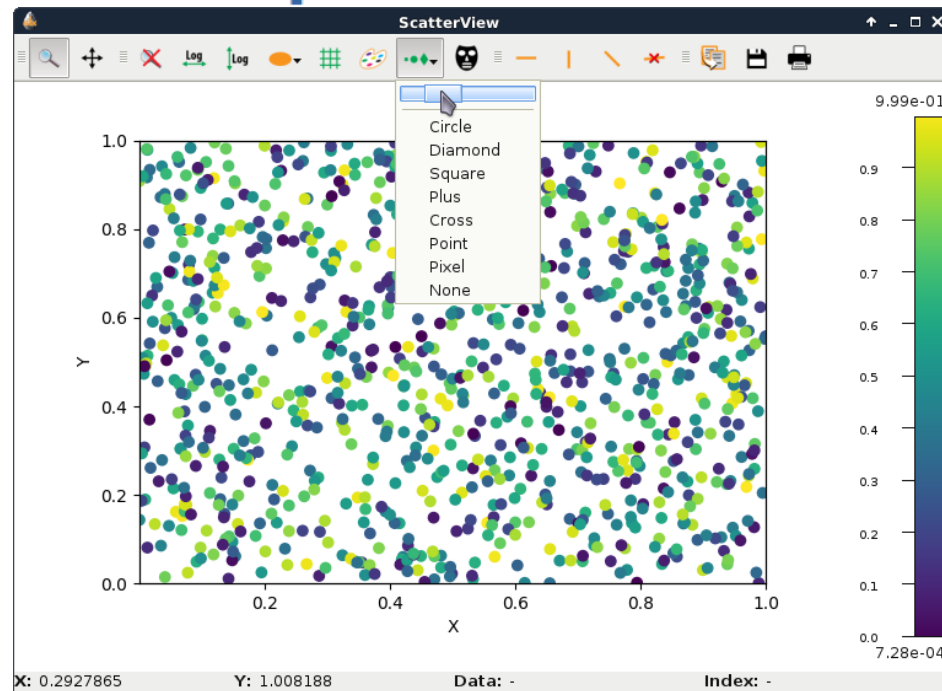
win.resetZoom()
win.show()
sys.exit(app.exec_())
```





ScatterView: Features

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

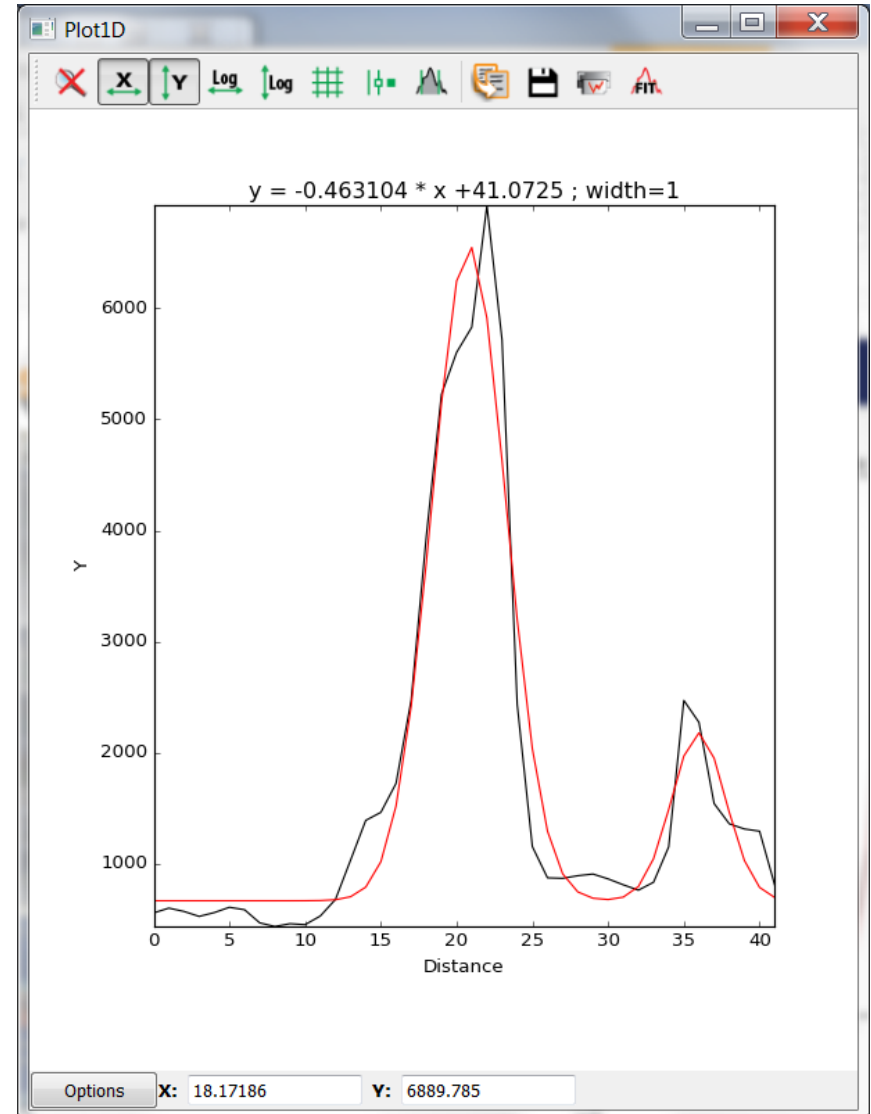
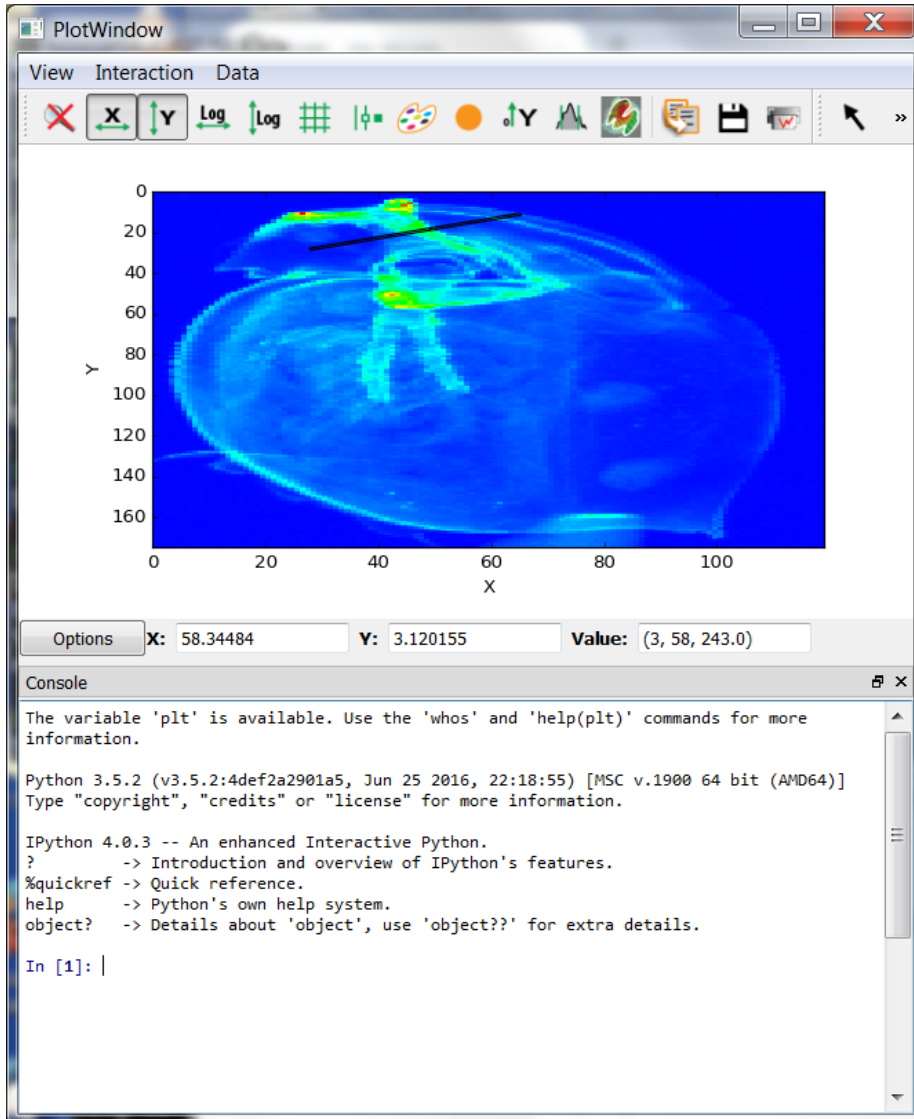


```
from silx.gui.plot.ScatterView import  
ScatterView
```

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



Full-featured widgets





Full-featured Widgets

ImageView

File Edit Profile

W: 1

Image

Y

X

1e8

+3.793e8

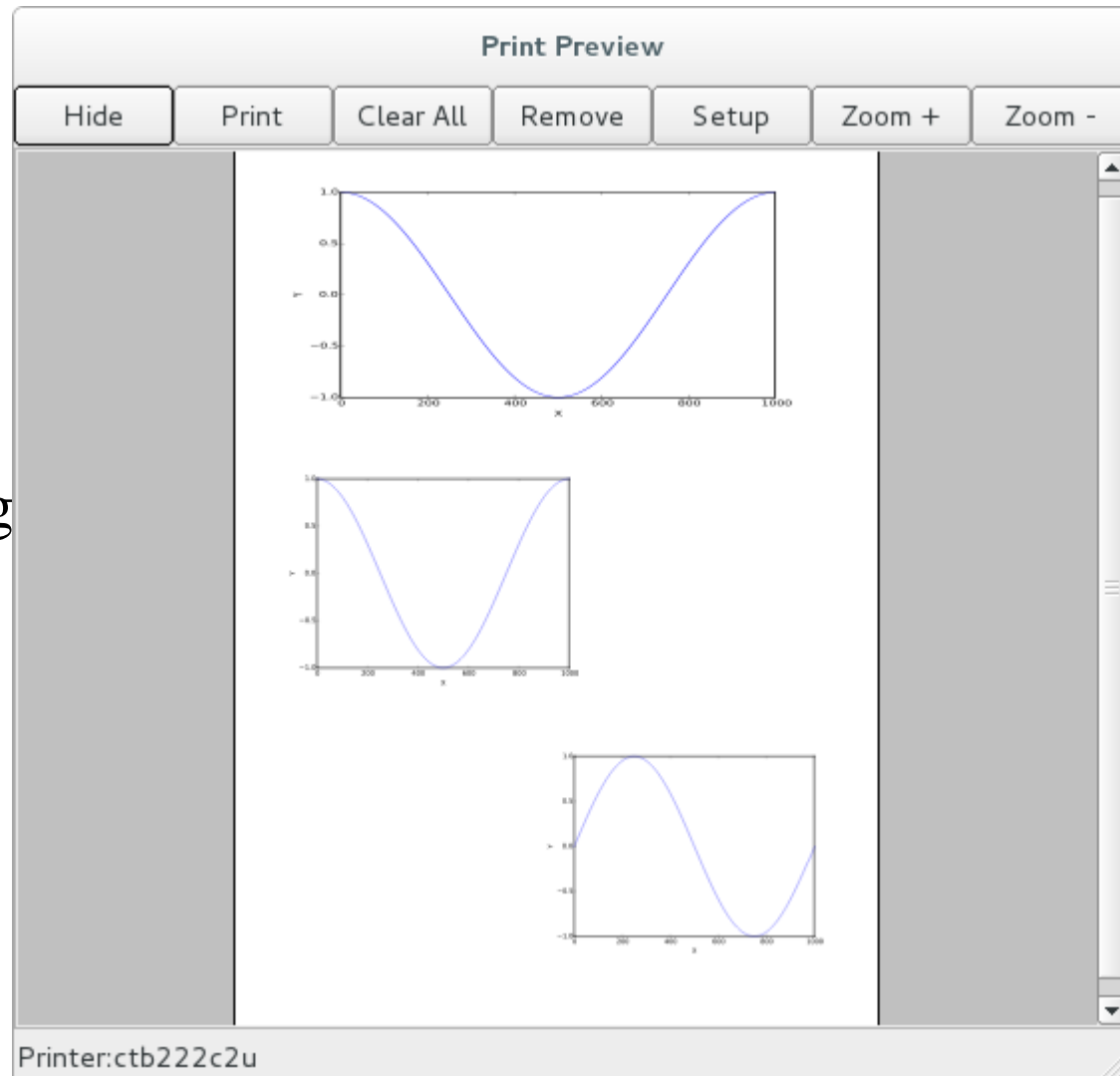
Limits: X: 202.667 755.2 Y: 225.466 889.499

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 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.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

The screenshot shows a window titled 'ArrayTableWidget' with a slider at the top. The slider is positioned at 4, with 'limits: 0, 7' displayed to its right. Below the slider are two dropdown menus: 'Rows dimension' set to 0 and 'Columns dimension' set to 2. The main area contains a table with 8 columns (0-7) and 8 rows (0-7). The cells contain scientific notation values, with alternating colors (green and blue) for the columns.

	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

- 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

periodicTable.py

PeriodicTable | PeriodicList | PeriodicCombo

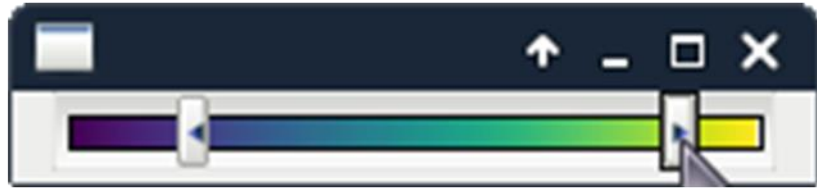
Ni(28) - nickel

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.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.



Stats Widget

Deal with:

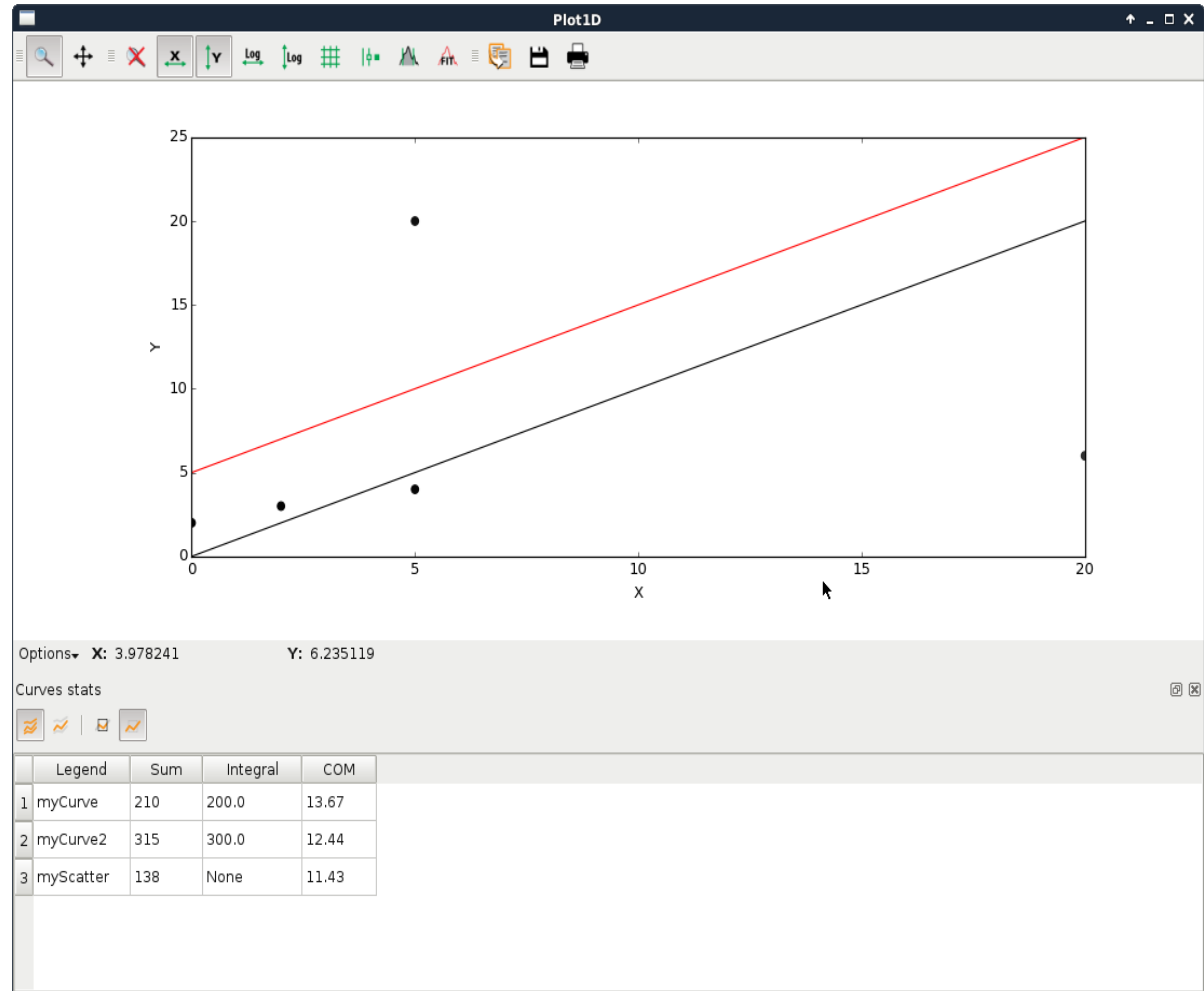
- curves
- Images
- Scatters
- Histograms

Can calculate on:

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

Example:

examples/plotStats.py





OpenGL in *plot3d* and *plot*

- Support for Qt \geq 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



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

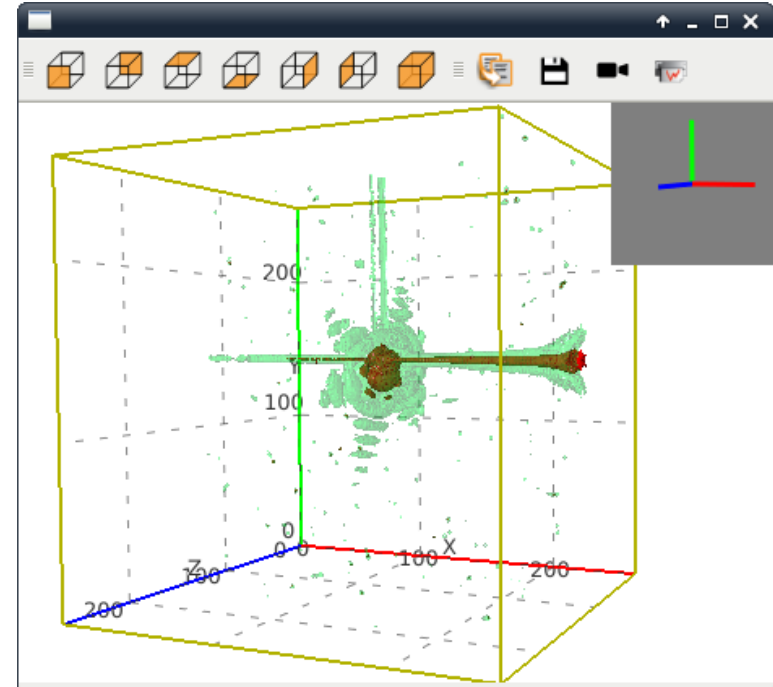
- Usage: Set argument `backend='gl'` in widget constructor for: `PlotWidget`, `PlotWindow`, `Plot1D`, `Plot2D`, `StackView`, `ImageView`
- Example:

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




Silx 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

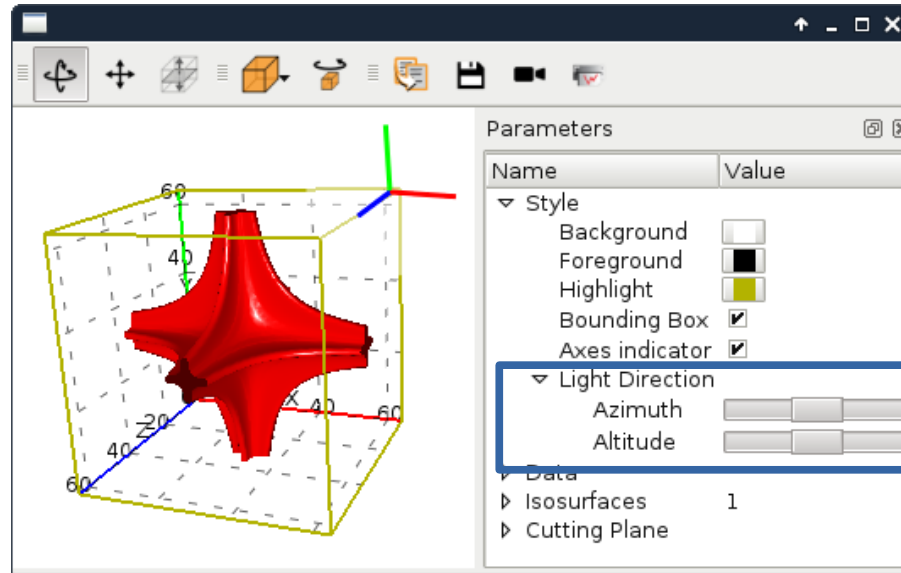


Name	Value
▼ Style	
Background	<input type="checkbox"/>
Foreground	<input type="checkbox"/>
Highlight	<input type="checkbox"/>
▶ Data	
▼ Isosurfaces	1
▶ <input checked="" type="checkbox"/> ■	10
	<input type="button" value="+"/> <input type="button" value="-"/>
▼ Cutting Plane	
<input type="checkbox"/> Visible	
Colormap	gray
Normalization	linear
Orientation	XZ-Plane
Autoscale	<input checked="" type="checkbox"/>
Min	
Max	



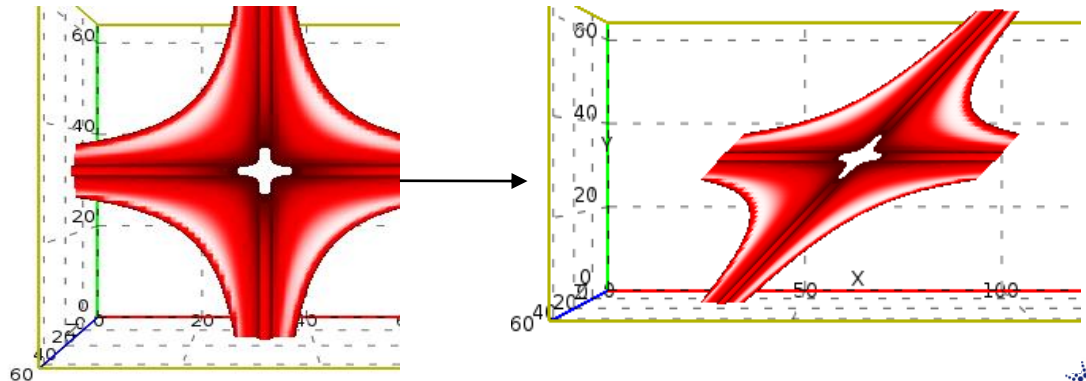
silx.gui.plot3d: ScalarFieldView

- Add light control



- 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.)))
```

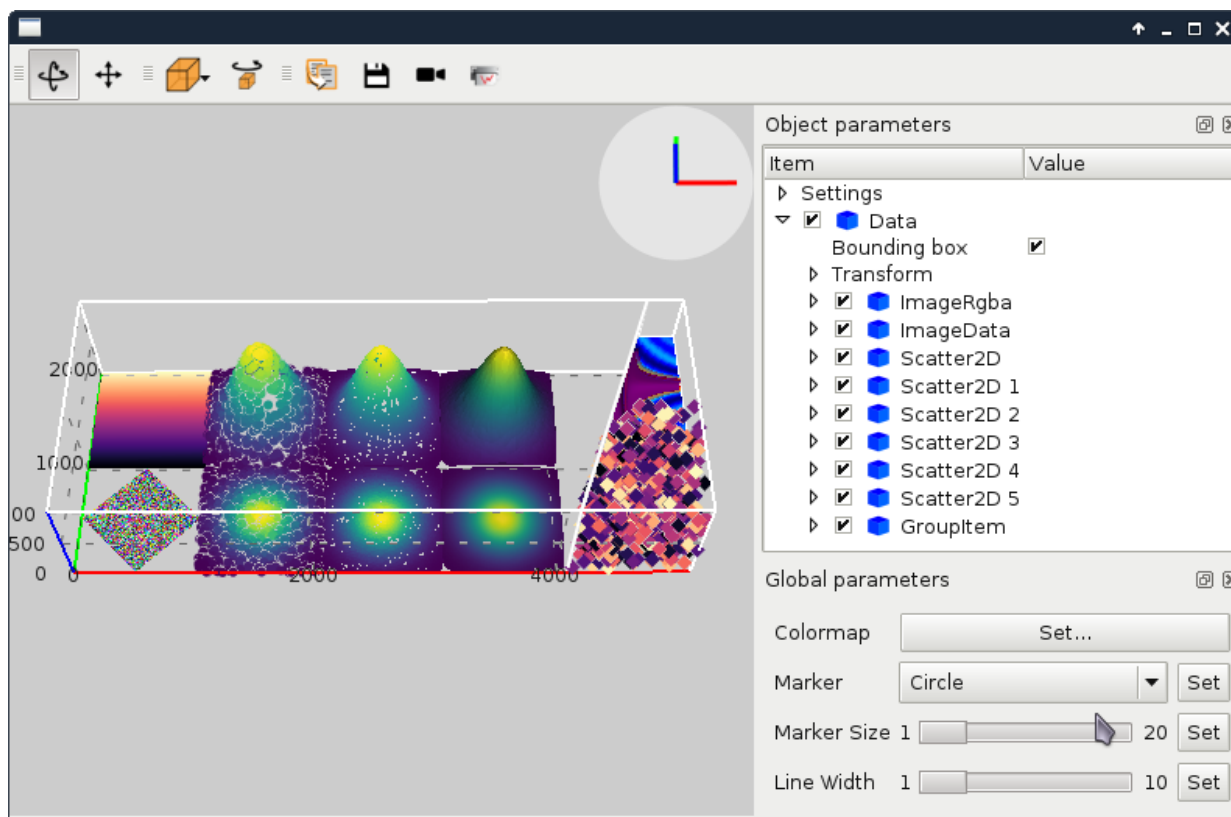




silx.gui.plot3d: Scene widgets

General purpose 3D visualization widget and associated tools:

- Goal: Replacement candidate for PyMca OpenGL tab

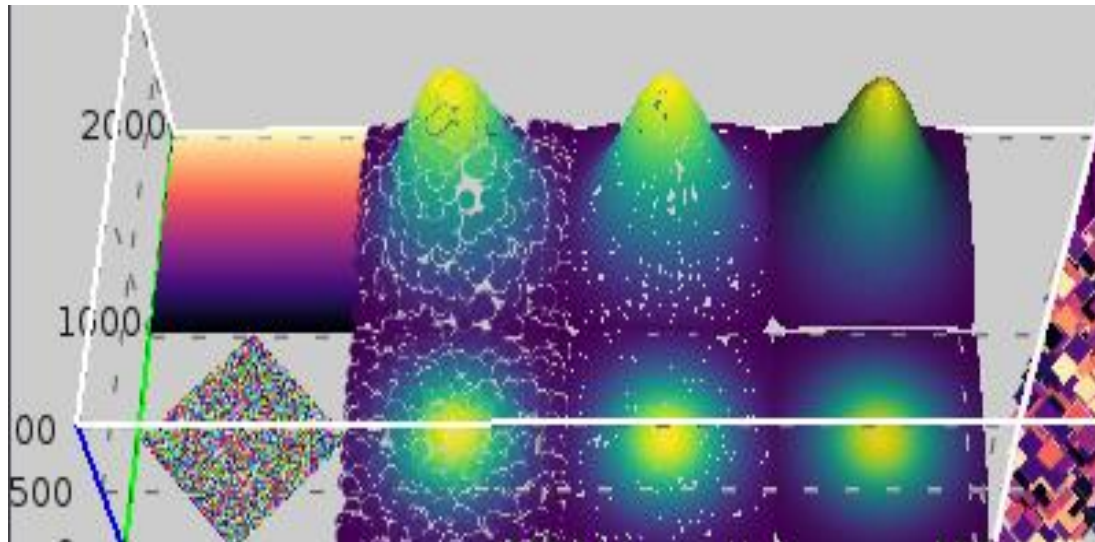




silx.gui.plot3d: Scene available items

`silx.gui.plot3d.items:`

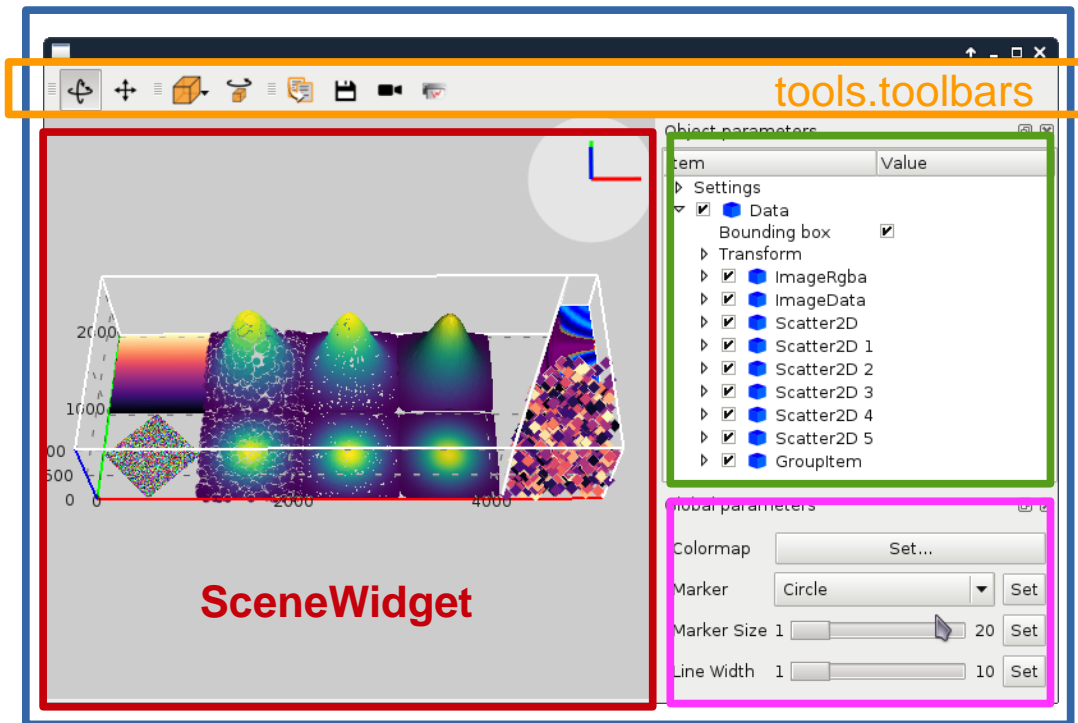
- **Images:** `ImageData`, `ImageRgba`
- **Scatter plots:** `Scatter2D`, `Scatter3D`
- **Scalar fields (with a cut plane and isosurfaces):** `ScalarField3D`
- **A clipping plane:** `ClipPlane`
- **3D meshes:** `Mesh`
- **Groups:** `GroupItem`, `GroupWithAxesItem`





silx.gui.plot3d: Scene widgets structure

SceneWindow



ParamTreeView

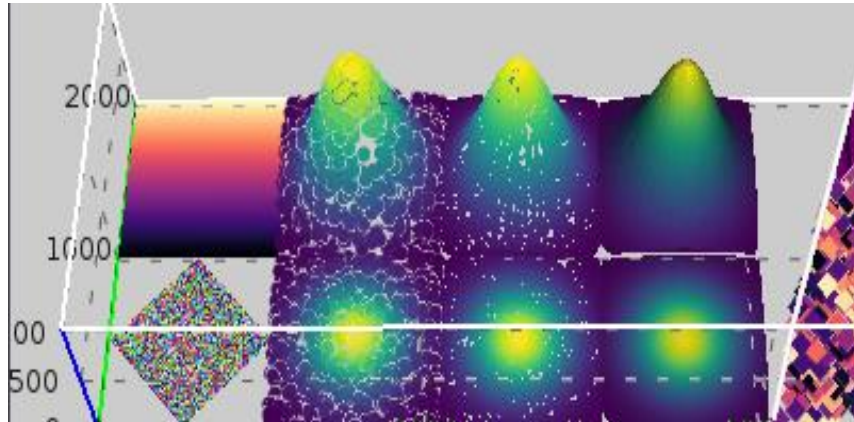
tools.GroupPropertiesWidget



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



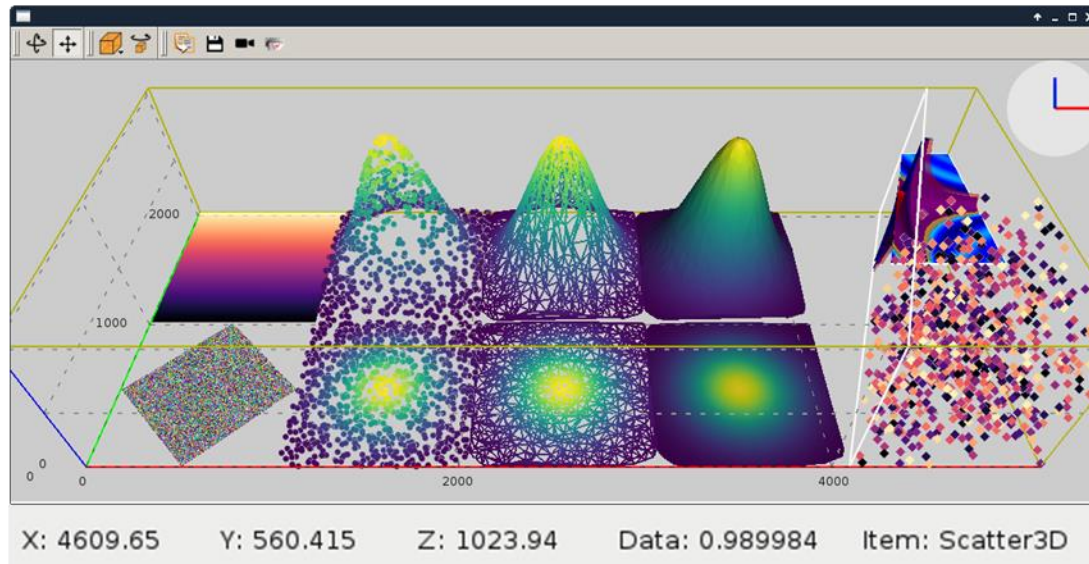
Item	Value
Settings	
Background	
Foreground	
Text	
Highlight	
Axes Indicator	<input checked="" type="checkbox"/>
Light Direction	
Data	<input checked="" type="checkbox"/>
Bounding box	<input checked="" type="checkbox"/>
Transform	
ImageRgba	<input checked="" type="checkbox"/>
ImageData	<input checked="" type="checkbox"/>
Scatter2D	<input checked="" type="checkbox"/>
Bounding box	<input type="checkbox"/>
Transform	
Mode	Points
Height map	<input type="checkbox"/>
Colormap	
Marker	Circle
Marker size	<input type="text"/>
Line width	
Scatter2D 1	<input checked="" type="checkbox"/>
Scatter2D 2	<input checked="" type="checkbox"/>
Scatter2D 3	<input checked="" type="checkbox"/>
Scatter2D 4	<input checked="" type="checkbox"/>
Scatter2D 5	<input checked="" type="checkbox"/>
GroupItem	<input checked="" type="checkbox"/>
Bounding box	<input type="checkbox"/>
Transform	
ClipPlane	<input checked="" type="checkbox"/>
Scatter3D	<input checked="" type="checkbox"/>
ScalarField3D	<input checked="" type="checkbox"/>



silx.gui.plot3d: PositionInfoWidget

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

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





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

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

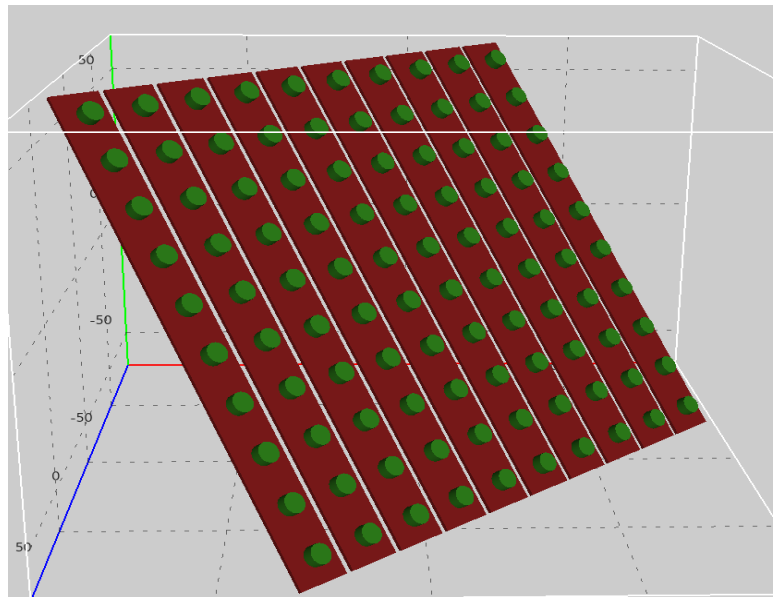
Implementation choices:

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



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:
 - Add interaction mode
 - Selection/edition of Region of Interest (line, box)
- 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.

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')
```



Median Filter (GPU)

silx.opencil.medfilt2d

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

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

```
img = ascent().astype("float32")  
%timeit filters.median_filter(img, (55,55))
```

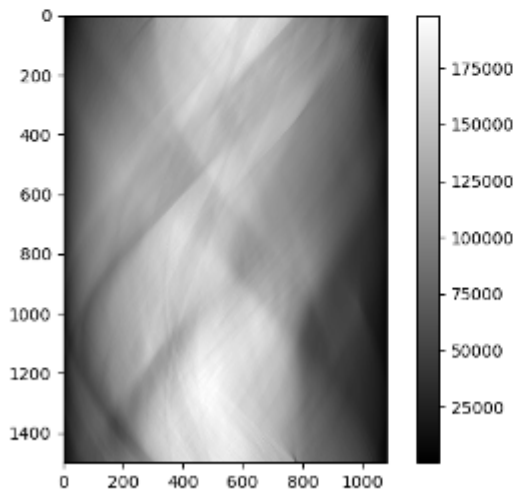
```
import silx.image  
%timeit silx.image.medfilt2d(img, (55,55))
```

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



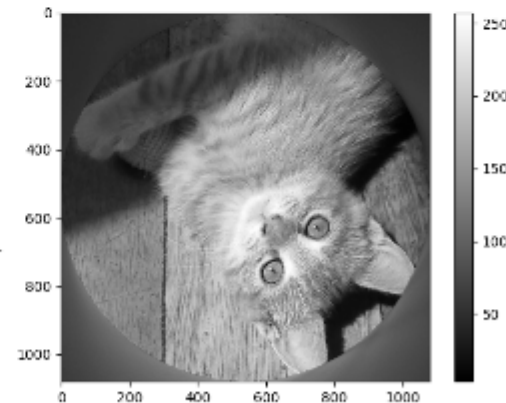
Filtered Back Projection in silx

- 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

FBP
→



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')

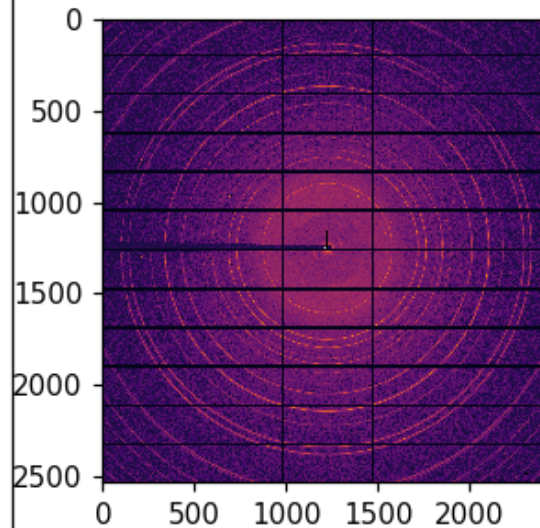
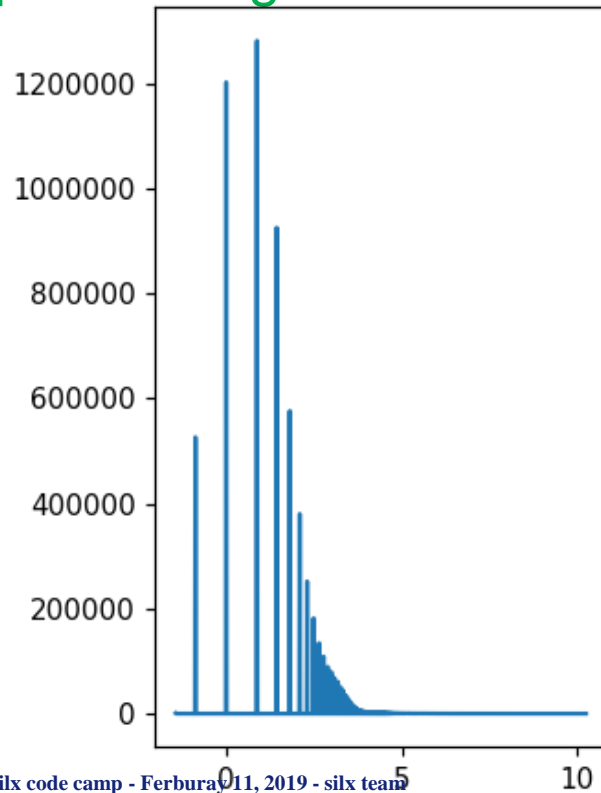
# Allocate the memory for volume reconstruction
num_sinos = sinograms_stack.shape[0]
reco = np.zeros((num_sinos,) + tomo_geometry.shape)
# Reconstruct
for i in range(num_sinos):
    reco[i] = tomo.fbp(sinograms_stack[i])
```





CoDec : Byte offset for CBF processing on GPU

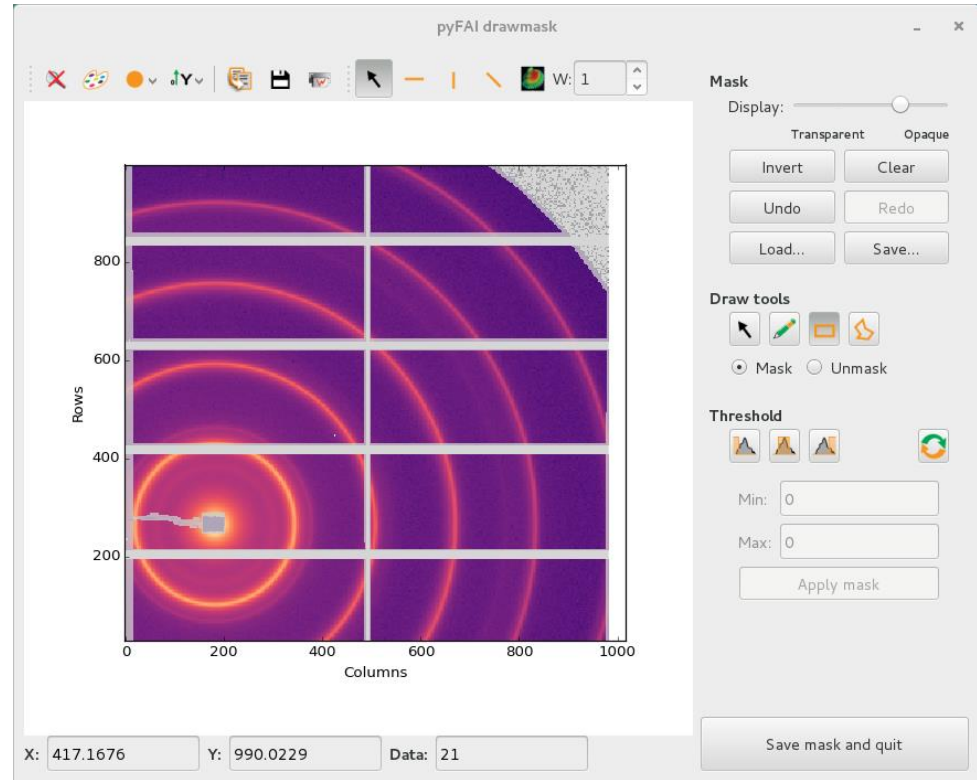
- `silx.opencl.codec.byte_offset`
 - OpenCL-based CBF compression
 - 10x speed-up for compression/decompression of CBF streams
 - Compatible with the new Image processing framework
 - Compatible with pyFAI azimuthal integration
- Accepted in J. Synchrotron Radiation
<https://doi.org/10.1107/S1600577518000607>





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



- Marching Squares
- Median Filter
 - GPU accelerated via OpenCL

- **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`

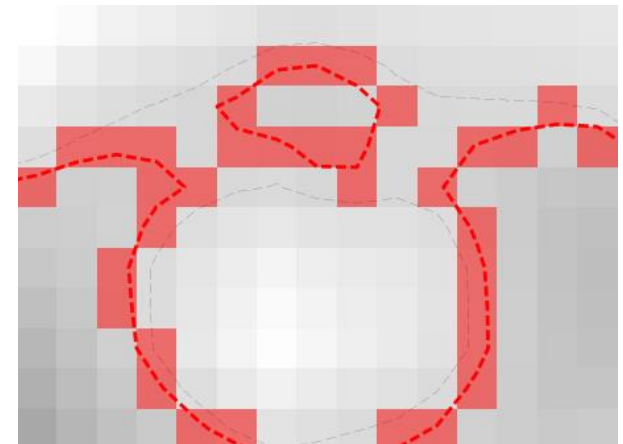
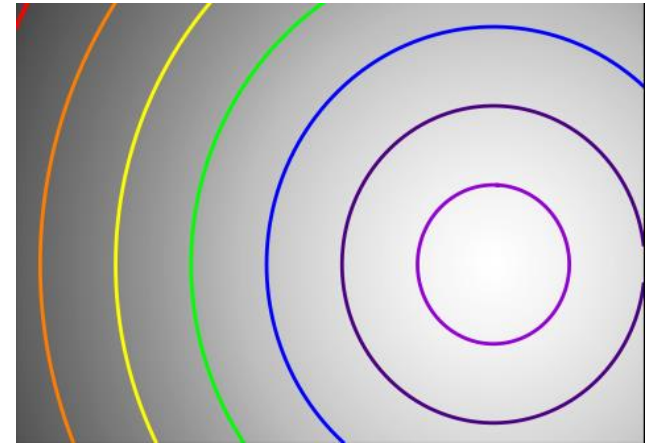


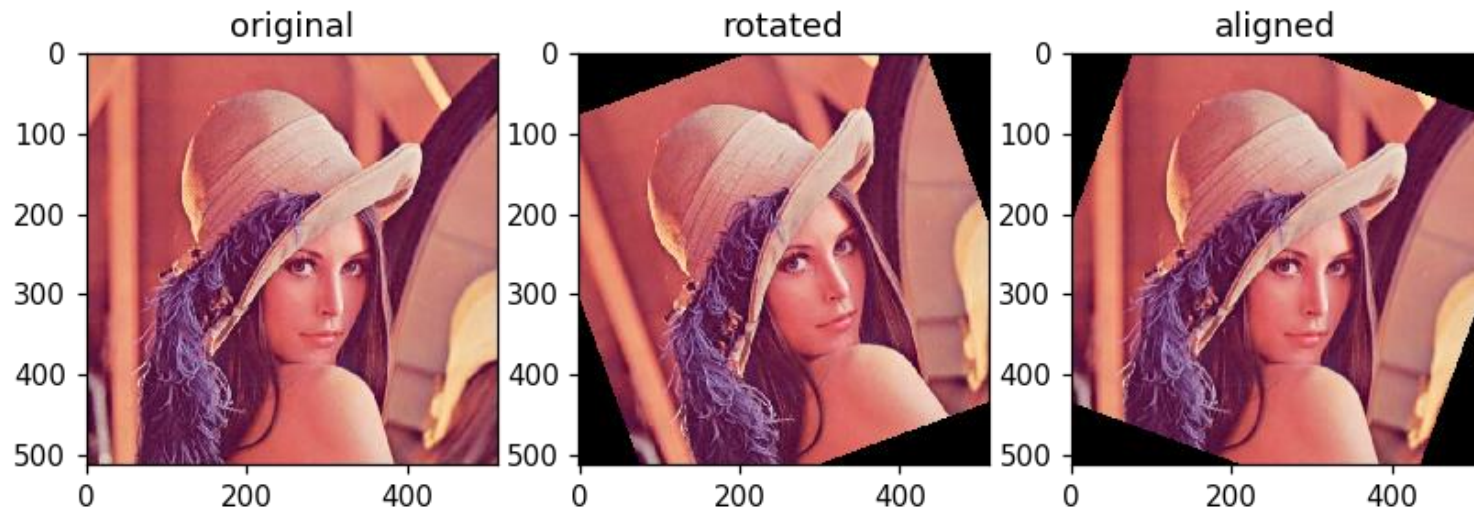


Image processing on opengl devices (GPU)

- 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

- 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>



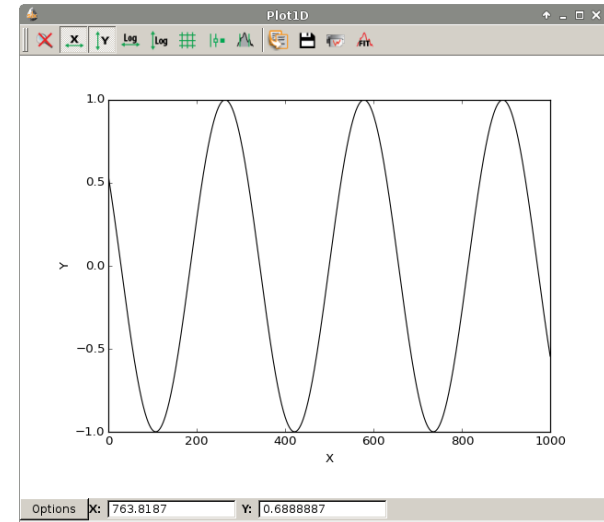


silx.sx: a module to simplify interactive use

pylab like module on steroids

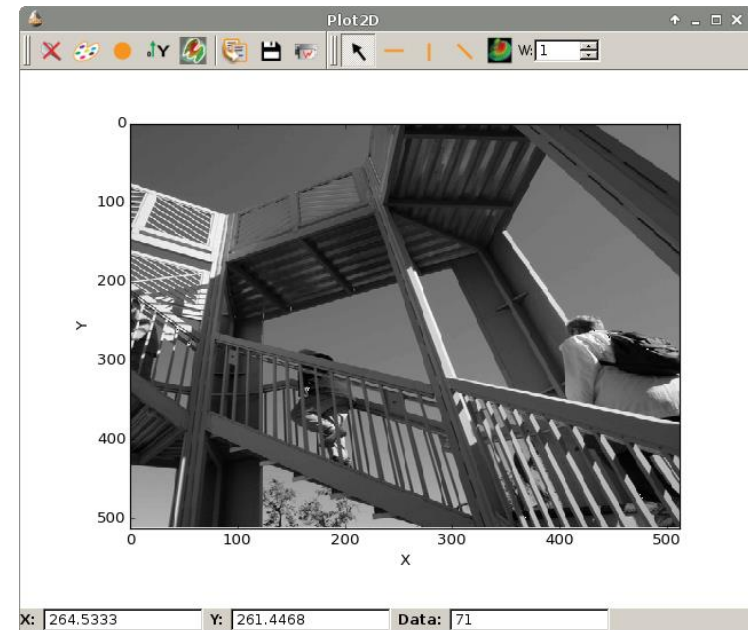
- 1D plotting: ROI, fitting & printing

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



- 2D display: intensity, mask, profile

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





- 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 \leftrightarrow HDF5 and octave \leftrightarrow 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



- 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*,...)



● 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)`



- 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



- 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
```

The screenshot shows the silx HDF5 viewer interface. The main pane displays a tree view of the file structure. The root node is 'ch09__mca_multiframe.h5'. Underneath it is a folder 'scan_0', which contains a folder 'instrument', which in turn contains a folder 'detector_0'. The 'detector_0' folder contains a 'data' node (highlighted in blue) and an 'others' folder. The 'data' node is a 3D array of float32 type with a shape of 71 x 80 x 2000. The 'others' folder contains four 1D data nodes: 'DCM_Energy' (float32, 71), 'Date' (string, 71), 'FocalLength' (float32, 71), and 'MCA a' (float32, 71).

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
```



- 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 limits: 0, 9

Dimension 1

Dimension 2

HDF5
 Curve
 Image
 Cube
 Raw
 Image stack

Create HDF5

Async load

Tree options

Enable sorting

Multi-selection

Drop external file

Reorder files

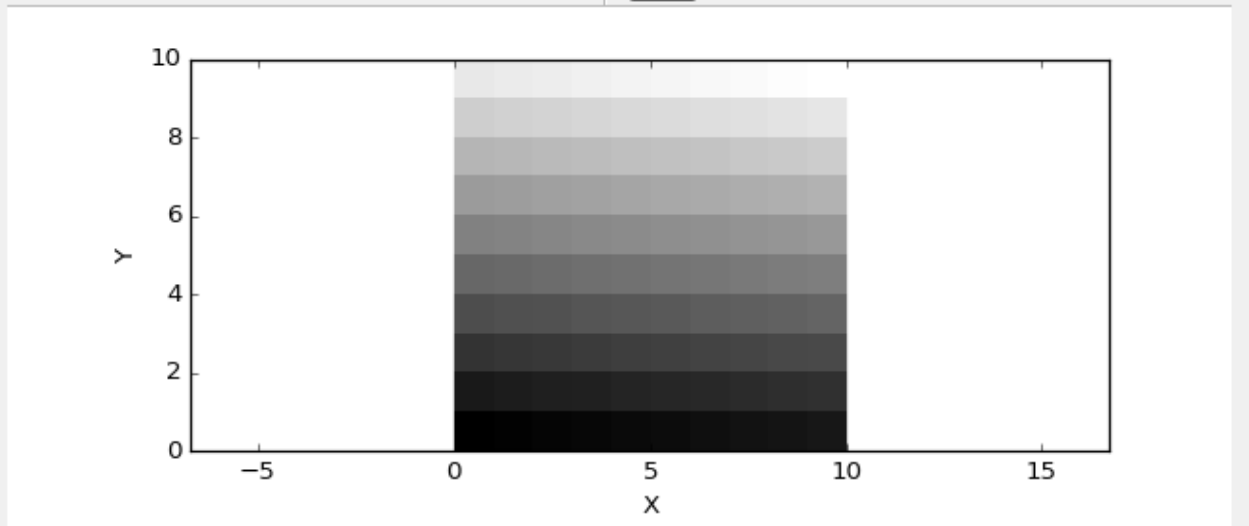
Header options

Auto-size headers

Popup to hide/show columns



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



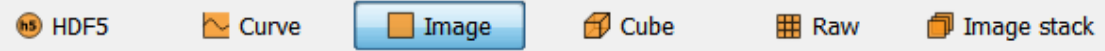
X: 2.606498 **Y:** 9.359807 **Data:** 92

Axis selection

Dimension 0 limits: 0, 9

Dimension 1

Dimension 2



Create HDF5

Async load

Tree options

Enable sorting

Multi-selection

Drop external file

Reorder files

Header options

Auto-size headers

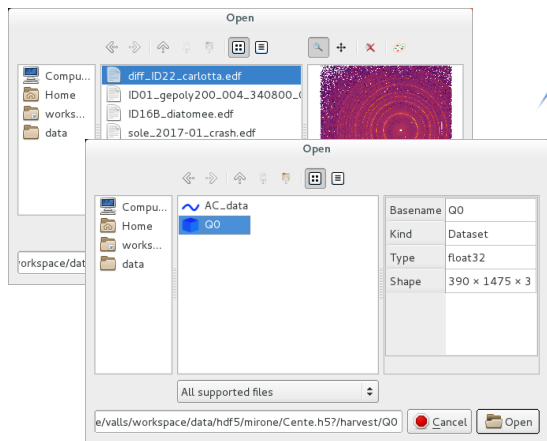
Popup to hide/show columns



Dialog to reach data



File system



silx.gui.dialog

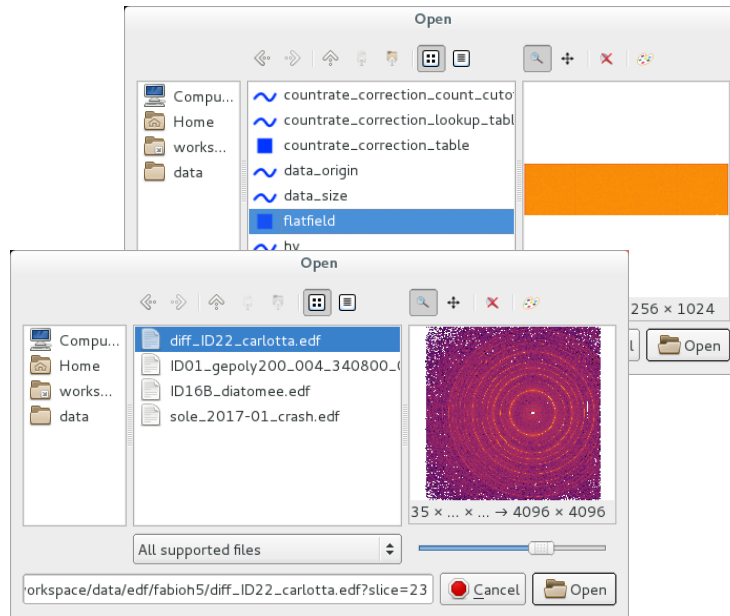
URL



silx.io.open
(h5py-like context)

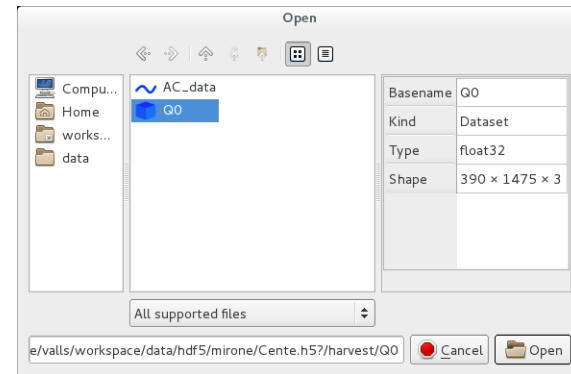


silx.io.get_data
(numpy data)



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*

The screenshot shows the Silx viewer interface. On the left, a file tree for 'test.h5' is displayed, with 'float_2d' selected. On the right, a data table is shown with 6 rows and 3 columns. Below the table, there are 'Axis selection' controls for Dimension 0 (set to 'col') and Dimension 1 (set to 'row'). At the bottom, there are buttons for 'HDF5', 'Curve', 'Image', and 'Raw'.

	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



- 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			

NXdata group /scatters/x_y_scatter

Options X: 0.09893982 Y: 0.4218765

Selector
Dimension 0

HDF5 NXdata

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



NXdata Viewer

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			

NXdata group /images/2D_irregular_data: data

X: 88.20926 Y: 57.95693 Data: -

Selector ✕

Dimension 0

Dimension 1

Displayed data: data[:, :]

HDF5 NXdata

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 ▼



- 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 – Growing family of applications using silx

The image displays a central collage of various silx application windows. A large, semi-transparent 'silx' logo is centered over the collage. The applications shown include:

- TDS2EL2**: A window showing a 2D image and a 3D reconstruction of a sample.
- Crispy**: A window for data processing with options like 'compare the center of rotation' and 'compare the center of rotation for absolute'.
- TomoGUI**: A window for tomographic reconstruction with a 'data source' dropdown and 'center of rotation' input.
- XSOCS**: A window showing a 3D volume rendering of a sample.
- silx view**: A window displaying a 2D image with a color scale.
- Tomwer**: A window showing a 2D image with a color scale.
- PyFAI Calibration**: A window showing a 2D image with concentric rings and a table of picked rings.

Name	Peaks	Ring number
a	956	1
b	357	2
c	358	3
d	265	4
e	230	5
f	225	6
g	79	7
h	36	8
- PyMca**: A window showing a 1D spectrum plot with peaks and a 'Help' section.



<https://github.com/kklmn/ParSeq>

ParSeq — Dummy

1 - currents

- parseq
- doc
- data
 - tst2.hs
 - entry704
 - DCM
 - mono1_... 51
 - definition scalar
 - end_time scalar
 - entry_ide... scalar
 - measure... scalar
 - plot_1
 - program... scalar
 - start_time scalar
 - title scalar
 - user
 - tst2.dat
 - cuo_rxes_La... 103
 - cuo_rxes_00... 14
 - CuO_Int1.fio 14
 - cu2o_rxes_L... 93
 - cu2o_rxes_0... 14
 - Cu2O_Int2.fio 14
 - Cu2O_Int1.fio 14**
 - Cu_rt2.fio 14
 - Cu_rt1.fio 14
 - Cu_Int2.fio 14
 - Cu_Int1.fio 14

data name	I0	I1
metal	4	4
Cu_Int1	✓	✓
Cu_Int2	✓	✓
Cu_rt1	✓	✓
Cu_rt2	✓	✓
oxides	3	3
Cu2O_Int1	✓	✓
Cu2O_Int2	✓	✓
CuO_Int	✓	✓

Options: X: 9537.839 Y: 38391.91

! Comment
%c
EXAFS-Scan started at 28-Mar-2001 21:30:02
Name: cuo_005 from 8950 to 10107.44

energy calibration TODO

type not implemented

E ref 8979.000

E shift at = 0.000

ParSeq dummy

[This is a link to MAX IV](#)

node1

This is heading 3

This is heading 4

This is heading 5

This is heading 6

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur tempus consequat

7 spectra

stop data propagation here (TODO)

combine selected data

stop propagation of contributing data at:

move selected data to a new group

Combine

1 selected spectrum: Cu2O_Int1



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 0.9.0 version
- For this code camp we'll use 0.10.0a, 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/>