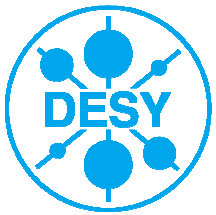
dpdak

directly programmable data analysis kit

Gunthard Benecke
HDRI-PNI
27.02.2012



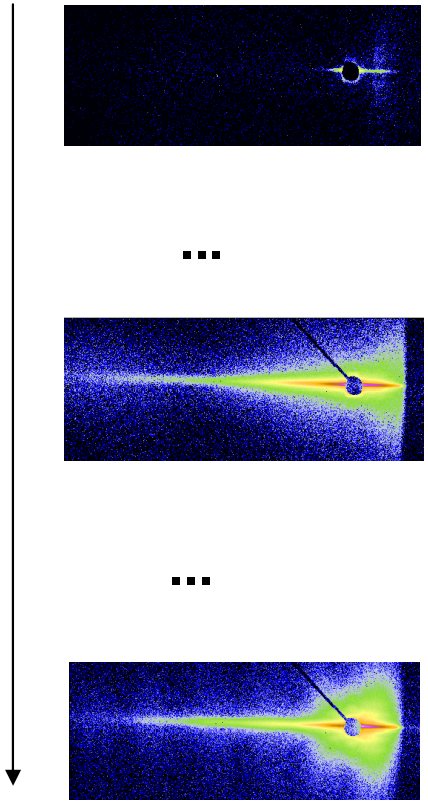
Max Planck Institute
of Colloids and Interfaces

Overview

- Plugin framework for primary online analysis of SAXS/WAXS and GISAXS experiments
- Developed in a DESY and MPIKG cooperation
- Goals of the project:
  - fast analysis allowing to classify the data during the experiment
  - deeper analysis of data after experiemnt
  - extending functionality by plugins

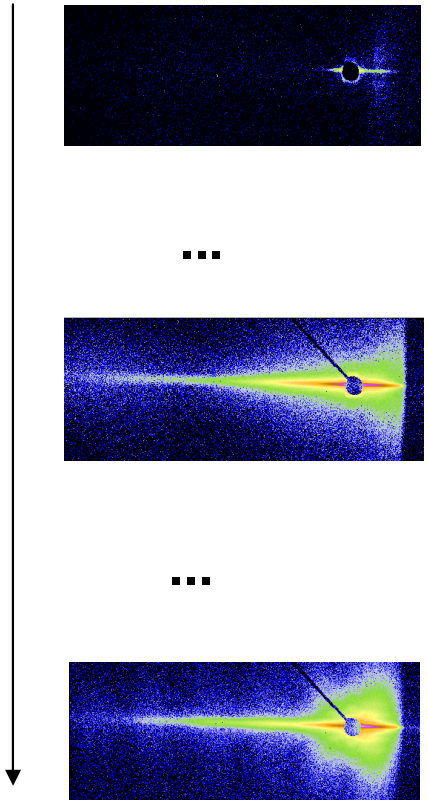
Basic processing schema

- Incoming Data

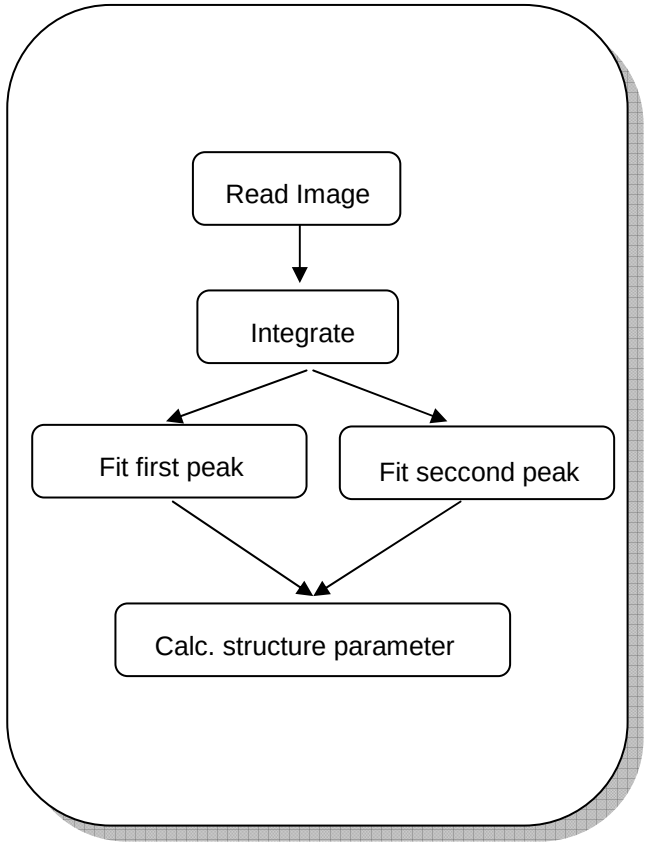


Basic processing schema

> Incoming Data

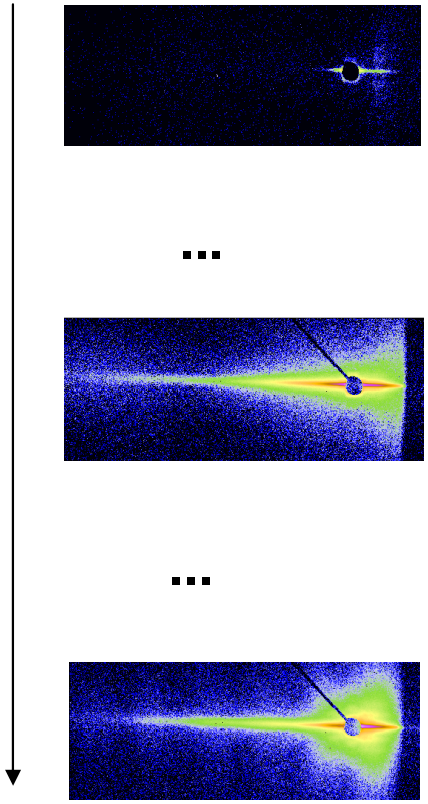


> Data reduction

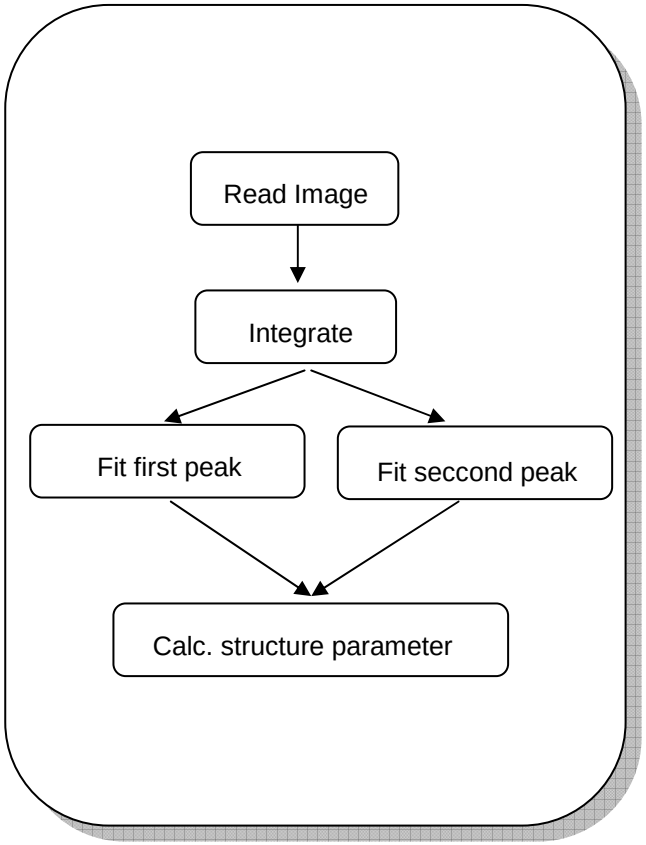


Basic processing schema

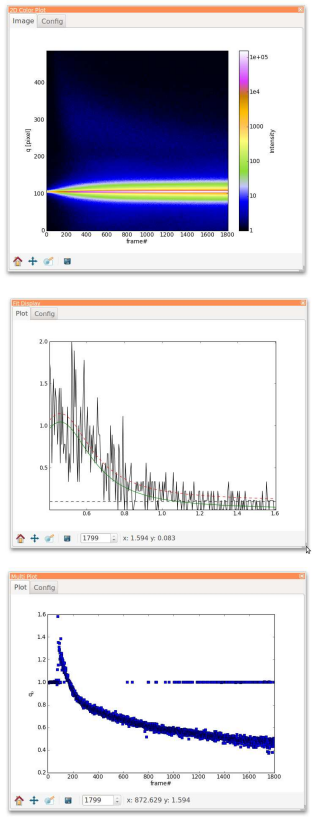
> Incoming Data



> Data reduction



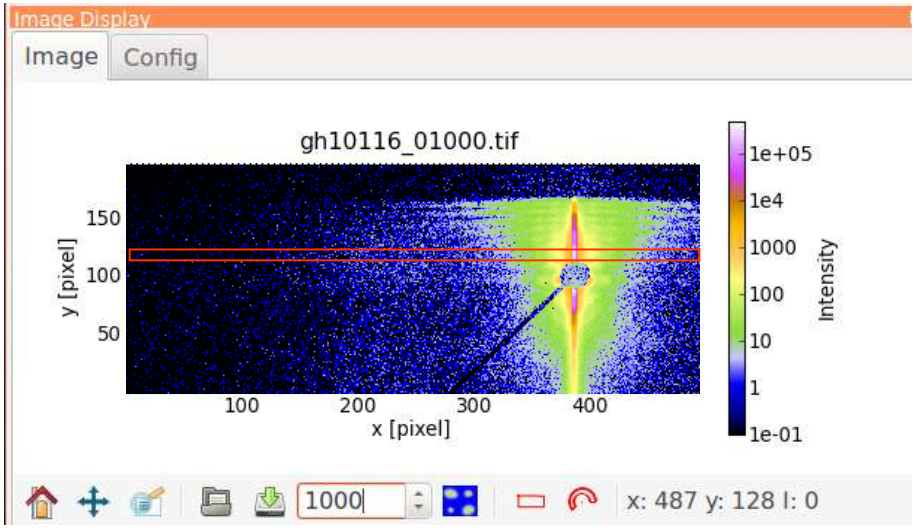
> Display results



Use Case I: Sputter Experiments

- Online (up to a frame every 20ms) plot of cuts show lateral structure

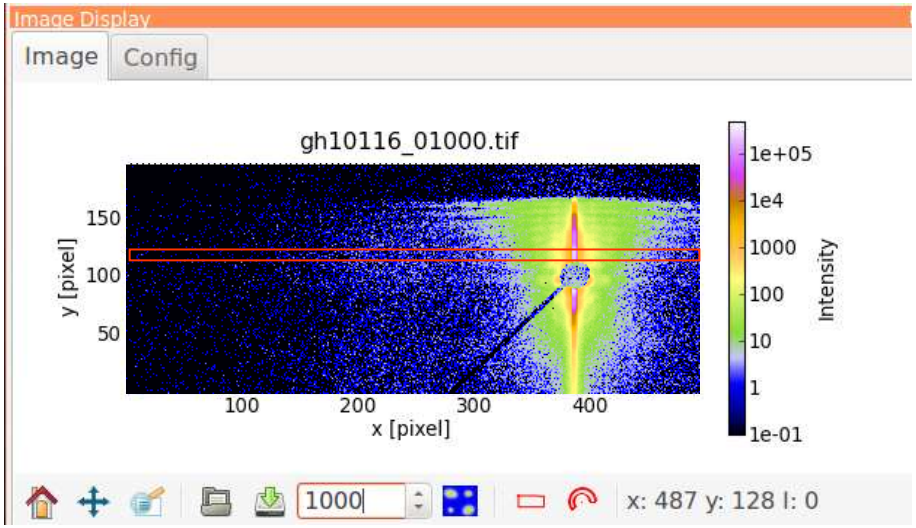
  - Detector image with cut tool



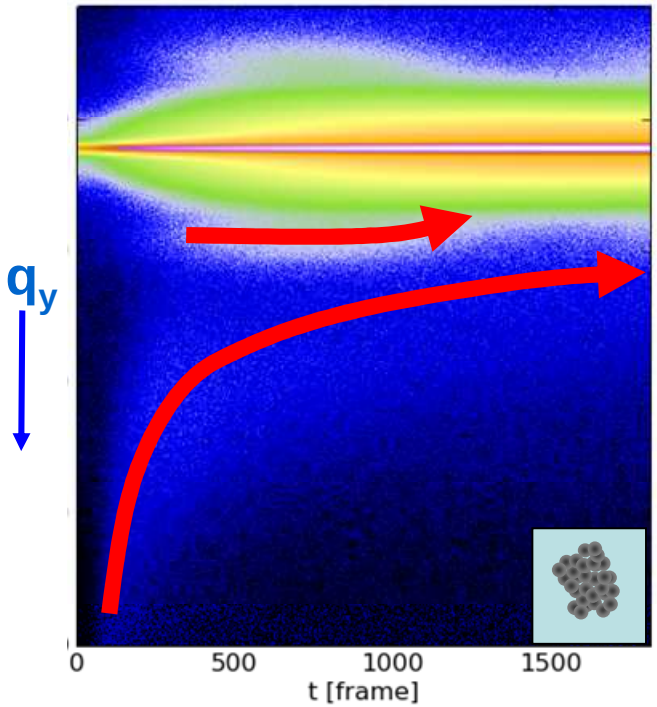
Use Case I: Sputter Experiments

- Online (up to a frame every 20ms) plot of cuts show lateral structure

- Detector image with cut tool

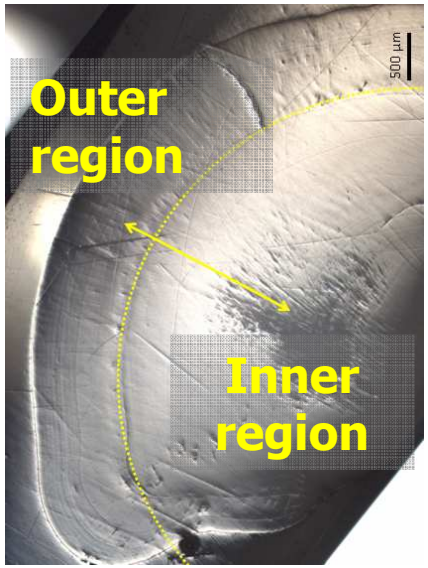


- Broad structure at ~30nm
- Co nanoparticles, R(Co) growing



Use Case II: Crayfish Gastrolith Nanostructure

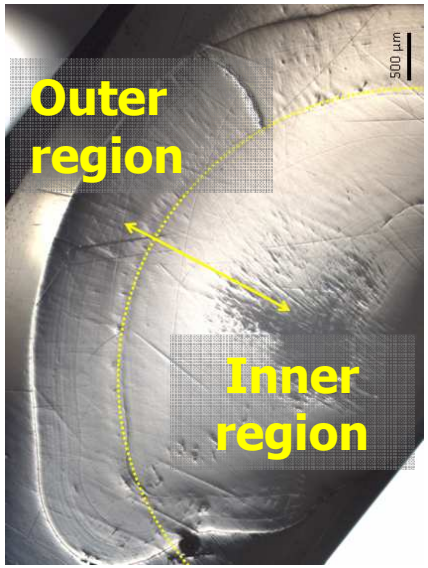
> Sample Data



- 2D scan of 150 mccd detector files
- 10 samples scaned
- 20GB data

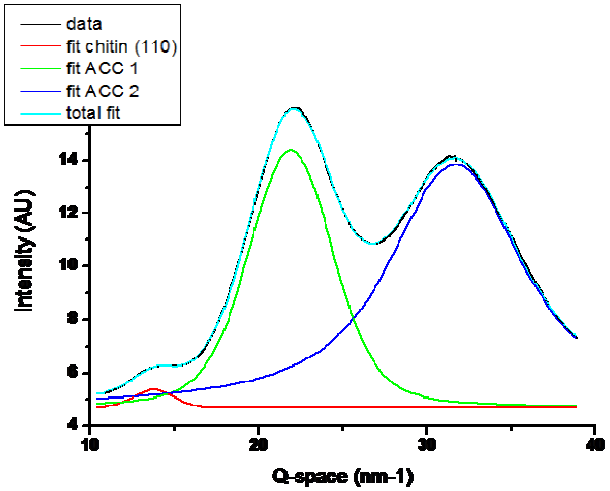
Use Case II: Crayfish Gastrolith Nanostructure

> Sample Data



- 2D scan of 150 mccd detector files
- 10 samples scaned
- 20GB data

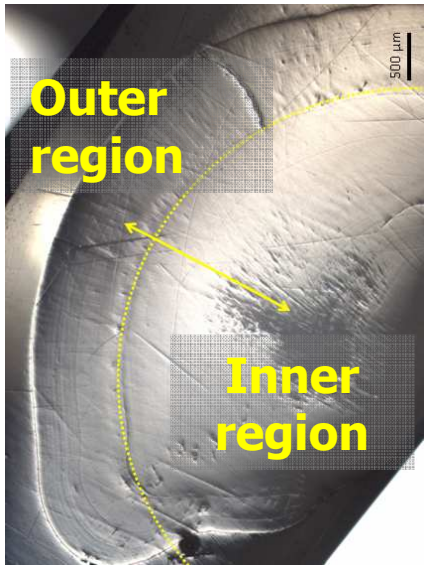
> Processing



- integrating 2D data
- fitting chitin and ACC peaks
- calculate area of peaks and ratios

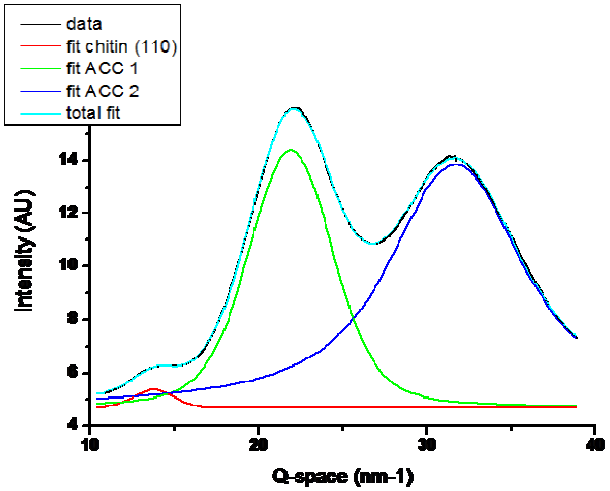
Use Case II: Crayfish Gastrolith Nanostructure

> Sample Data



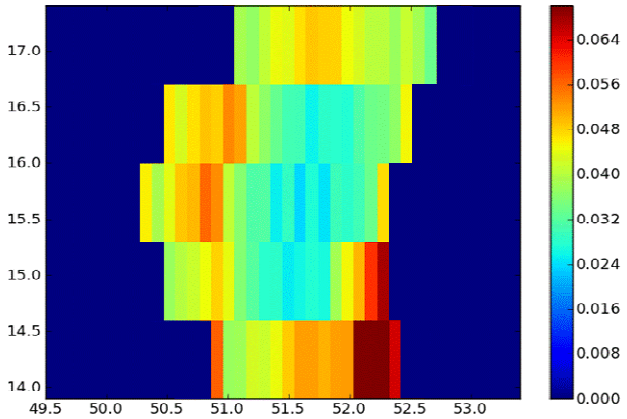
- 2D scan of 150 mccd detector files
- 10 samples scaned
- 20GB data

> Processing



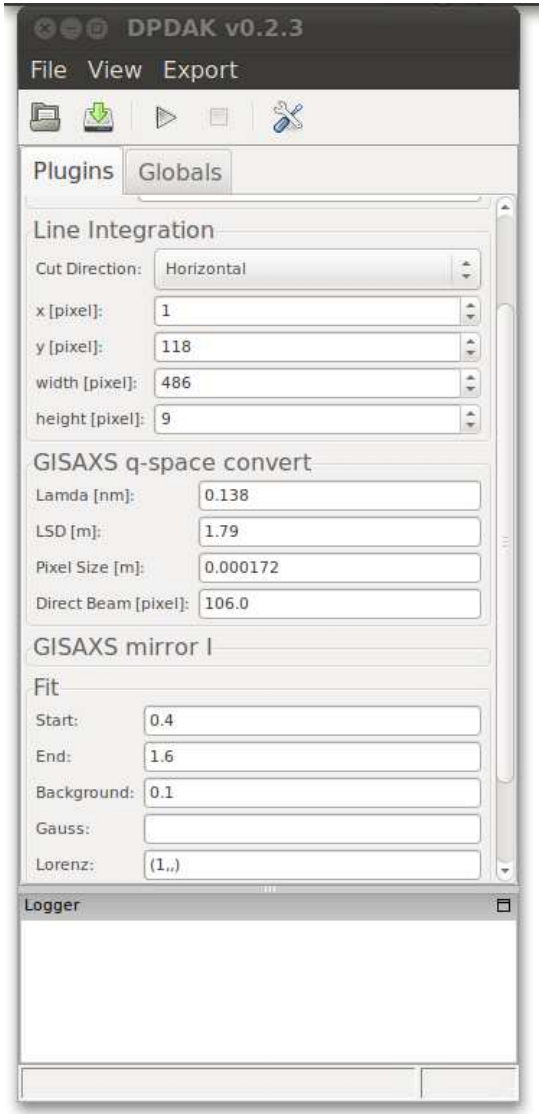
- integrating 2D data
- fitting chitin and ACC peaks
- calculate area of peaks and ratios

> Result

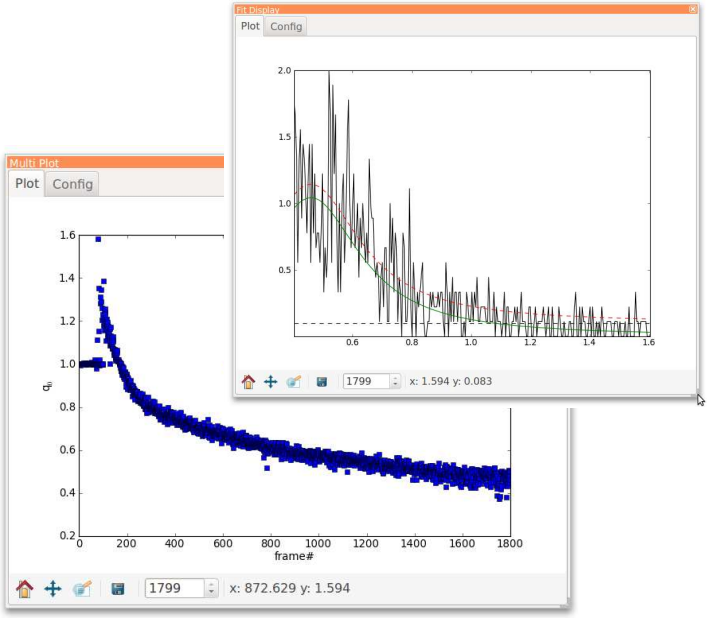


- 2D plots of ratios
- created online during the scan

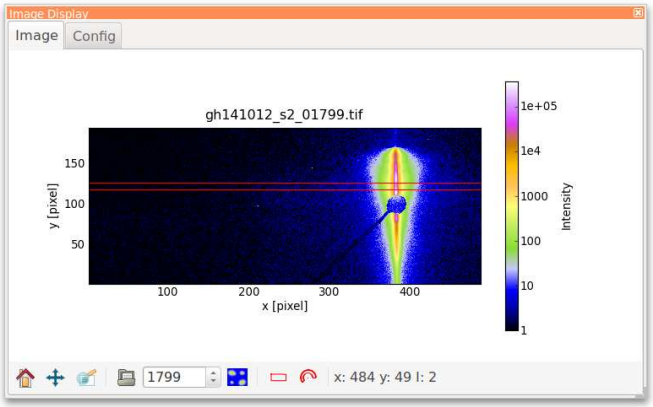
Screenshot of a analysis session

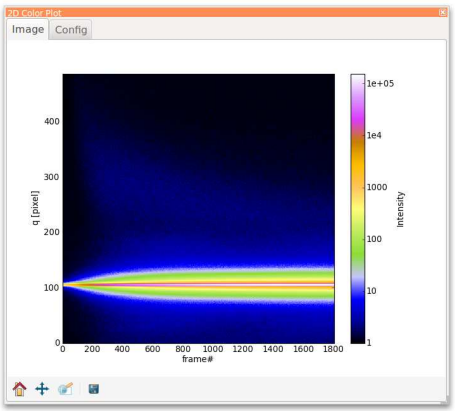


- Configuration window for plugins

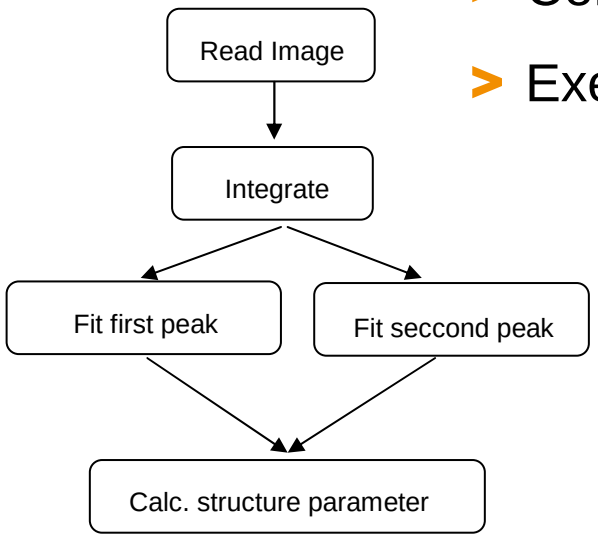


- Displays for analysis results





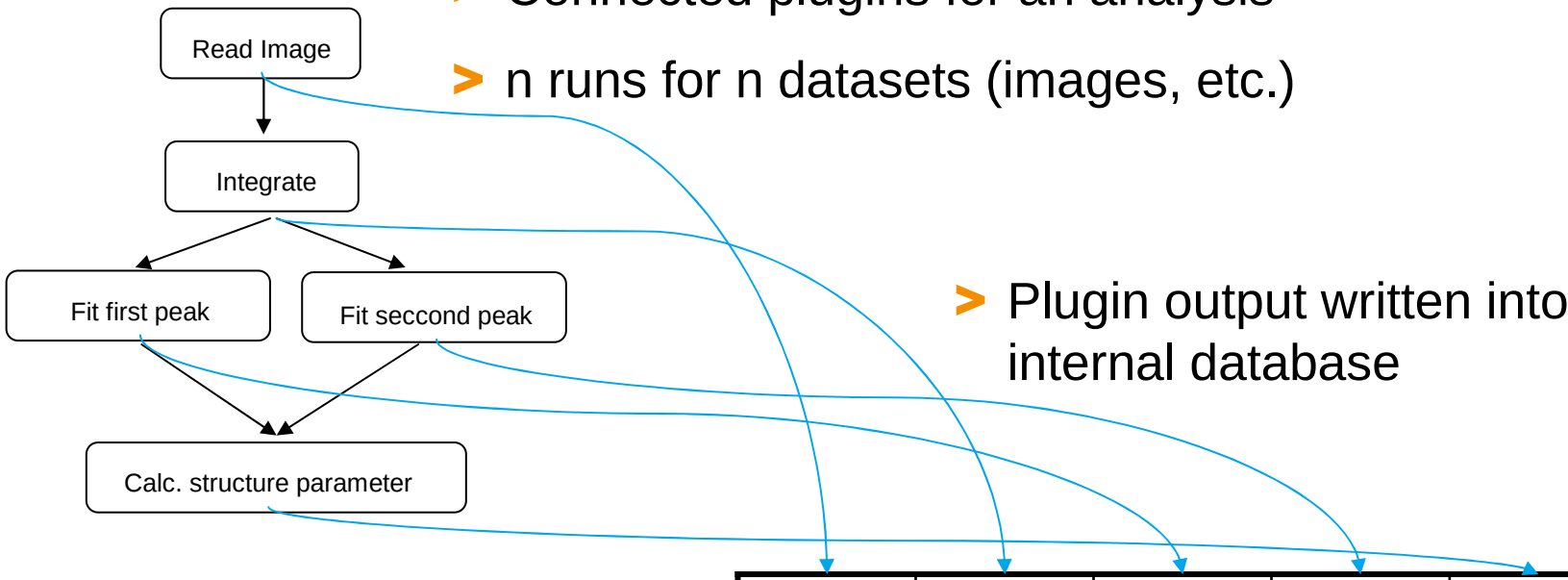
Plugin Framework



- Connected plugins for an analysis
- Execute analysis for each data point in the dataset

Plugin Framework

- Connected plugins for an analysis
- n runs for n datasets (images, etc.)

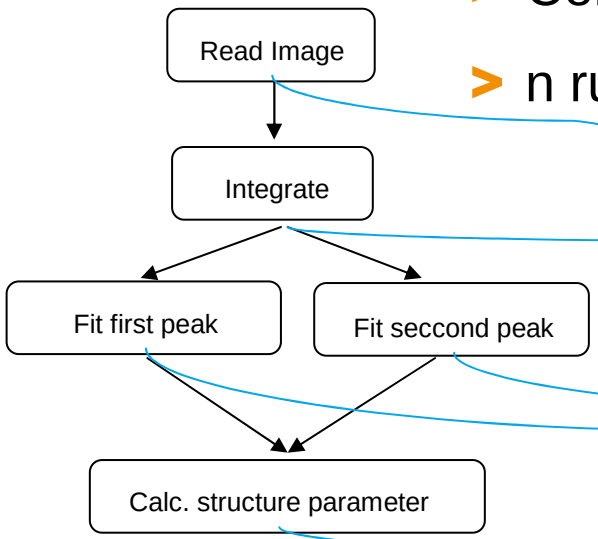


- Plugin output written into a internal database

Out1	Out2	Out3	Out4	Out5
frame1	cut1	pos1	pos1	param1
frame2	cut2	pos2	pos2	param2

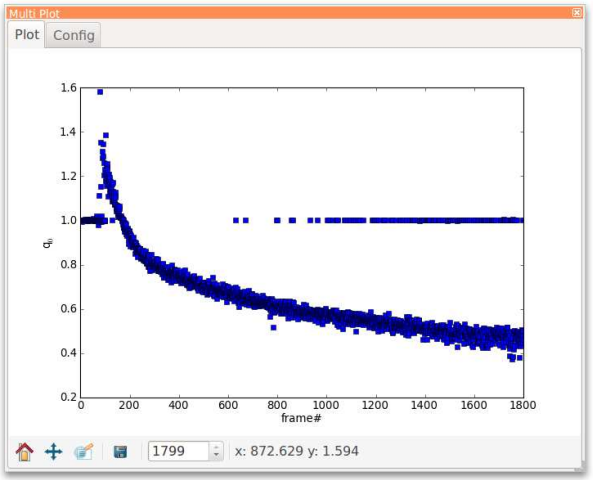
Plugin Framework

- Connected plugins for an analysis
- n runs for n datasets (images, etc.)



- Plugin output written into a internal database

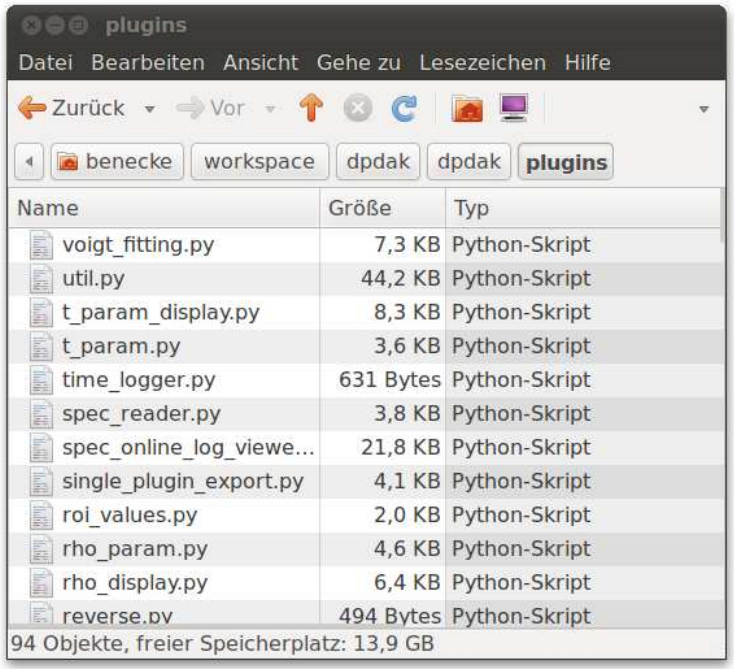
Out1	Out2	Out3	Out4	Out5
frame1	cut1	pos1	pos1	param1
frame2	cut2	pos2	pos2	param2



- Display plugins read the database and visualize the results

Plugins

- Plugins are python scripts
- Framework import plugin scripts on runtime
- Plugin interfaces for:
  - Data input & analysis (base plugins)
  - Visualisation (display plugins)
  - Export to different formats (export plugins)
- Base plugins are well defined:
  - Typed parameter
  - Typed inputs
  - Typed outputs



Plugin Example

```python
import numpy

import dpdak

class ROIValues(dpdak.BasePlugin):

    def get_info(self):
        info = dpdak.PluginInfo()
        info.set_info('ROI Values', 'Gunthard Benecke', '1.0')

        info.add_input('Image', dpdak.IMAGE)

        info.add_output('Sum', dpdak.SCALAR)
        info.add_output('Mean', dpdak.SCALAR)
        info.add_output('Min', dpdak.SCALAR)
        info.add_output('Max', dpdak.SCALAR)

        info.add_parameter('x', dpdak.INTEGER, 0)
        info.add_parameter('y', dpdak.INTEGER, 0)
        info.add_parameter('width', dpdak.INTEGER, 10)
        info.add_parameter('hight', dpdak.INTEGER, 10)

        return info

    def process(self, counter, parameter, input):
        x, y = parameter['x'], parameter['y']
        width, height = parameter['width'], parameter['height']

        image = input['Image'][y:y+height, x:x+width]
        data = {}
        data['Sum'] = image.sum()
        data['Mean'] = sum / (image.shape[0] * image.shape[1])
        data['Min'] = image.min()
        data['Max'] = image.max()

        return (dpdak.STATUS_OK, data)
```

Features

- Software
  - Python + numpy, scipy, wx, matplotlib, PIL, h5py
  - Windows (XP/Vista/7), Linux
  - GNU GPL licence
- Formats
  - Detector images (tif, cbf, mccd, edf, ima, txt, ...)
  - SPEC & Hasylab Online log file → **a standart data format would improve usability**
  - Spreadsheet text files
- Analysis
  - Integration (azimuthal/radial) & cuts
  - Basic fitting
  - Structure parameter (Rho/T/L)
- Visualisation of detector images, 2D maps of cuts and plots

Contact

- Web:
  - http://www.desy.de/~beneckeg/dpdak/
  - http://dpdak.desy.de (wiki, March 2012)
- SVN:
  - https://svnsrv.desy.de/viewvc/dpdak/
  - open hoster (sourceforge, github, google google)