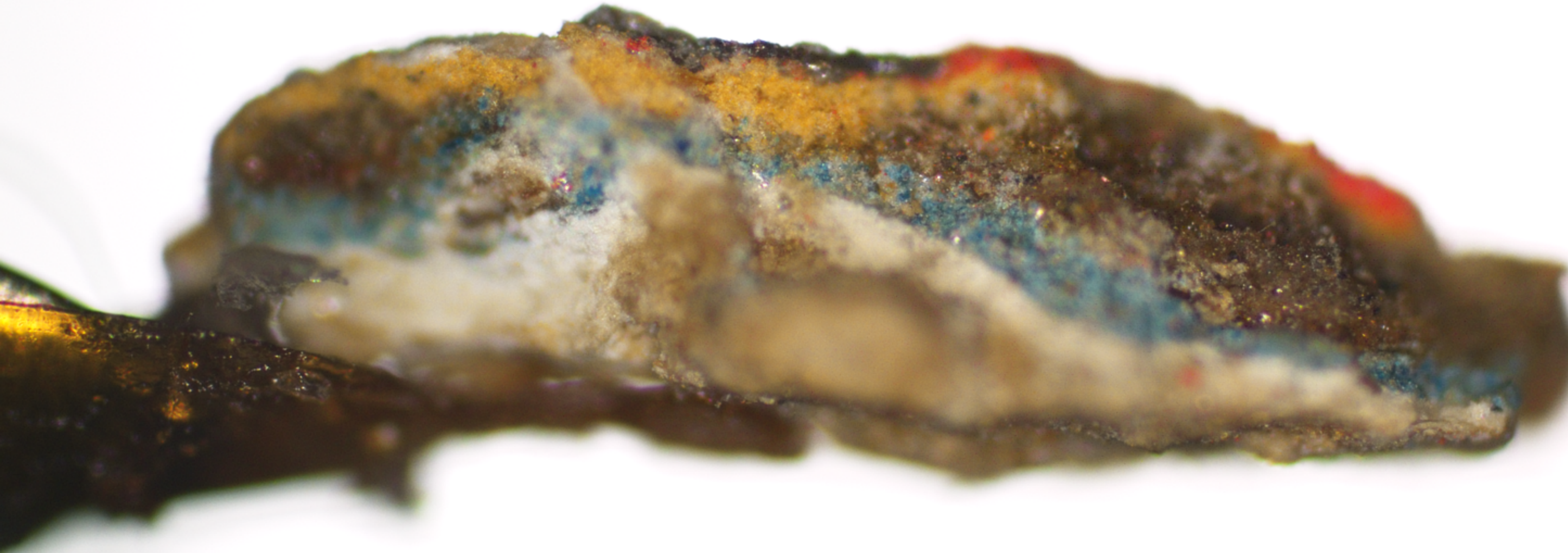
Patrimalp
Univ. Grenoble Alpes

Use of **PyFAI/Jupyter Notebook** to help processing data gathered on cultural heritage artefacts on D2AM beamline

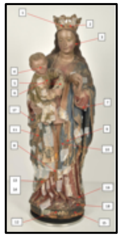
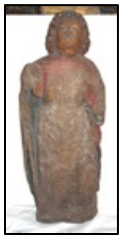


Florian Kergourlay, Pauline Martinetto, Nils Blanc, Nathalie Boudet, Stephan Arnaud, Catherine Dejoie, Pierre Bordet, Jean-Louis Hodeau, Claire Chanteraud

Experimental corpus

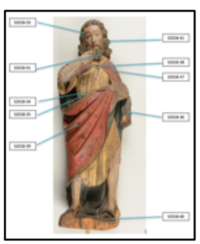
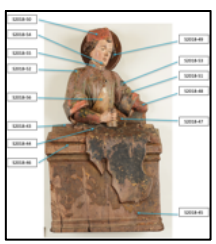
Savoy Duchy, France, 1460-1530
13 locations / 18 statues / more than 100 samples

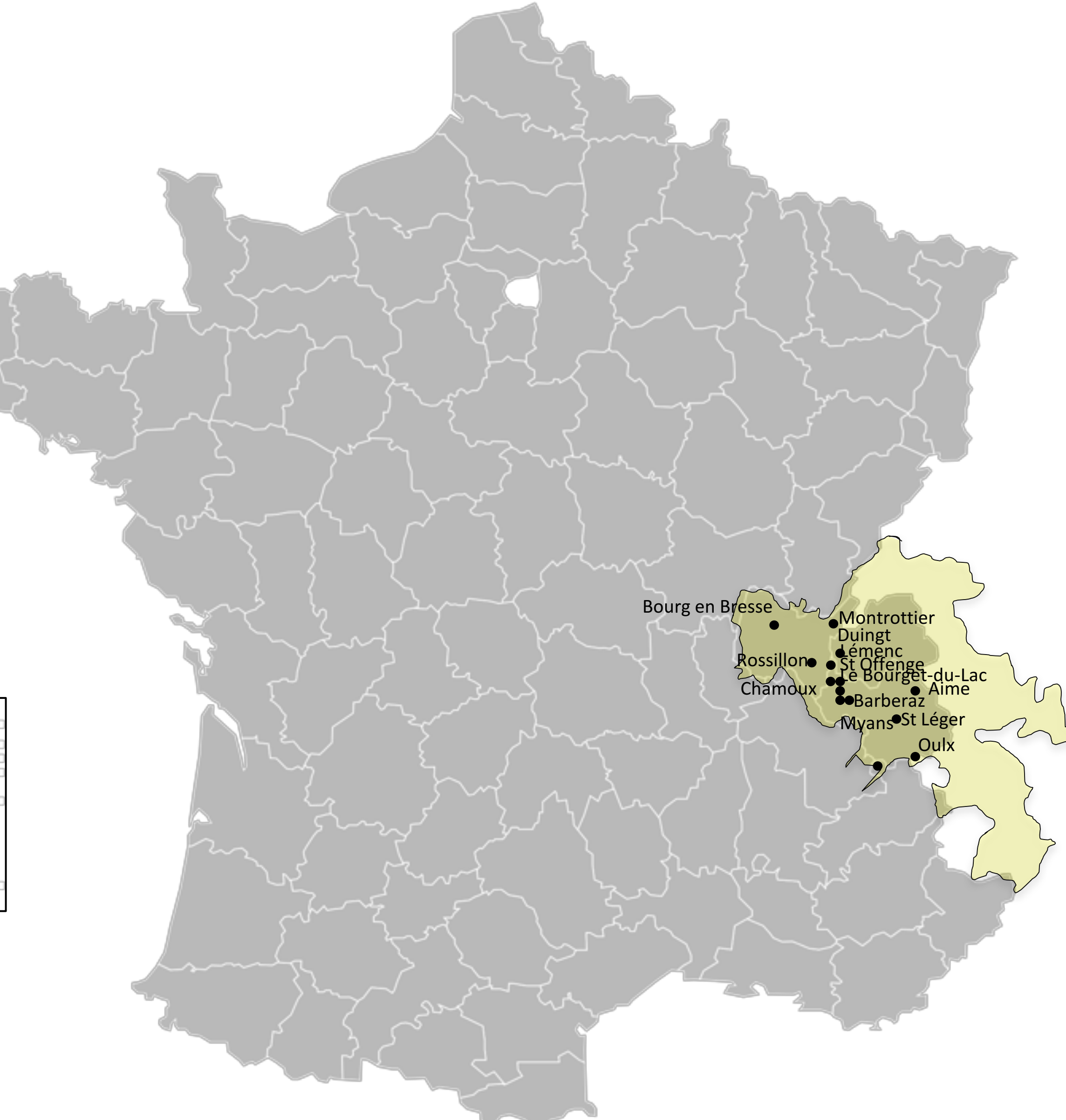


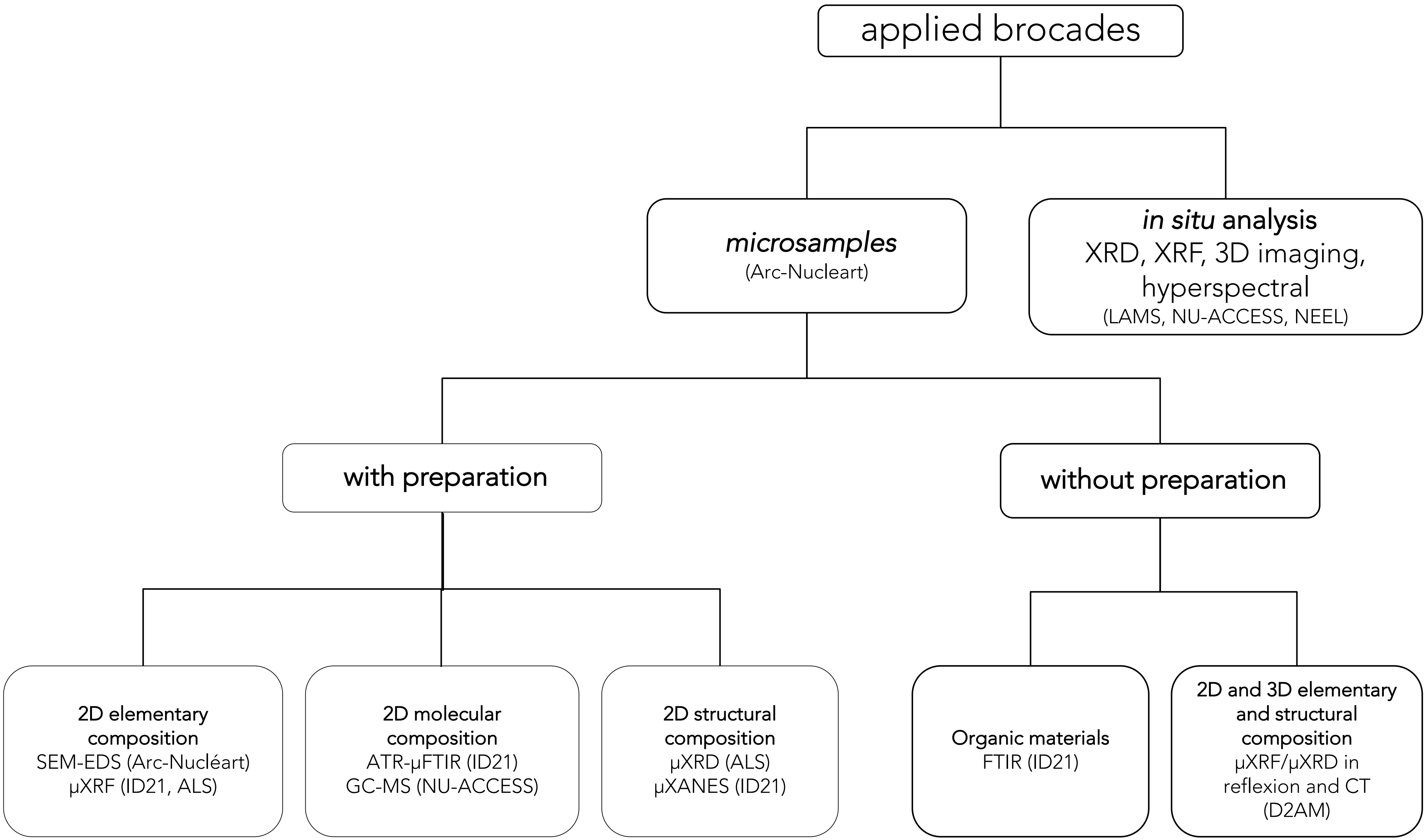




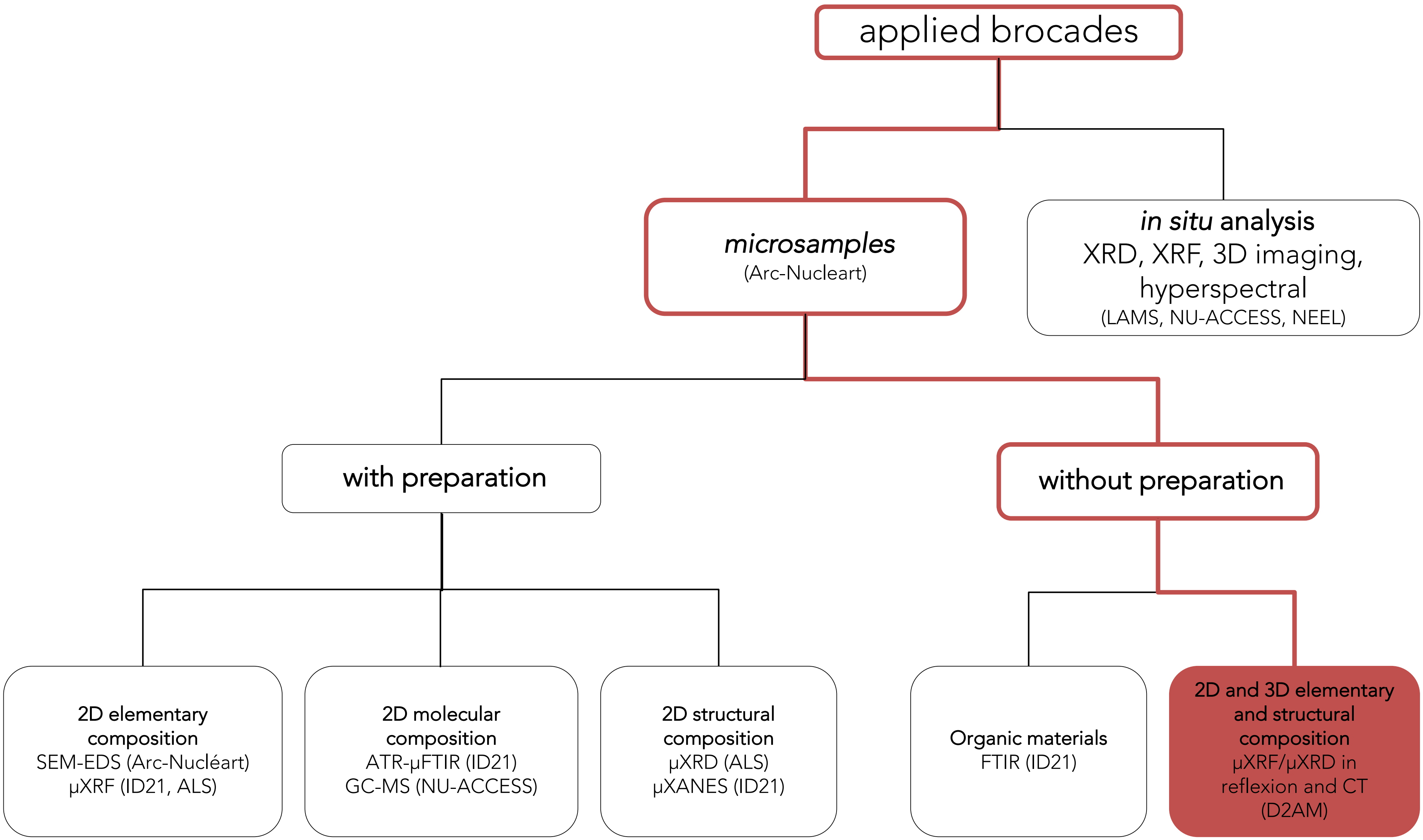




Experimental methodology



Experimental methodology



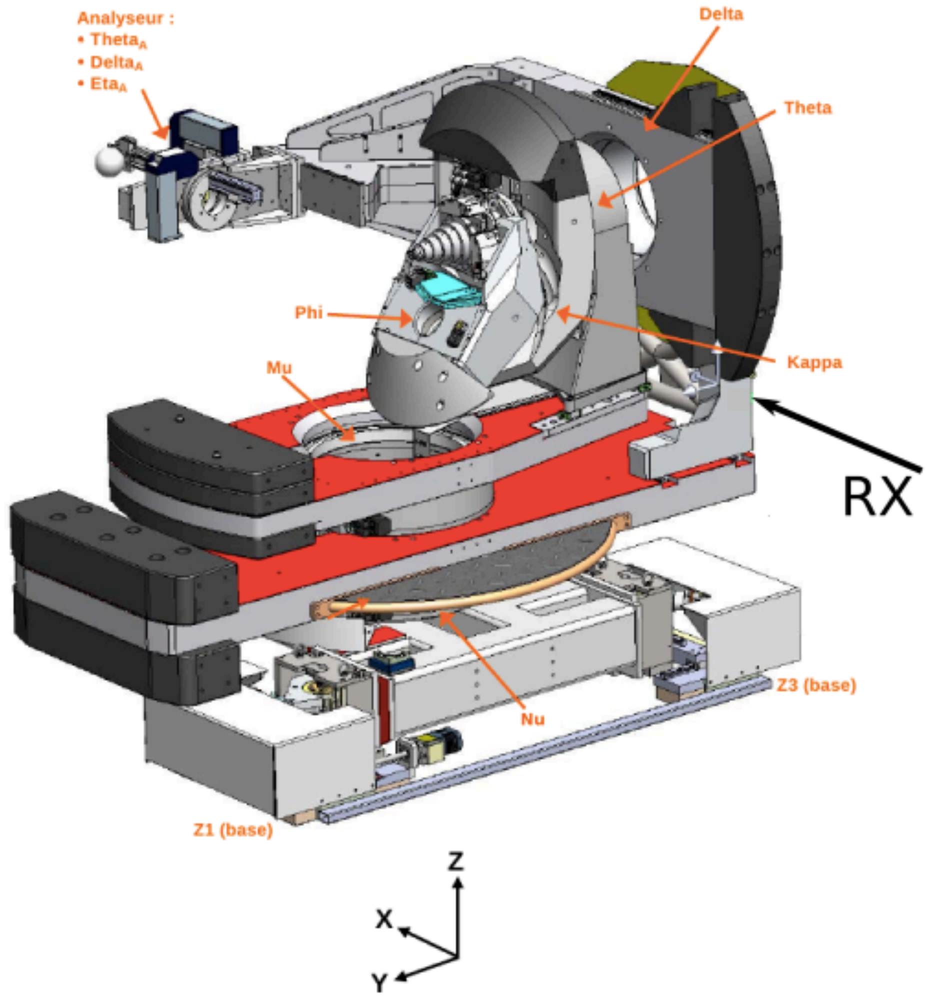
Definitions

Goniometer an instrument that allows an object to be rotated to a precise angular position

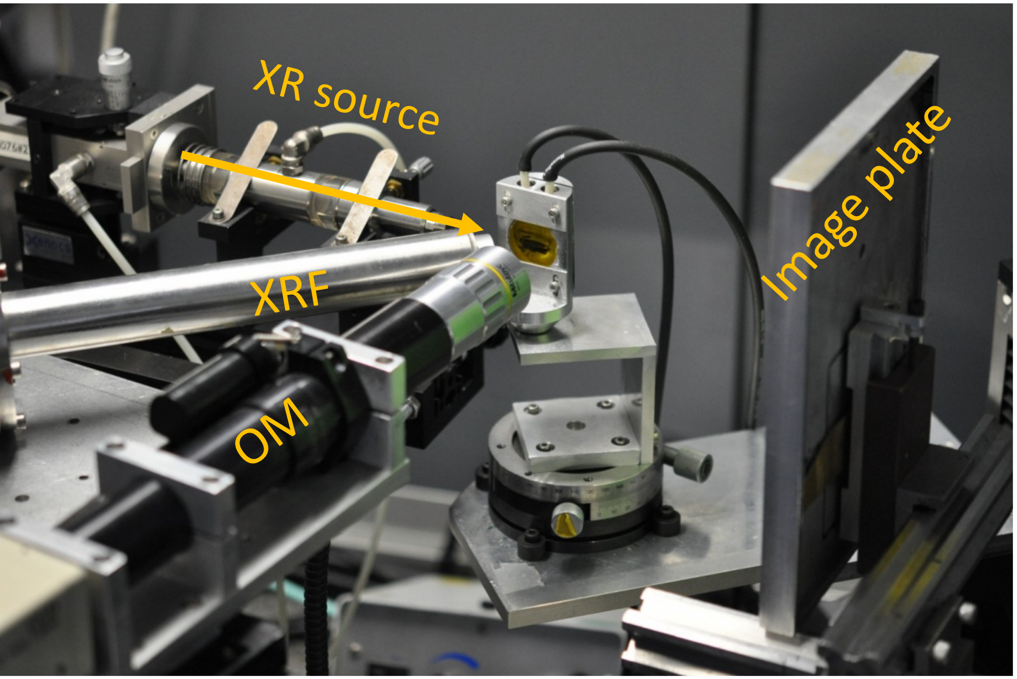
Fit2D a multi-purpose data reduction, analysis and visualization program

PyFAI Fast Azimuthal Integration using python

Jupyter notebook open-source web application that allows to create and share documents containing live code, visualizations and narrative text



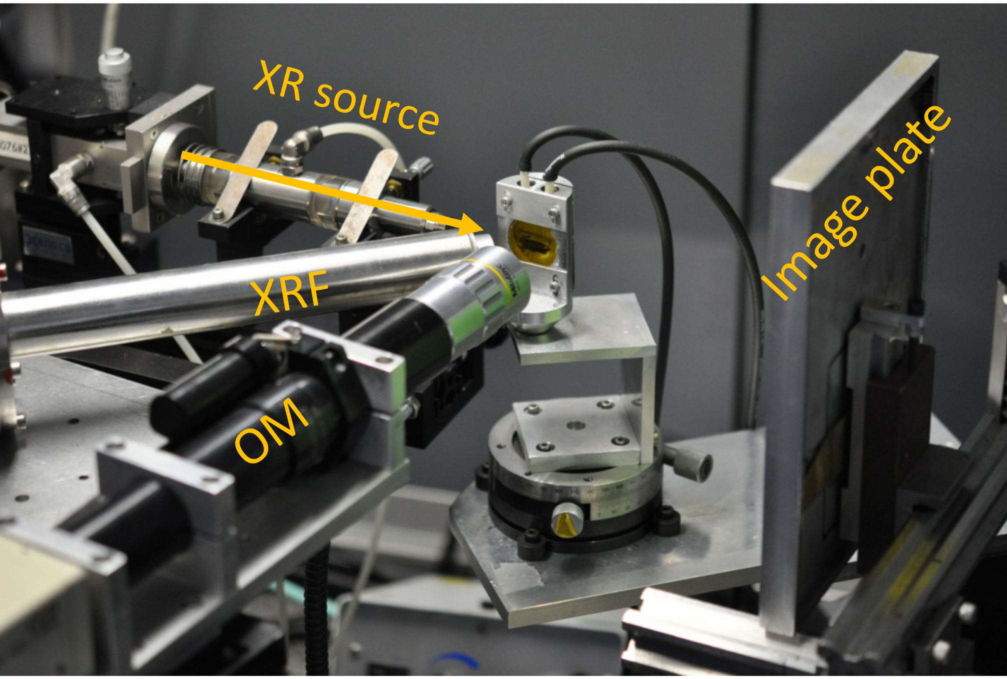
µXRF/µXRD, rotating anode in laboratory



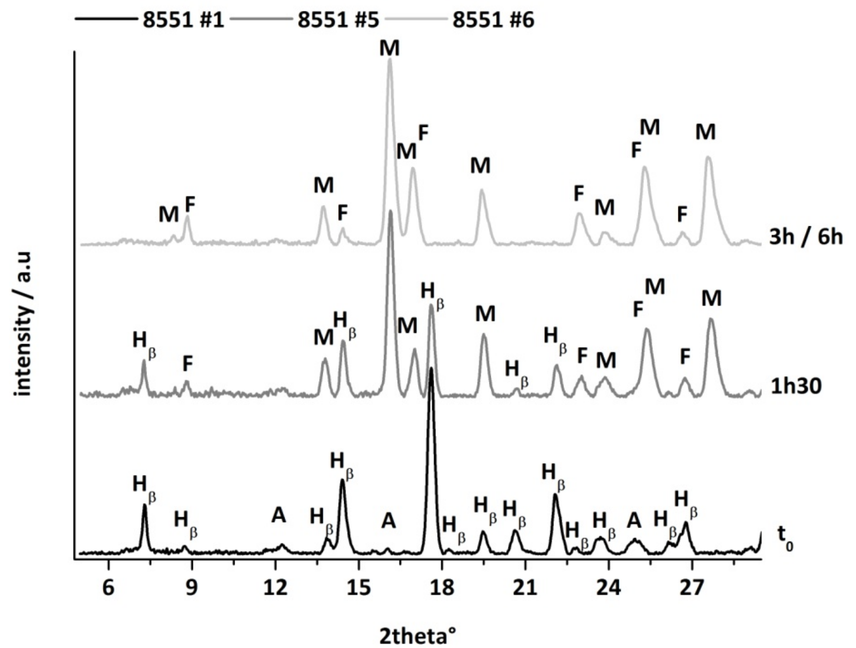
in reflexion, motionless detector and sample (one geometry)

17 keV
30x600 μm^2 beamsize
flux of about 10^6 photons/s
90min/point

µXRF/µXRD, rotating anode in laboratory



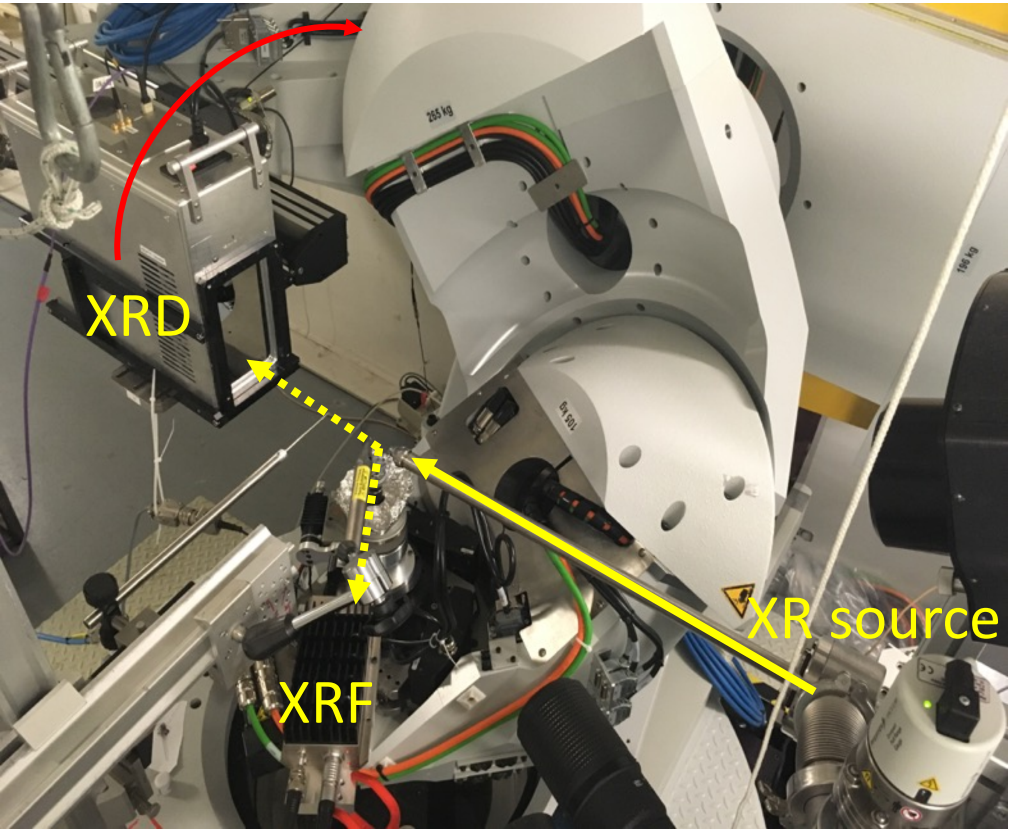
azimuthal integration with Fit2D



in reflexion, motionless detector and sample (one geometry)

17 keV
30x600 μm^2 beamsize
flux of about 10^6 photons/s
90min/point -> 1 XRD diagram

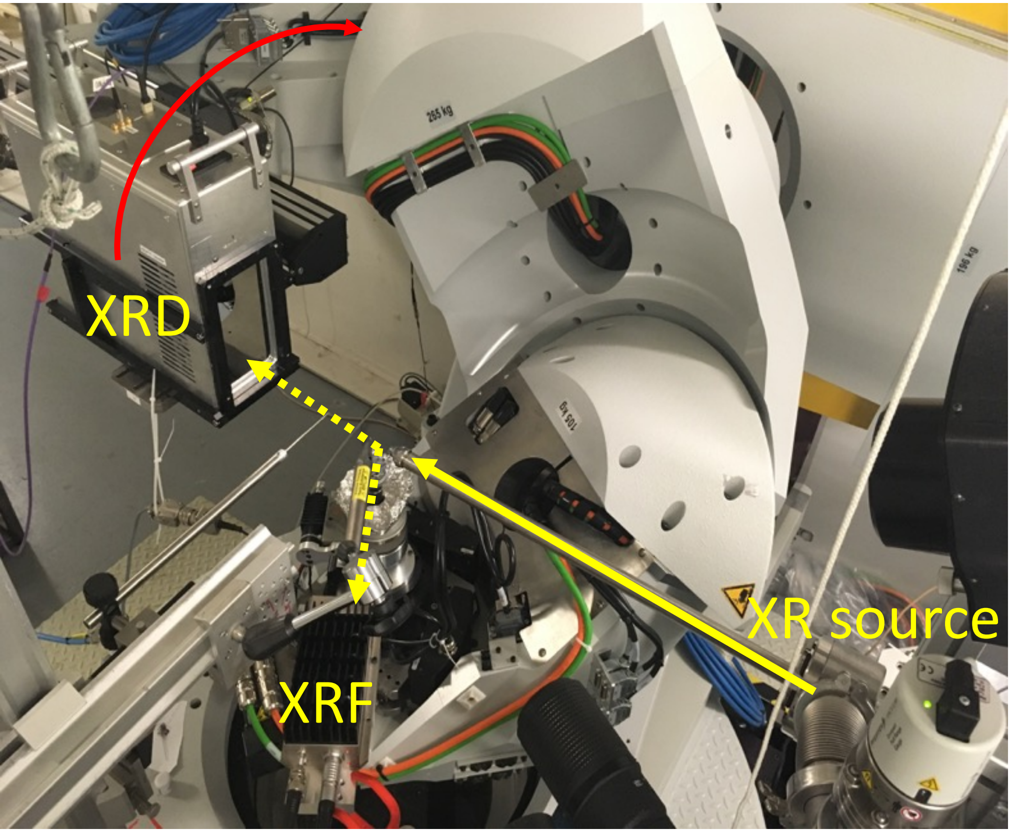
μXRF/μXRD, synchrotron source on D2AM



in reflexion, moving detector (multigeometry goniometer)

20 keV
30x40 μm^2 beam with KB mirrors
flux of about 10^8 photons/s
2s/point

µXRF/µXRD, synchrotron source on D2AM

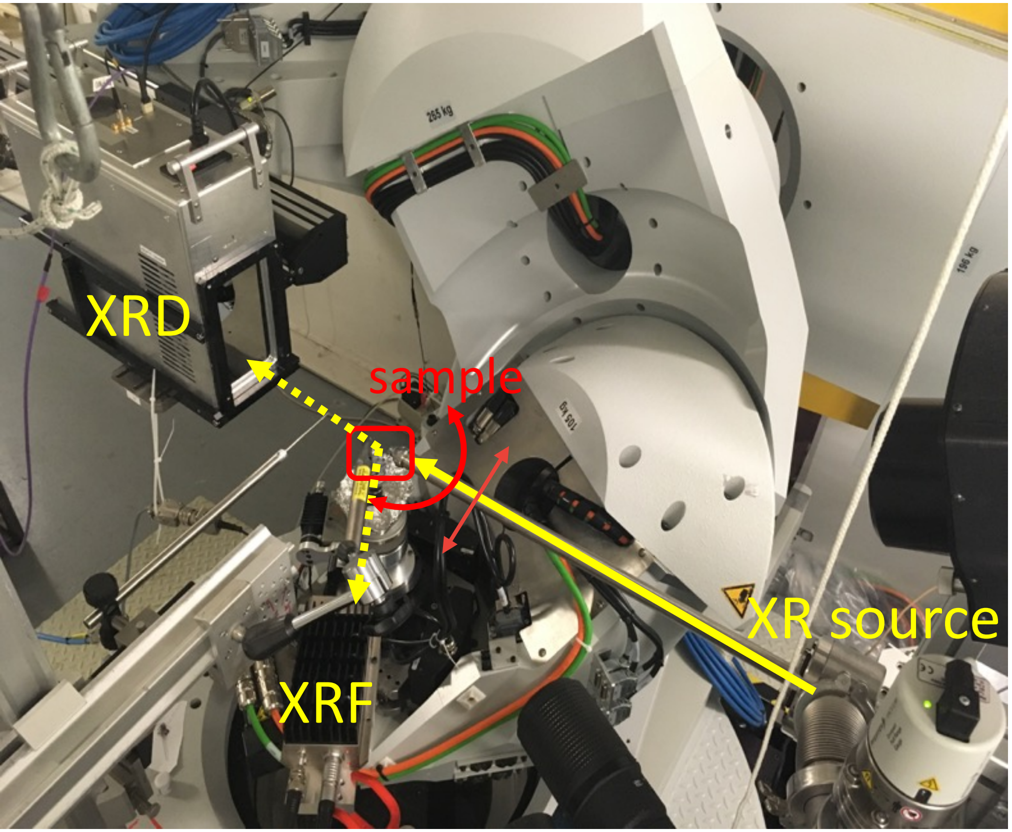


~~Fit2D~~

in reflexion, moving detector (multigeometry goniometer)

20 keV
30x40 μm^2 beam with KB mirrors
flux of about 10^8 photons/s
2s/point -> 20 XRD diagrams/different geometries

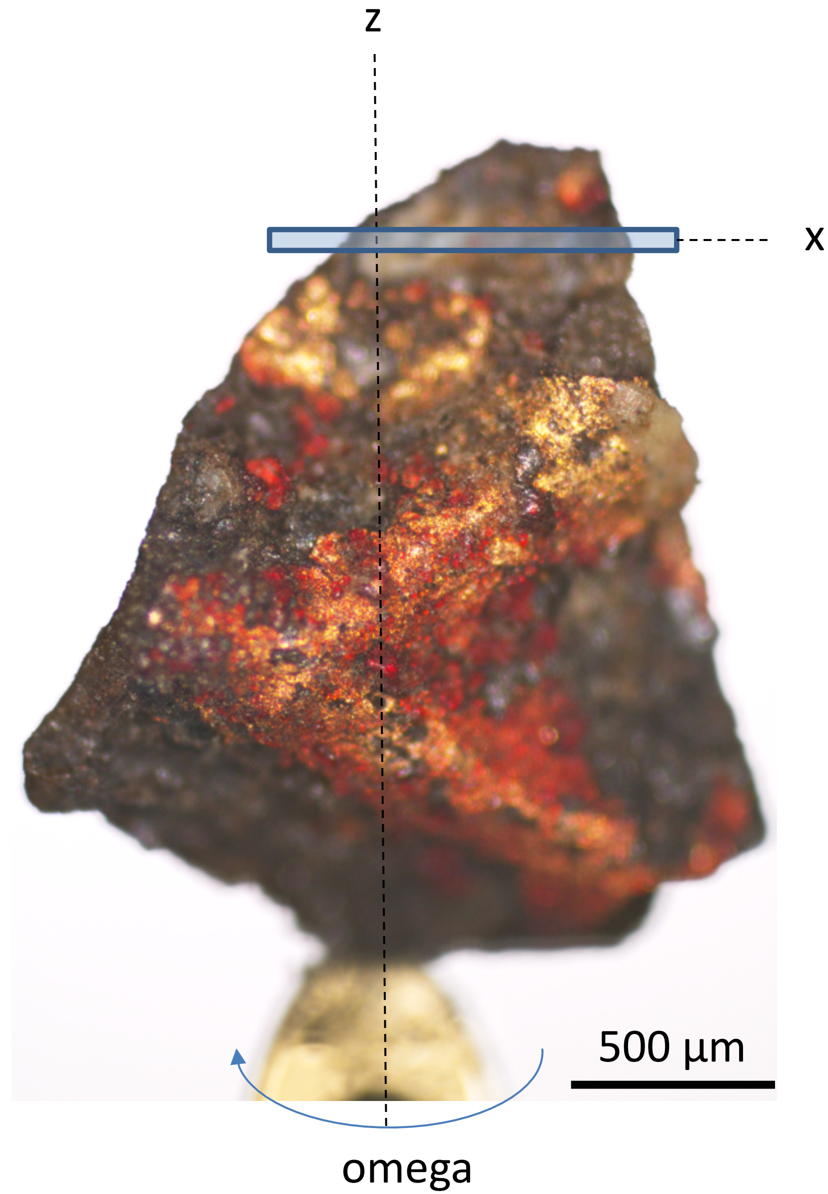
µXRF/µXRD-CT, synchrotron source on D2AM



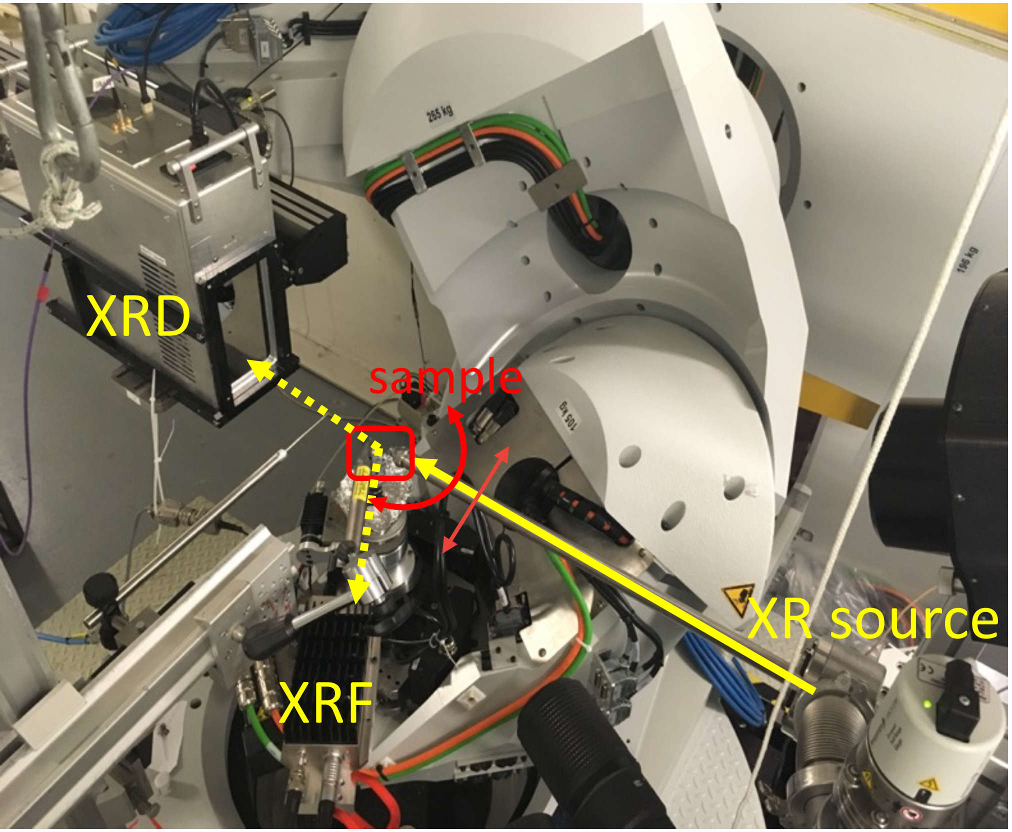
in transmission, moveable sample in rotation and translation

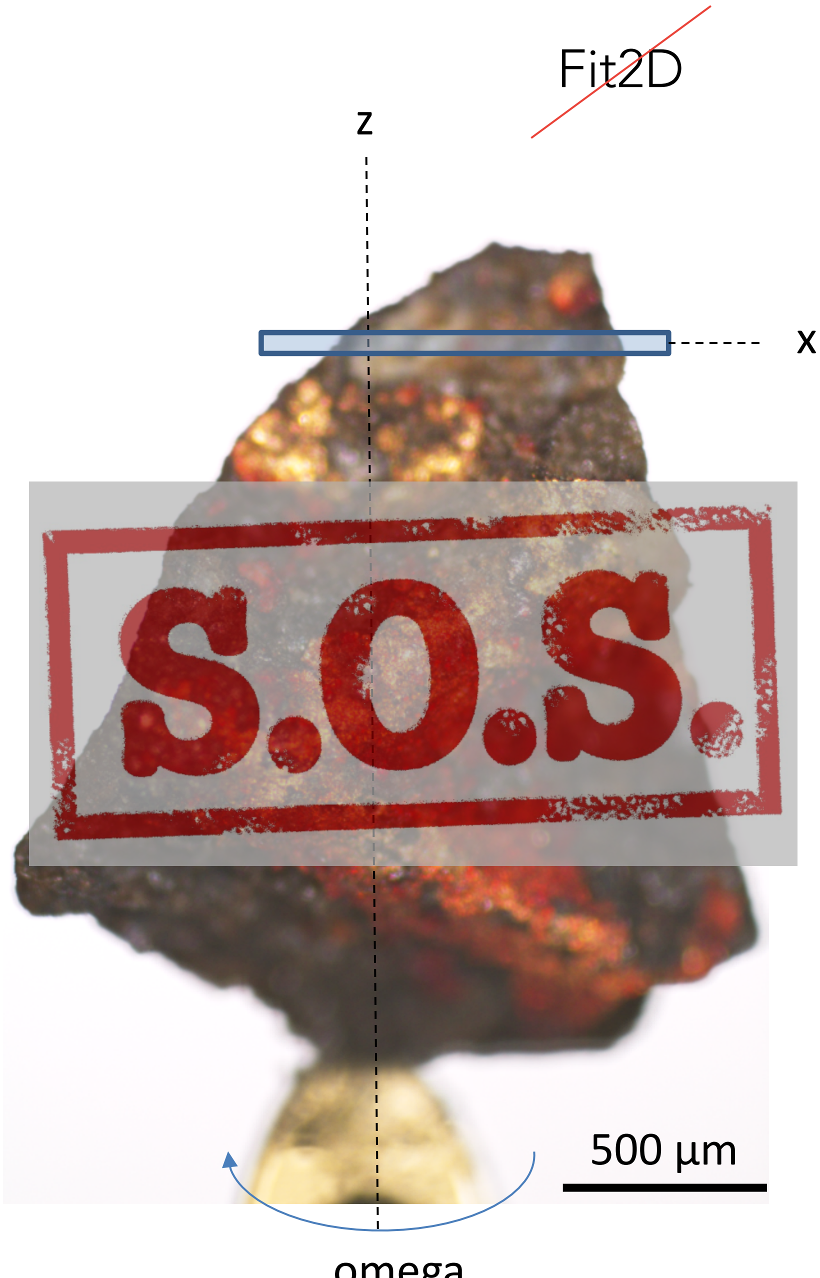
20 keV
30x40 µm^2 beam with KB mirrors
flux of about 10^8 photons/s
1s/point

sample mounted on a "racket" on a goniometer
continuous acquisition on 360° every 2° (180 rotations)
for a defined range of x (40-80 translations) and z
3 min / 360° -> 2-4h / layer



µXRF/µXRD-CT, synchrotron source on D2AM





in transmission, moveable sample in rotation and translation

20 keV
30x40 μm^2 beam with KB mirrors
flux of about 10^8 photons/s
1s/point -> 10000 XRD diagrams

sample mounted on a "racket" on a goniometer
continuous acquisition on 360° every 2° (180 rotations)
for a defined range of x (40-80 translations) and z
3 min / 360° -> 2-4h / layer

Use of PyFAI+Jupyter Notebook

Use of PyFAI+Jupyter Notebook
1st case XRD in reflexion mode with moveable detector

import dedicated libraries/modules

definition of the direct beam (poni: points of normal incidence)

load images and calibrants for fitting poni

definition of the goniometer parameters

definition of the geometry refinement

geometry refinement function

definition of the multigeometry integrator

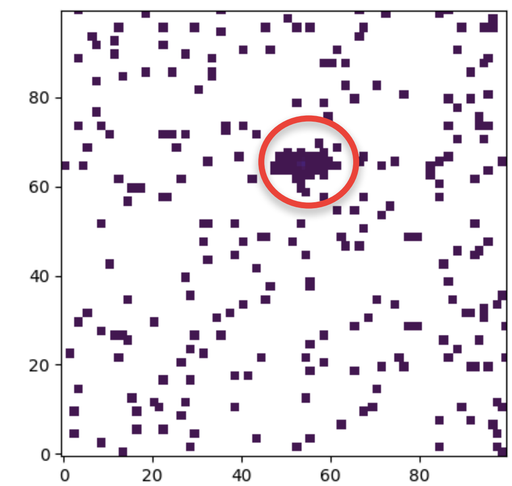
In [2]:
```
#Loading of a few libraries
import ipywidgets as widgets
import os,time
import glob
import random
import fabio
import pyFAI
import numexpr
import sys
#sys.path.append("/home/nblanc/SCRIPTS/PYFAI")
#import D5SizeAdjust
sys.path.append("/data/bm02/SCRIPTS")
import Flat
from pyFAI.goniometer import GeometryTransformation, GoniometerRefinement, Goniometer
from pyFAI.gui import jupyter
start_time = time.time()
from ipywidgets import interact, interactive, fixed, interact_manual
```

Use of PyFAI+Jupyter Notebook
1st case XRD in reflexion mode with moveable detector

- import dedicated libraries/modules
- **definition of the direct beam (poni: points of normal incidence)**
- load images and calibrants for fitting poni
- definition of the goniometer parameters
- definition of the geometry refinement
- geometry refinement function
- definition of the multigeometry integrator

In [4]:

```
import pyFAI
import fabio
from pyFAI.distortion import Distortion
import sys

D5_img=fabio.open("/data/bm02/nblanc2/IH-HG-6/raw/18Aug27D5_0153.edf.gz")
D5=pyFAI.detector_factory("/data/bm02/SCRIPTS/PYFAI/geomD5_V1/D5_V1Geom-nils.h5")

print(D5)

D5_dis  = Distortion(D5, resize=True)
D5_raw = D5_img.data
D5_cor = D5_dis.correct(D5_raw)

figure()
imshow(numpy.log(D5_cor[900:1000,250:350]),interpolation="nearest",origin="lower",vmin=3,vmax=11)

from scipy import ndimage
cen=ndimage.measurements.center_of_mass(D5_cor[900:1000,250:350])
#print(cen[0]+400,cen[1])

print('PONI1 = ',(cen[0]+900)*.130e-3,'PONI2 = ',(cen[1]+250)*.130e-3)
```




```
PONI1 =  0.12465946444761818 PONI2 =  0.03942788662147352
```

Use of PyFAI+Jupyter Notebook

1st case XRD in reflexion mode with moveable detector

- import dedicated libraries/modules
- definition of the direct beam (poni: points of normal incidence)
- **load images and calibrants for fitting poni**
- definition of the goniometer parameters
- definition of the geometry refinement
- geometry refinement function
- definition of the multigeometry integrator

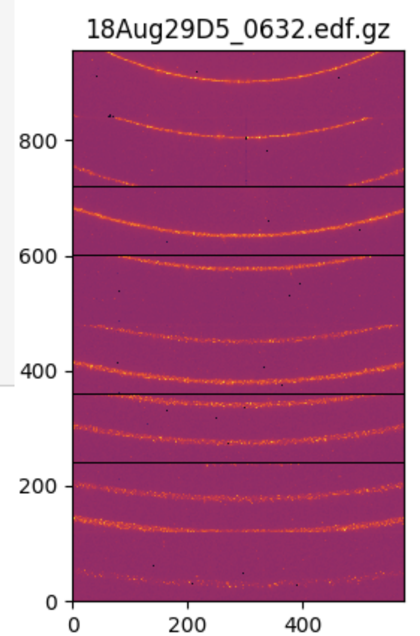
In [10]:

```
#loading of the list of files, and display of the last one with its headers

image_files = glob.glob("*.edf.gz")
image_files.sort()

print("List of images: " + ", ".join(image_files) + "." + os.linesep)
print(image_files)
fimg = fabio.open(image_files[0])

print("Image headers:")
for key, value in  fimg.header.items():
    print("%s: %s"%(key,value))

jupyter.display(fimg.data, label=fimg.filename)
```

In [8]:

```
wavelength = pyFAI.units.hc/20. *1e-10
print(wavelength)
from pyFAI.calibrant import ALL_CALIBRANTS
# c = Cell.cubic(4.1568260)
# c.save("LaB6", dmin=0.2)
LaB6 = ALL_CALIBRANTS("LaB6")
LaB6.wavelength = wavelength
print("2theta max: ", numpy.degrees(LaB6.get_2th()[-1]))
print("Number of reflections: ", len(LaB6.get_2th()))

#Use a few manually calibrated images:
img_files = [i for i in glob.glob("*.edf.gz") if "new" not in i]
npt_files = [i for i in glob.glob("*.npt") if "new" not in i]
npt_files.sort()
img_files.sort()
npt_files[0]
print("Number of hand-calibrated images :",len(npt_files))
```

```
6.19920986982036e-11
2theta max:  172.11488486407407
Number of reflections:  151
Number of hand-calibrated images : 5
```



Use of PyFAI+Jupyter Notebook

1st case XRD in reflexion mode with moveable detector

import dedicated libraries/modules

definition of the direct beam (poni: points of normal incidence)

load images and calibrants for fitting poni

definition of the goniometer parameters

definition of the geometry refinement

geometry refinement function

definition of the multigeometry integrator

In [11]:

```
#Definition of the goniometer translation function:
# The detector rotates vertically, around the horizontal axis, i.e. rot2

goniotrans = GeometryTransformation(param_names = ["dist", "poni1", "poni2",
                                                   "rot1", "rot2", "rot3", "scale1", "scale2" ],
                                    dist_expr="dist",
                                    poni1_expr="poni1",
                                    poni2_expr="poni2",
                                    rot1_expr="pi*(scale1 * pos + 180.* rot1/pi)/180.",
                                    rot2_expr="pi*(scale2 * pos + 180.* rot2/pi)/180.",
                                    rot3_expr="rot3")


#Definition of the function reading the goniometer angle from the filename of the image.

def get_angle(metadata):
    """Takes the angle from the first motor position and returns the angle of the goniometer arm"""
    return float(metadata["motor_pos"].split()[0])


print('filename', fimg.filename, "angle:",get_angle(fimg.header))
```

```
filename 18Aug29D5_0632.edf.gz angle: 14.9998
```

Use of PyFAI+Jupyter Notebook
1st case XRD in reflexion mode with moveable detector

- import dedicated libraries/modules
- definition of the direct beam (poni: points of normal incidence)
- load images and calibrants for fitting poni
- definition of the goniometer parameters
- **definition of the geometry refinement**
- geometry refinement function
- definition of the multigeometry integrator

In [12]:
```
#Definition of the geometry refinement: the parameter order is the same as the param_names
d5 = pyFAI.detector_factory("/data/bm02/SCRIPTS/PYFAI/geomD5_V1/D5_V1Geom-nils.h5")

rot3 = 0
#poni1 = 0.06808247550356585
#poni2 = 0.010213828943413071
scale1 = 0
scale2 = 1.0

param = {"dist":0.5,
         "poni1":0.05,
         "poni2":0.05,
         #"poni1":poni1,
         #"poni2":poni2,
         "rot1":0,
         "rot2":0,
         "rot3": rot3,
         "scale1": scale1,
         "scale2": scale2,
        }

#Defines the bounds for some variables
bounds = {"dist": (0.2, 0.8),
#           "poni1": (poni1, poni1), ## on fixe poni1 et poni2 aux valeurs fitées
#           "poni2": (poni2, poni2),
          "poni1": (0., .2),
          "poni2": (0.,.1),
          "rot1": (-1, 1),
          "rot2": (-1, 1),
          "rot3": (rot3, rot3), #strict bounds on rot3
          #"scale1": (scale1, scale1),
          #"scale2": (scale2, scale2),
         }

gonioref = GoniometerRefinement(param, #initial guess
                                bounds=bounds,
                                pos_function=get_angle,
                                trans_function=goniotrans,
                                detector=d5, wavelength=wavelength)
```

Use of PyFAI+Jupyter Notebook
1st case XRD in reflexion mode with moveable detector

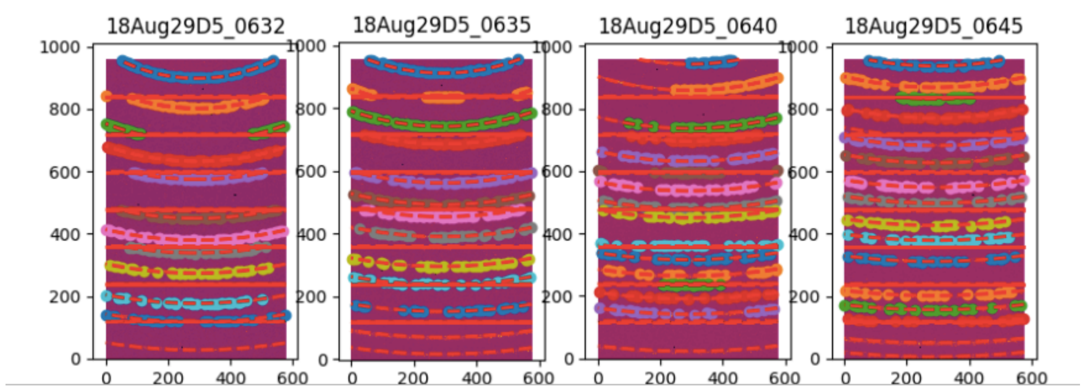
- import dedicated libraries/modules
- definition of the direct beam (poni: points of normal incidence)
- load images and calibrants for fitting poni
- definition of the goniometer parameters
- definition of the geometry refinement
- **geometry refinement function**
- definition of the multigeometry integrator

In [13]:
```
gonioref.refine2()
```

```
Cost function before refinement: 0.0353716739054859
[0.5  0.05 0.05 0.   0.   0.   0.   1.  ]
      fun: 8.796619561108617e-07
      jac: array([-6.35488838e-07,  2.82667295e-06,  2.89236802e-07,  9.22722975e-07,
        9.33636088e-07,  1.91846539e-13, -2.11789725e-07,  3.86270372e-07])
  message: 'Optimization terminated successfully.'
     nfev: 354
      nit: 35
     njev: 35
   status: 0
  success: True
        x: array([ 0.33708487,  0.128258  ,  0.04097444,  0.00635238, -0.0120192 ,
        0.        ,  0.0062817 ,  1.00752562])
Cost function after refinement: 8.796619561108617e-07
GonioParam(dist=0.33708487043080915, poni1=0.1282579957641337, poni2=0.04097443659444216,
rot2=-0.012019201941454392, rot3=0.0, scale1=0.006281695298798726, scale2=1.00752562225594
maxdelta on: dist (0) 0.5 --> 0.33708487043080915
```

Out[13]:
```
array([ 0.33708487,  0.128258  ,  0.04097444,  0.00635238, -0.0120192 ,
        0.        ,  0.0062817 ,  1.00752562])
```

In [20]:
```
width=4
height=int(ceil(len(gonioref.single_geometries)/width))
fig,ax = subplots(height, width,figsize=(10,15))
for idx, sg in enumerate(gonioref.single_geometries.values()):
    sg.geometry_refinement.set_param(gonioref.get_ai(sg.get_position()).param)
    jupyter.display(sg=sg, ax=ax[idx//width, idx%width])
```



Use of PyFAI+Jupyter Notebook
1st case XRD in reflexion mode with moveable detector

import dedicated libraries/modules

definition of the direct beam (poni: points of normal incidence)

load images and calibrants for fitting poni

definition of the goniometer parameters

definition of the geometry refinement

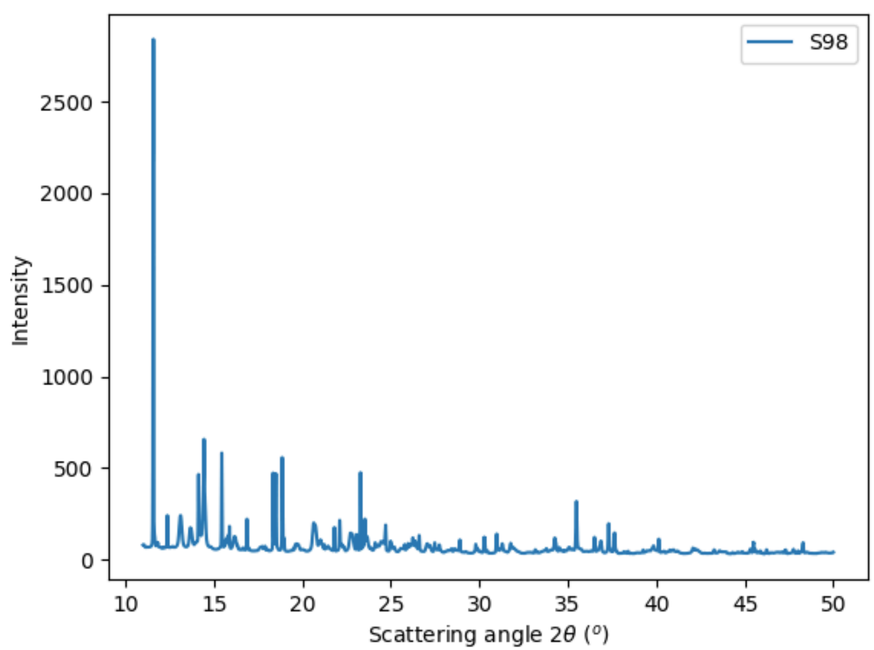
geometry refinement function

definition of the multigeometry integrator

In [25]:
```
#Create a MultiGeometry integrator from the refined geometry:
angles = []
images = []
monitor = []
for sg in gonioref.single_geometries.values():
    angles.append(sg.get_position())
    images.append(sg.image)
    monitor.append(sg.metadata)

multigeo = gonioref.get_mg(angles)
multigeo.radial_range=(0, 55)
print(multigeo)
```

```
MultiGeometry integrator with 6 geometries on (0, 55) radial range (2th_deg)
```

In [26]:
```
gonioref.save("kappa-d2am_D5vert-IH-HG-6.json")
```

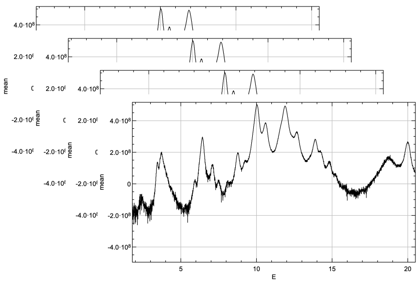


Use of PyFAI+Jupyter Notebook

2nd case XRD in transmission and tomography mode

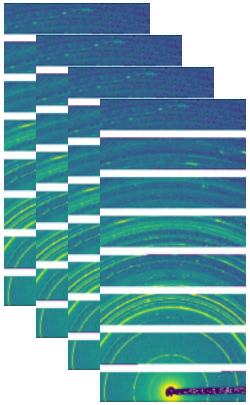
µXRF/µXRD-CT processing workflow

XRF



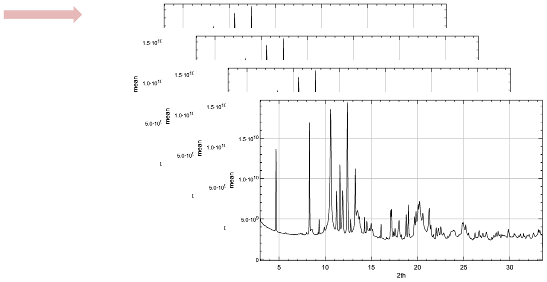
N_x x N_w spectra

XRD



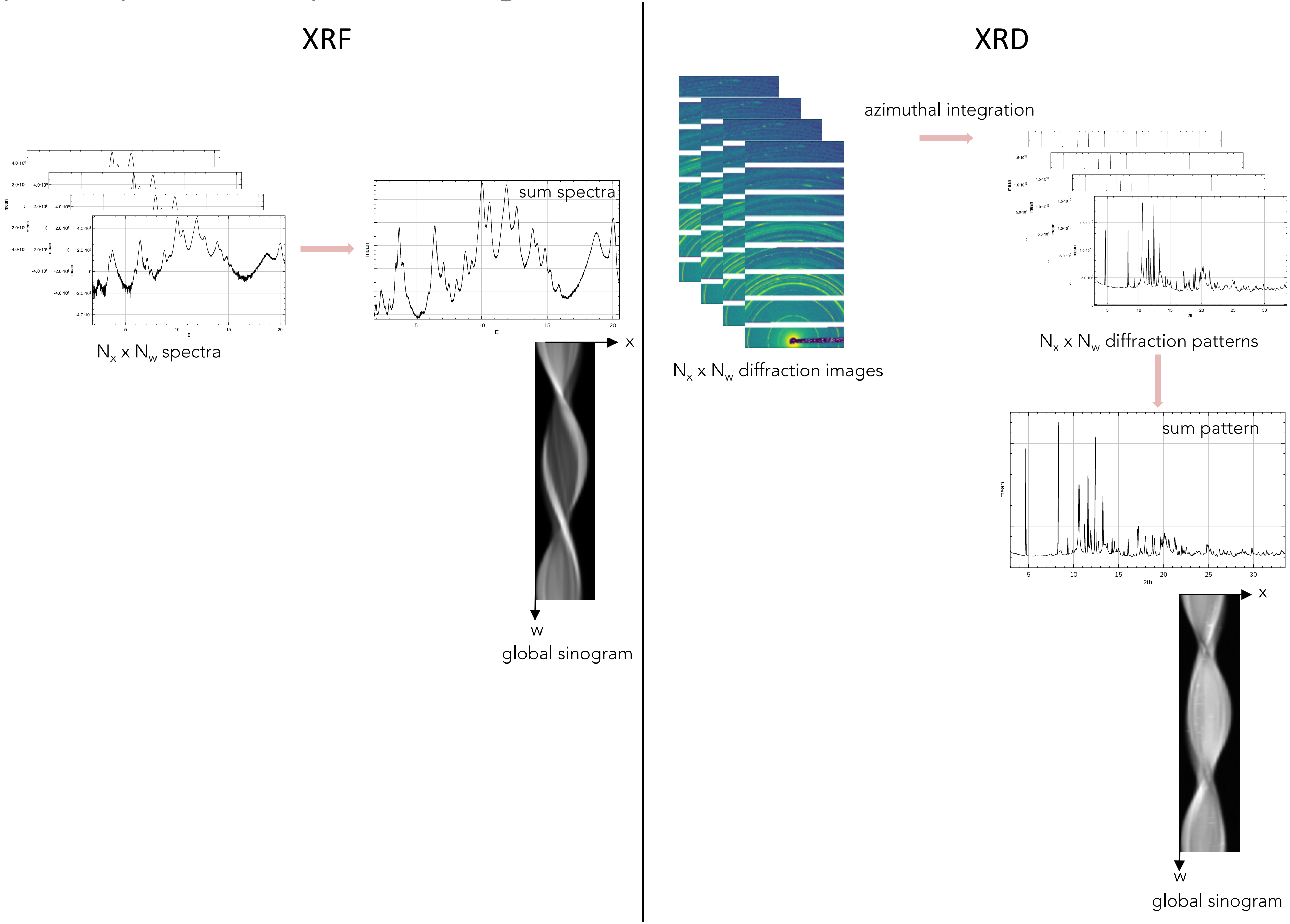
N_x x N_w diffraction images

azimuthal integration

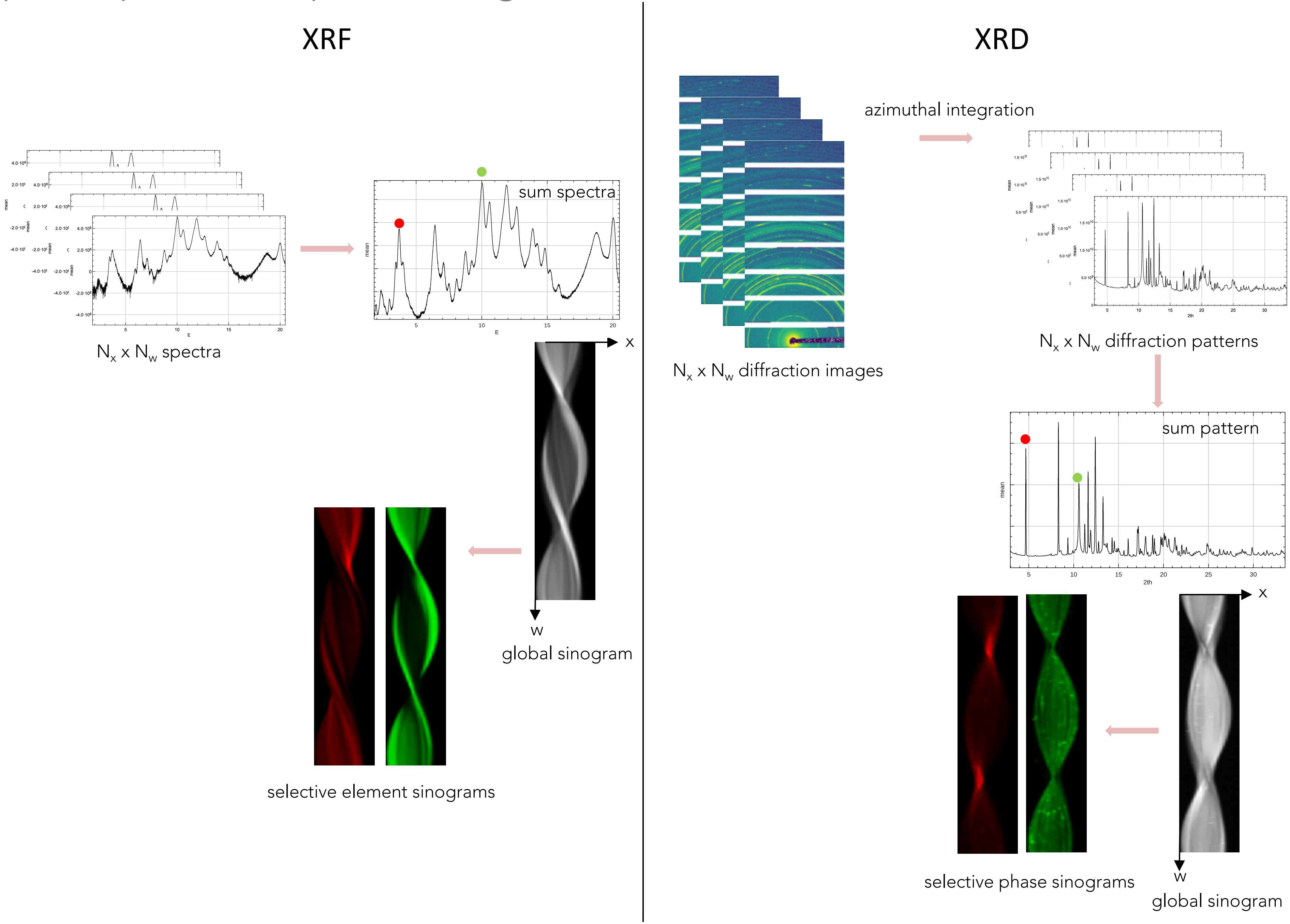


N_x x N_w diffraction patterns

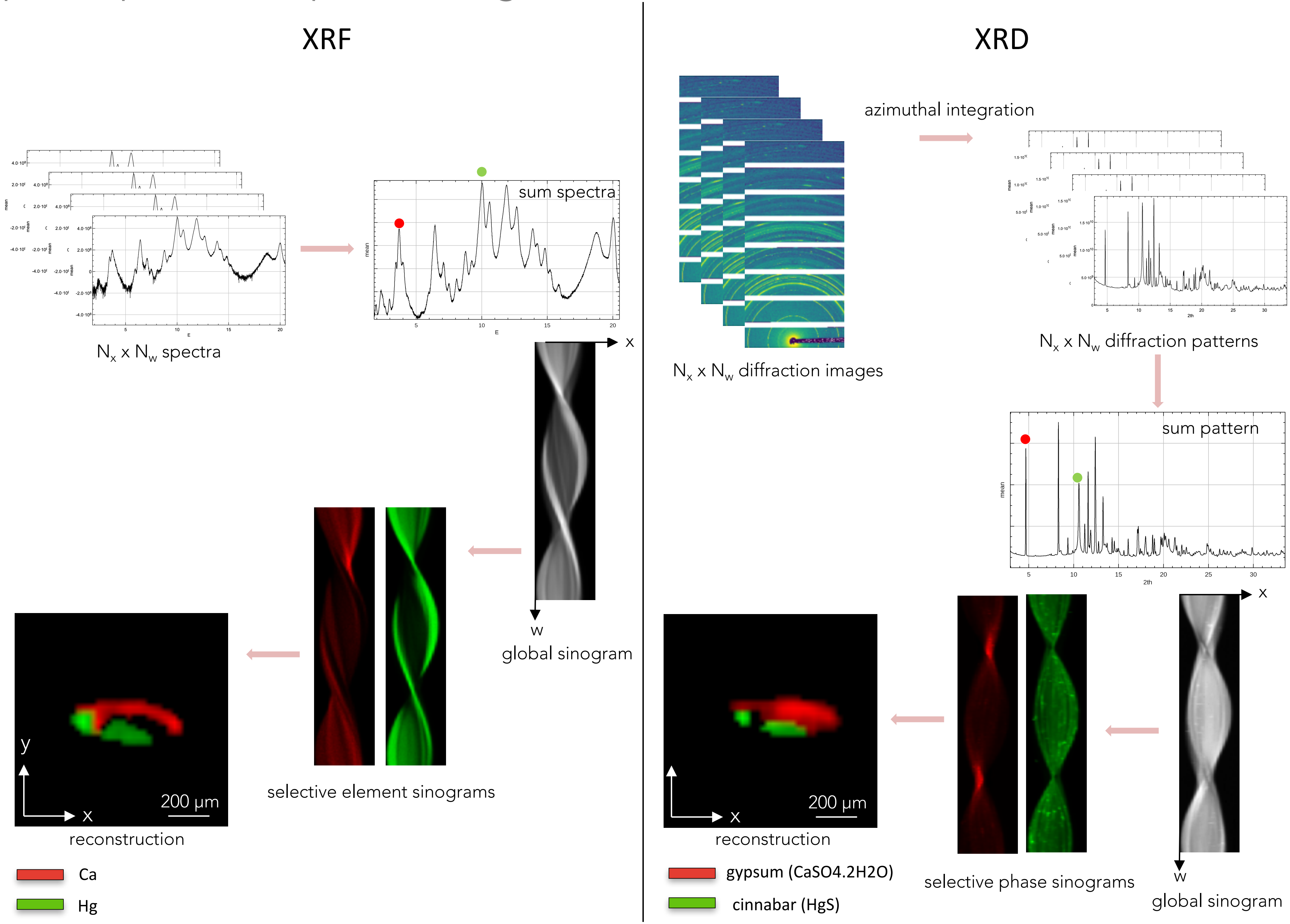
µXRF/µXRD-CT processing workflow



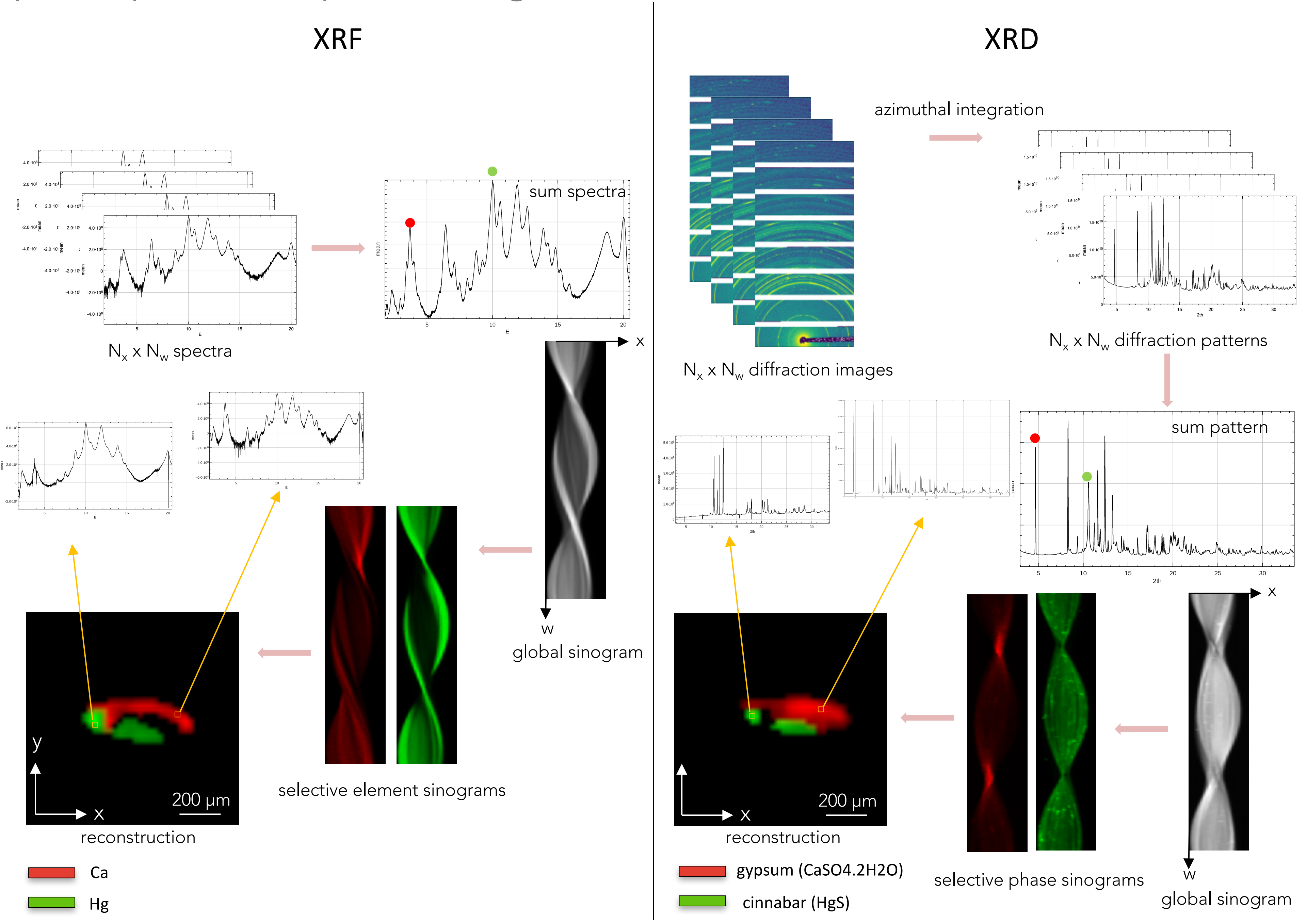
µXRF/µXRD-CT processing workflow



µXRF/µXRD-CT processing workflow



µXRF/µXRD-CT processing workflow



Use of PyFAI+Jupyter Notebook

2nd case XRD in transmission and tomography mode

import dedicated libraries/modules

azimuthal integration

definition of functions

results

```
In [1]: %pylab nbagg

Populating the interactive namespace from numpy and matplotlib

In [2]: import fabio
        import pyFAI
        import sys
        sys.path.append("/mntdirect/_data_bm02/SCRIPTS/")
        import spec_reader_nb
        import pickle
        import glob
        import os
        from skimage.transform import iradon
        from skimage.transform import iradon_sart
```

Use of PyFAI+Jupyter Notebook
2nd case XRD in transmission and tomography mode

import dedicated libraries/modules

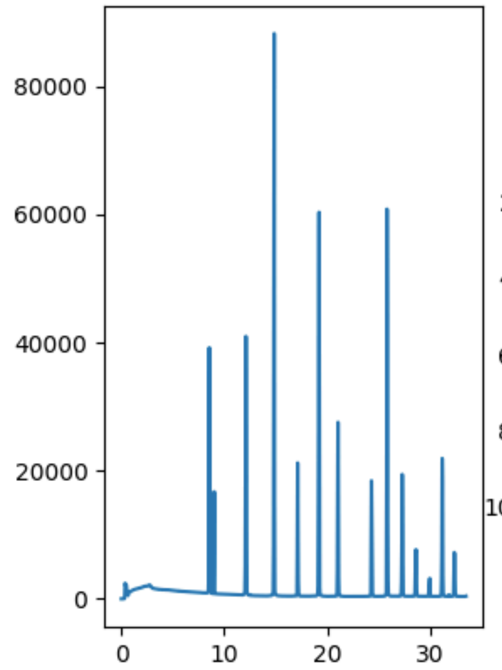
azimuthal integration

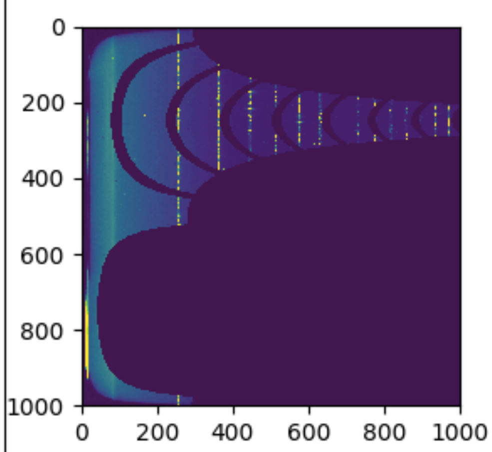
definition of functions

results

```
In [112]: ai=pyFAI.load('/data/bm02/nblanc2/IH-HG-9/raw/18Nov30D5_0680.poni')
```

```
In [113]: LaB6=fabio.open('/data/bm02/nblanc2/IH-HG-9/raw/18Nov30D5_0680.edf')
          figure()
          subplot(1,2,1)
          ILaB6=ai.integrate1d(LaB6.data,1000,unit='2th_deg')
          plot(*ILaB6)
          subplot(1,2,2)
          I2dLaB6=ai.integrate2d(LaB6.data,1000,1000,unit='2th_deg')
          imshow((I2dLaB6[0]),vmax=5000)
```





Use of PyFAI+Jupyter Notebook
2nd case XRD in transmission and tomography mode

import dedicated libraries/modules

azimuthal integration

definition of functions

In [19]:

```
def listscan(dataspecfilename,samplename) :
    dataspec=spec_reader_nb.SpecFile(dataspecfilename)
    listscannum=[]
    normval=[]
    listscan=[]

    for val, keys in dataspec.scan_dict.items():
        listscannum.append(val)
    listscannum=array(listscannum)

    for i in listscannum:
        scan=spec_reader_nb.Scan(dataspec,int(i))
        if (scan.comments) and (scan.comments.split()[5] == samplename) :
            listscan.append((int(scan.comments.split()[10]),i))
            normval.append((scan.zap_vct4,scan.tphi,scan.motors['tsy'][0],scan.motors['tsz'][0]))

    #print(listscannum)
    #print(normval)
    #print(listscan)

    return normval,listscan,samplename

def integrscan(listscan,poni,eachtsz=True) :
    ai = pyFAI.load(poni)
    imgnum = array(listscan[1]).T[0]
    radix = listscan[2]
    foldername = '/data/bm02/nblanc2/IH-HG-9/raw/zap/' + radix + '/'
    scanlist=[radix + '_d5_{0:0>4d}_0000_0000.edf'.format(num) for num in imgnum]
    data=[]
    #print(scanlist)
    tszunique=0

    for num, val in enumerate(scanlist):
        #print('scan :', val)
        stack=[]

        for frame in fabio.open(foldername + val) :
            stack.append(frame)
            #print(len(stack))
            #print(stack)
        if len(stack) != 360 :
```

results

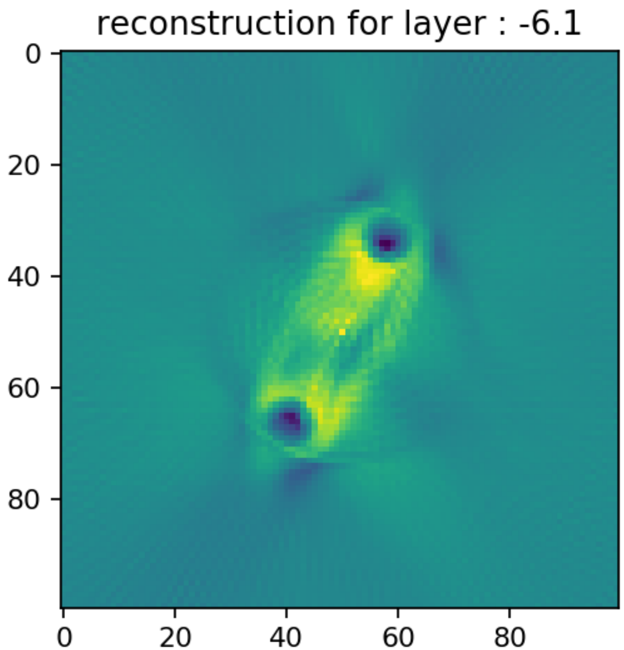
etc ...

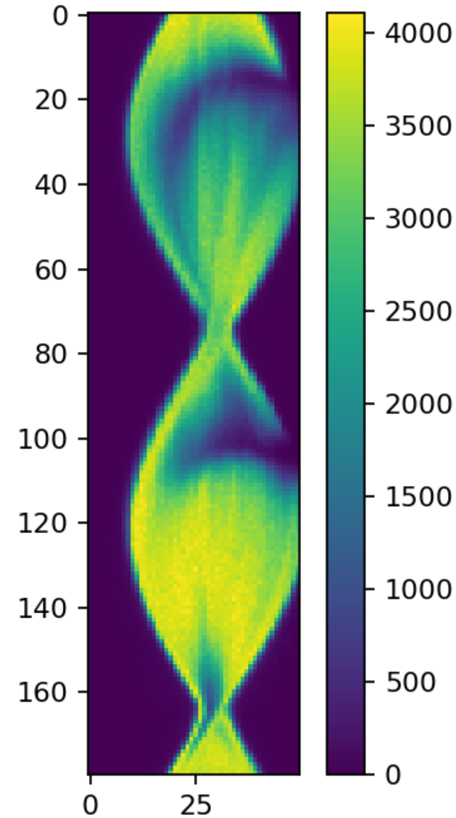
Use of PyFAI+Jupyter Notebook
2nd case XRD in transmission and tomography mode

import dedicated libraries/modules

azimuthal integration

definition of functions





results

Conclusion

- PyFAI
  - useful library to process huge amount of data
  - very efficient for azimuthal integration
  - fast and versatile
  - a lot of libraries/modules

- Jupyter Notebook
  - super practical to use during an experiment and come back later
  - easy to interact with
  - easy to share
  - easy to keep as a logbook

-> PyFAI + Jupyter Notebook: powerful combination for data processing thank you for your attention