NoBUGS 2024
PyFAI satellite

Edgar Gutierrez Fernandez

PIONEERING SYNCHROTRON SCIENCE

ESRF The European Synchrotron

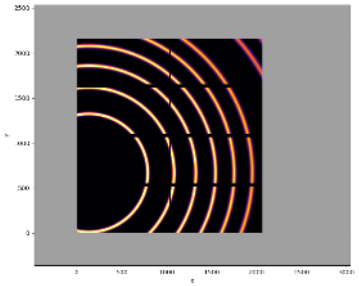
30 years of science

Single frame pipeline in pyFAI: standard azimuthal integration

Data Acquisition

2D array

Raw data



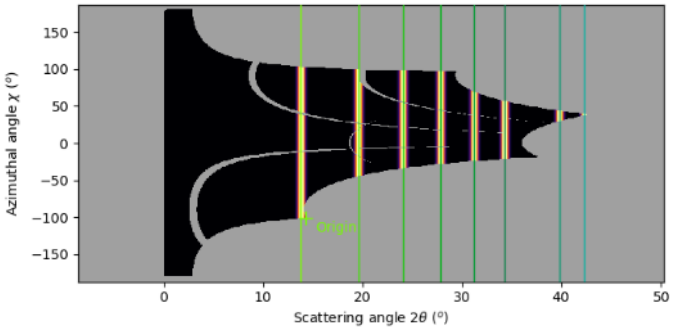
Data Reduction

PyFAI

Fast Azimuthal Integration

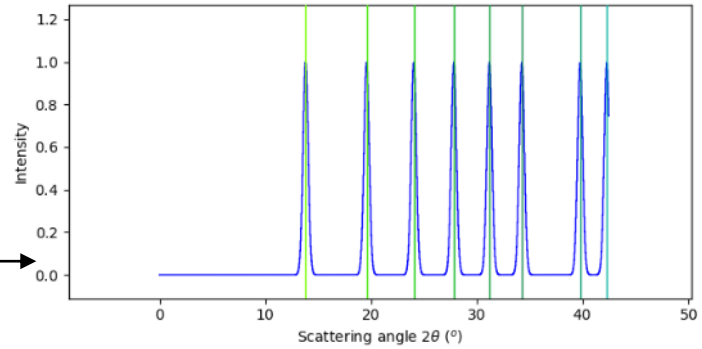
2D array

Reshaped map into azimuthal angle and scattering angle coordinates (or customized units)



1D array

Azimuthal averaged intensity around the position of the direct beam
Scattering angle units



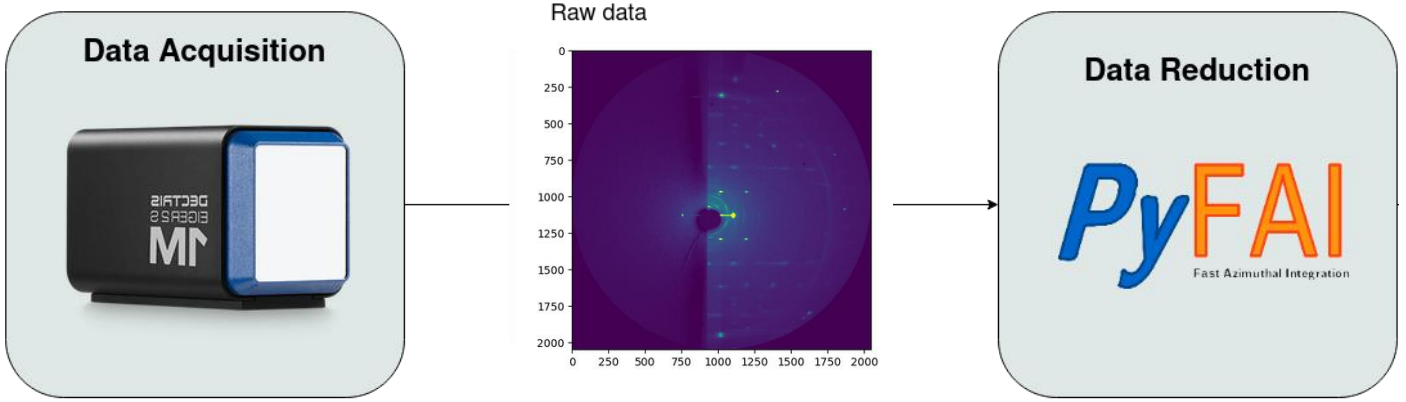
$$data(pix_x, pix_y) \rightarrow data(\chi(°), q(nm^{-1}))$$

x = 0 y = 0	x = 1 y = 0
x = 0 y = 1	x = 1 y = 1

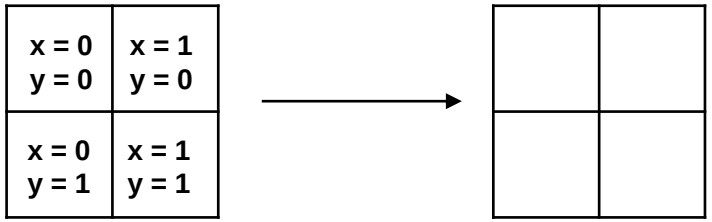
χ = 91 q = 3.5	χ = 90 q = 3.4
χ = 92 q = 3.4	χ = 91 q = 3.2

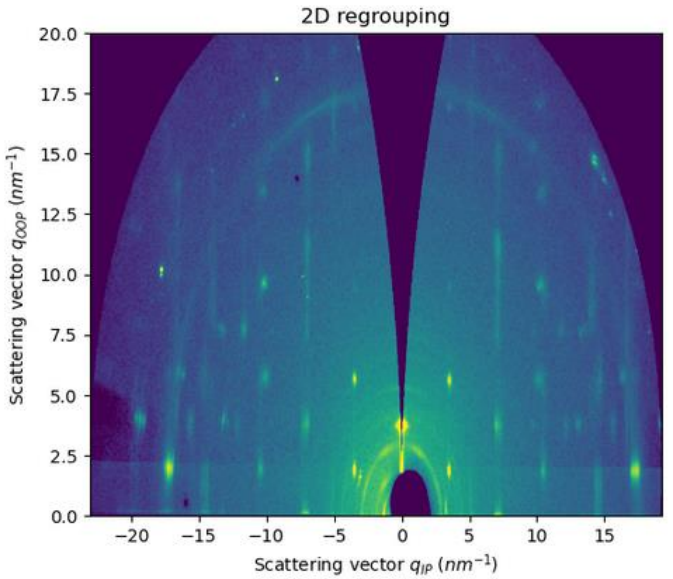
$$I(q) = \int_{-180°}^{180°} data(\chi, q)\, d\chi$$

Single frame pipeline in pyFAI: generic data integration



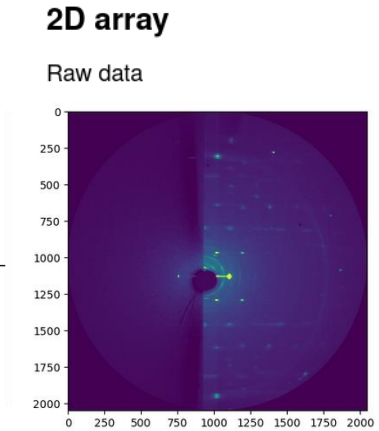
$$data(pix_x, pix_y) \rightarrow data(unit_1, unit_2)$$

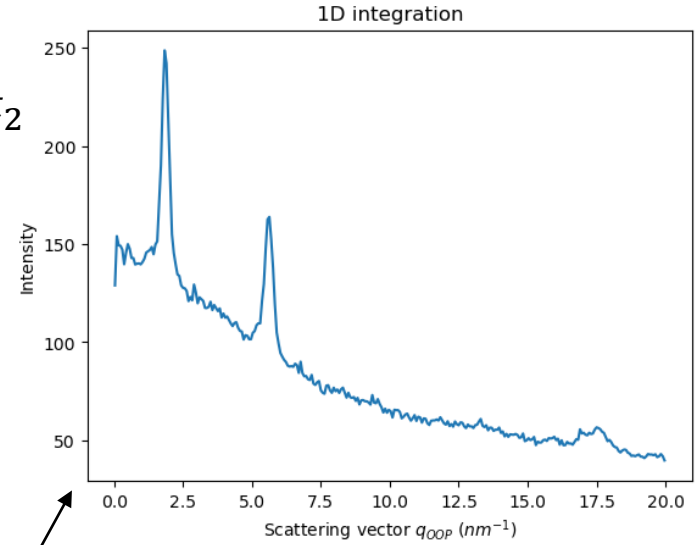


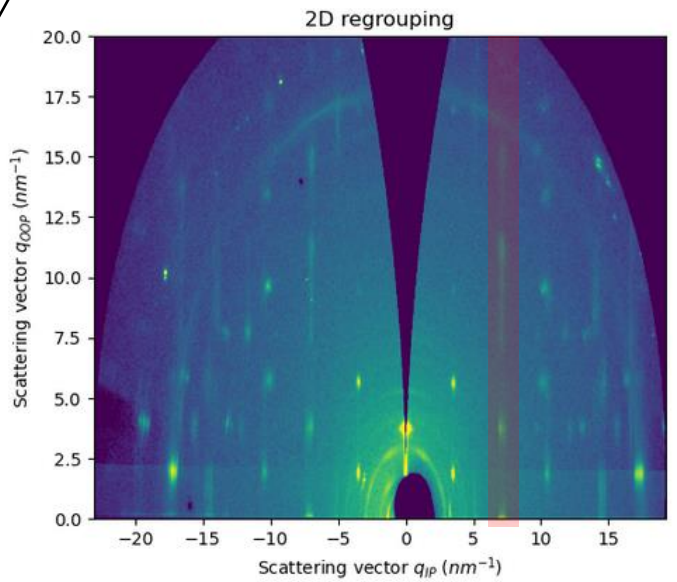


Single frame pipeline in pyFAI: generic data integration

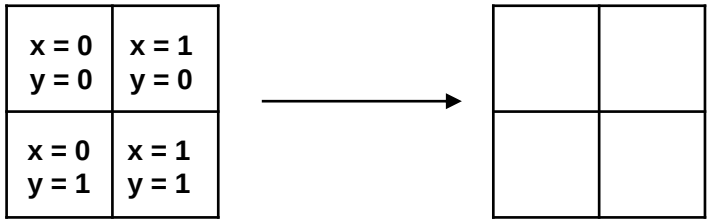
$$I(unit_1) = \int_a^b data(unit_1, unit_2)\, dunit_2$$



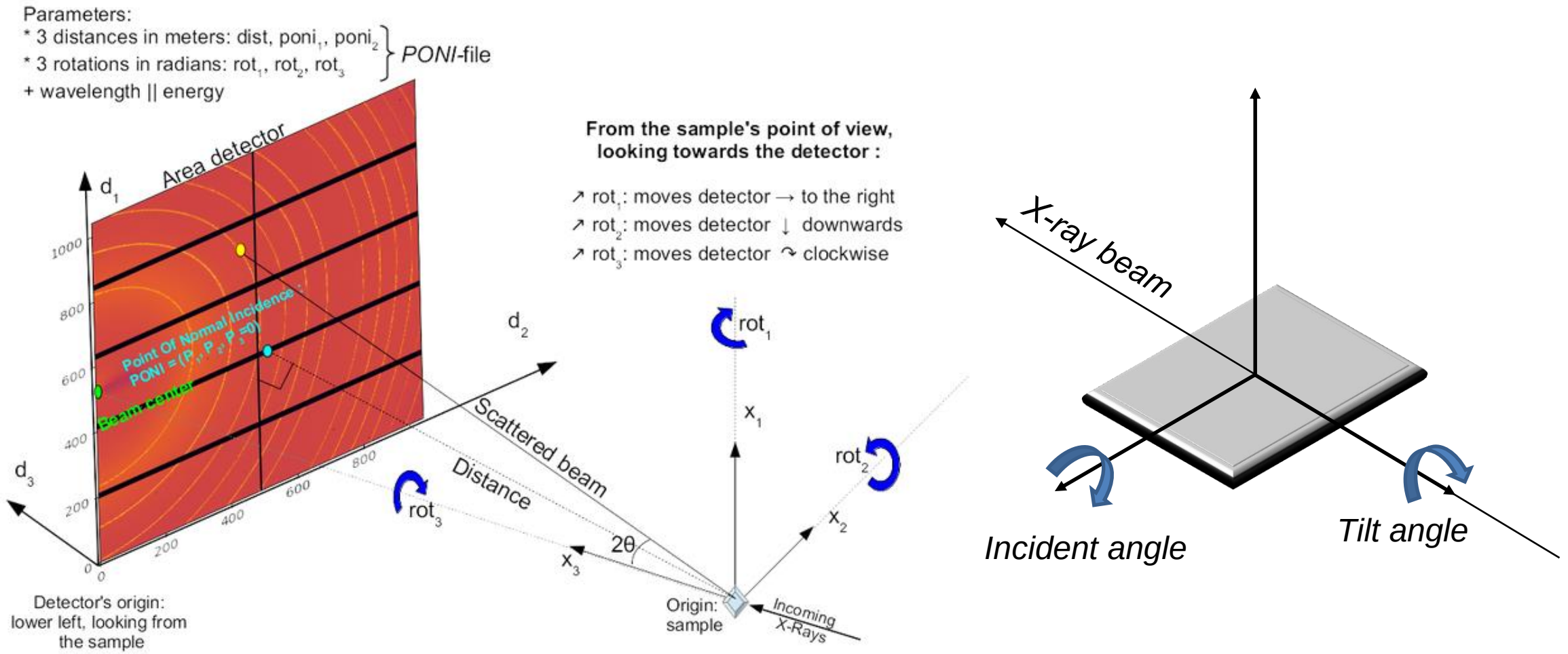




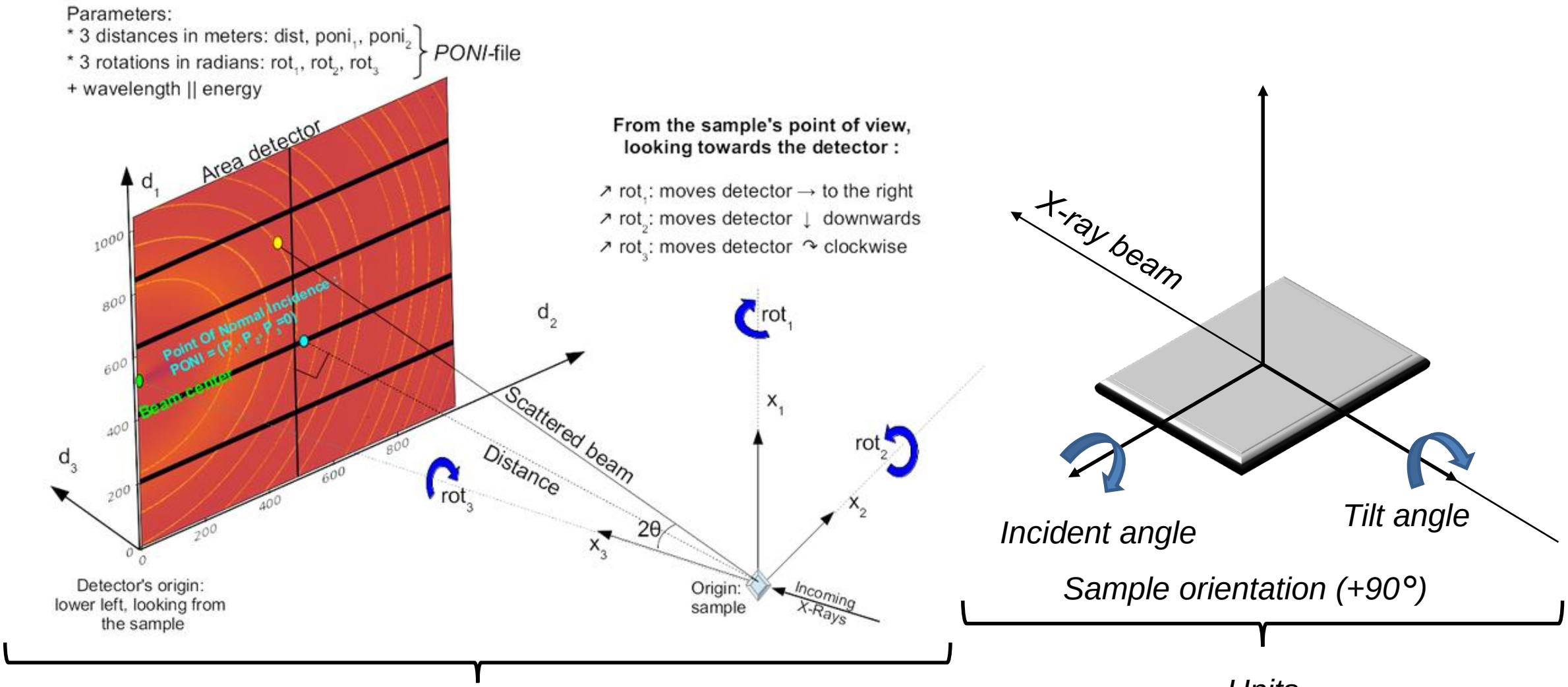
$$data(pix_x, pix_y) \rightarrow data(unit_1, unit_2)$$



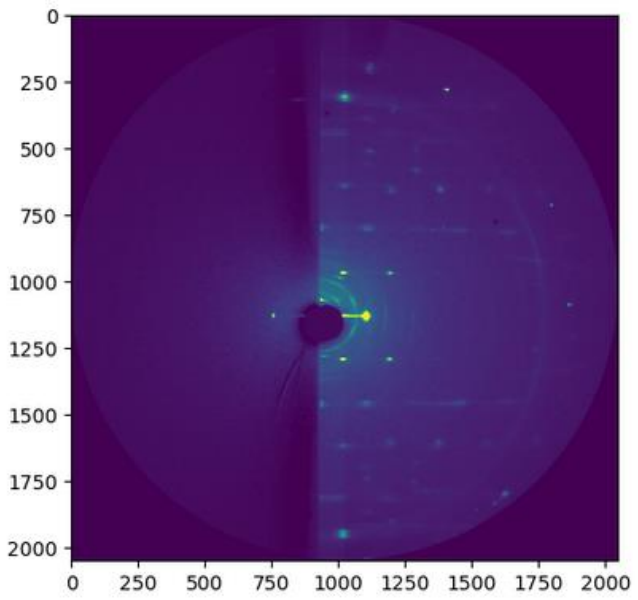
pyFAI geometry + two rotations



pyFAI geometry + two rotations



pyFAI geometry: components of the q vector



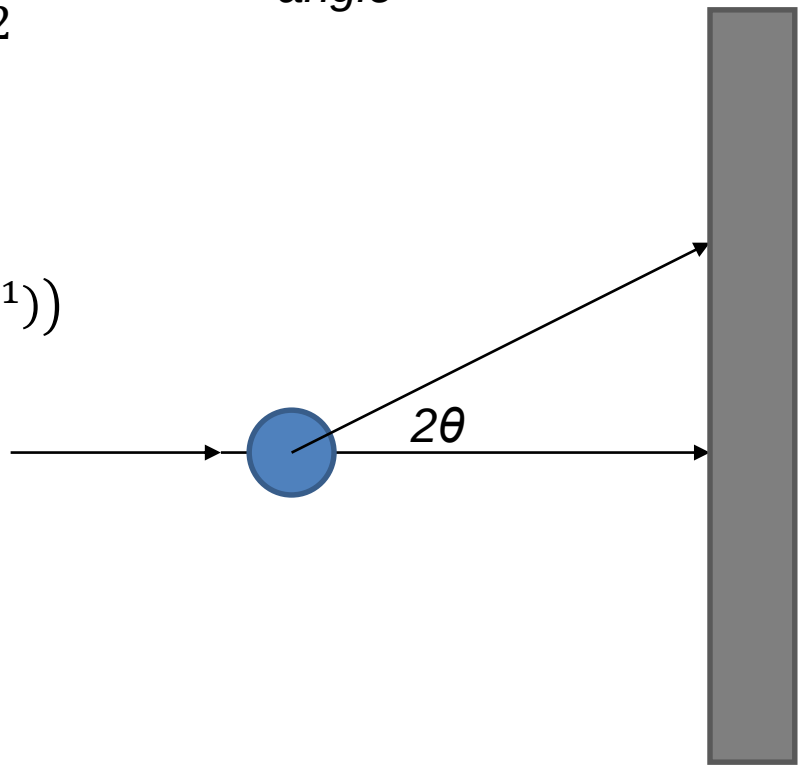
Bragg Law

$$|\boldsymbol{q}| = \frac{4\pi}{\lambda}\sin\frac{2\theta}{2}$$

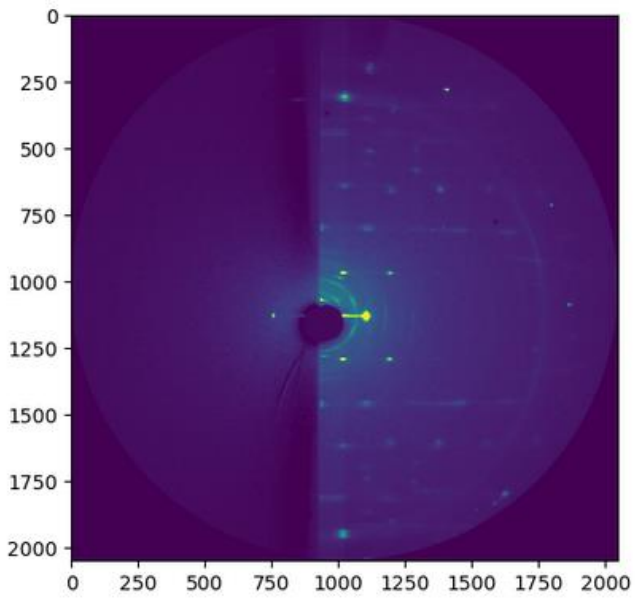
Scattering angle

$$data(pix_x, pix_y) \rightarrow data(\chi(°), q(nm^{-1}))$$

$$I(q) = \int_{-180°}^{180°} data(\chi, q)\, d\chi$$



pyFAI geometry: components of the q vector



Bragg Law

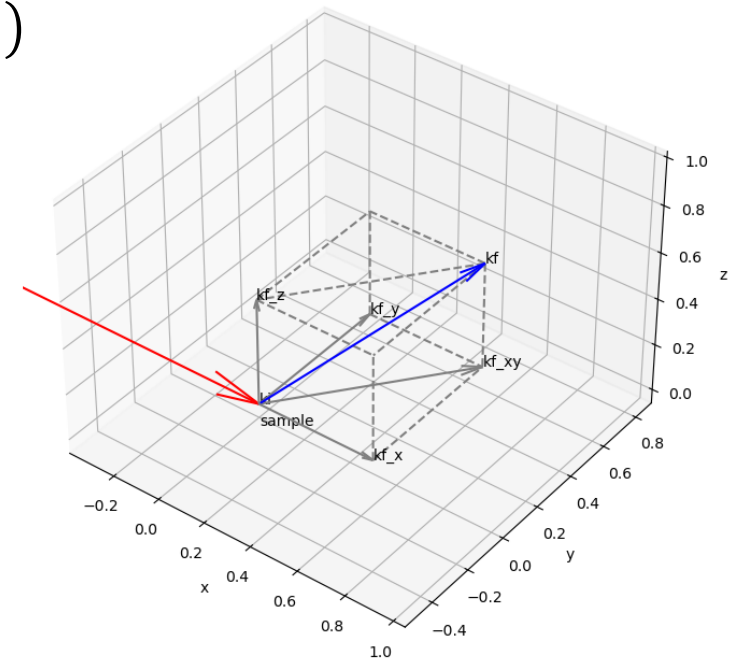
$$|\boldsymbol{q}| = \frac{4\pi}{\lambda}\sin\frac{2\theta}{2}$$

Scattering angle

$$data(pix_x, pix_y) \rightarrow data(\chi(°), q(nm^{-1}))$$

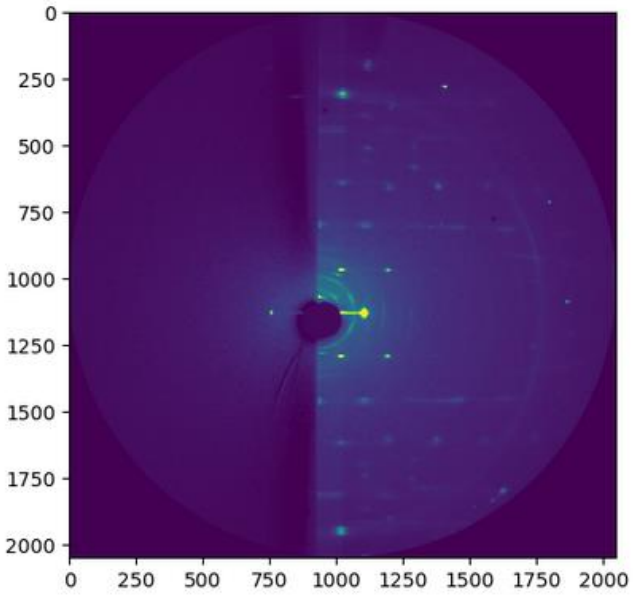
$$I(q) = \int_{-180°}^{180°} data(\chi, q)\, d\chi$$

$$q_x = f(pix_x, pix_y, incident, tilt)$$
$$q_y = f(pix_x, pix_y, incident, tilt)$$
$$q_z = f(pix_x, pix_y, incident, tilt)$$

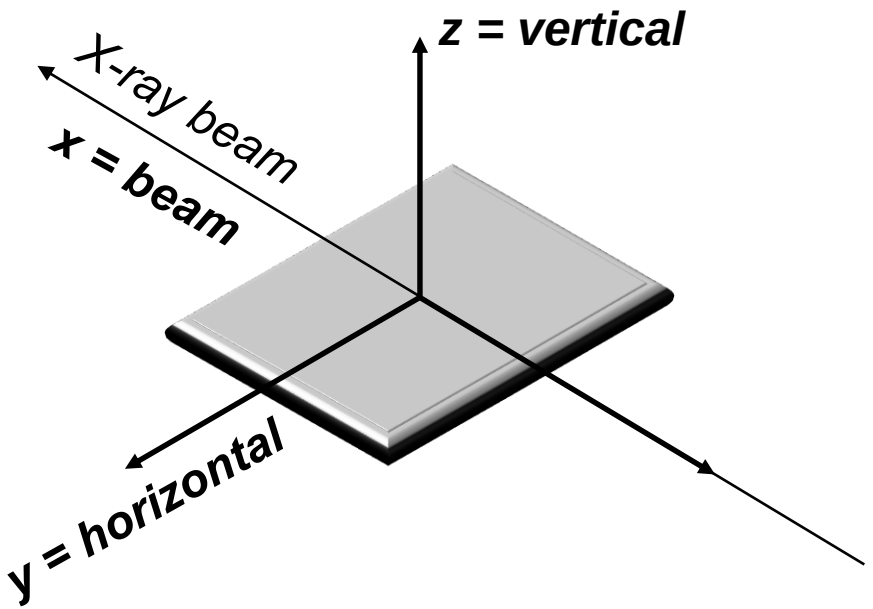


pyFAI geometry: components of the q vector

$$data(pix_x, pix_y) \rightarrow data(\chi(°), q(nm^{-1})) \rightarrow data\left(\sqrt{{q_x}^2 + {q_y}^2}, q_z\right) = data(q_{IP}, q_{OOP})$$

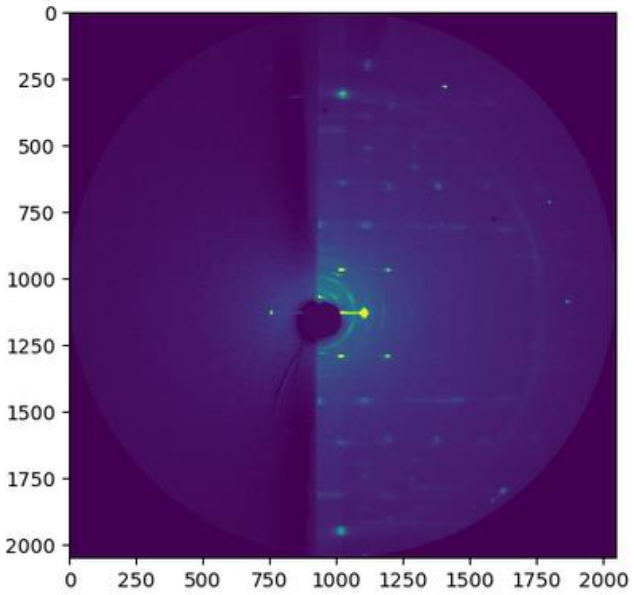


$$q_{IP} = \sqrt{{q_x}^2 + {q_y}^2} \qquad q_{OOP} = q_z$$

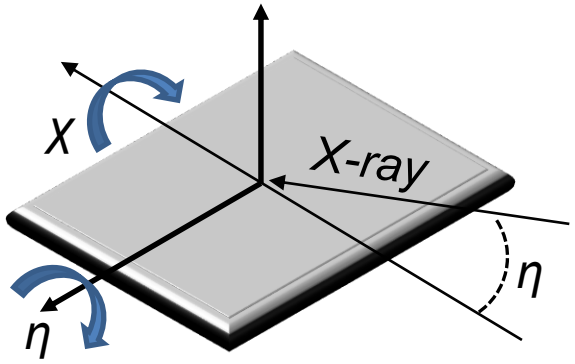


pyFAI geometry: components of the q vector

$$data(pix_x, pix_y) \rightarrow data(\chi(°), q(nm^{-1})) \rightarrow data\left(\sqrt{{q_x}^2 + {q_y}^2}, q_z\right) = data(q_{IP}, q_{OOP})$$



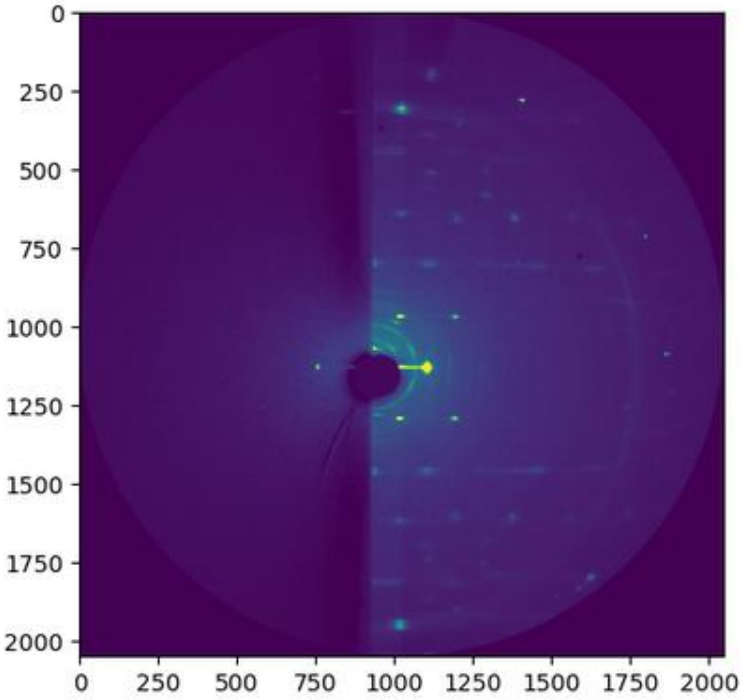
$$q_{IP} = \sqrt{{q_x}^2 + {q_y}^2} \qquad q_{OOP} = q_z$$

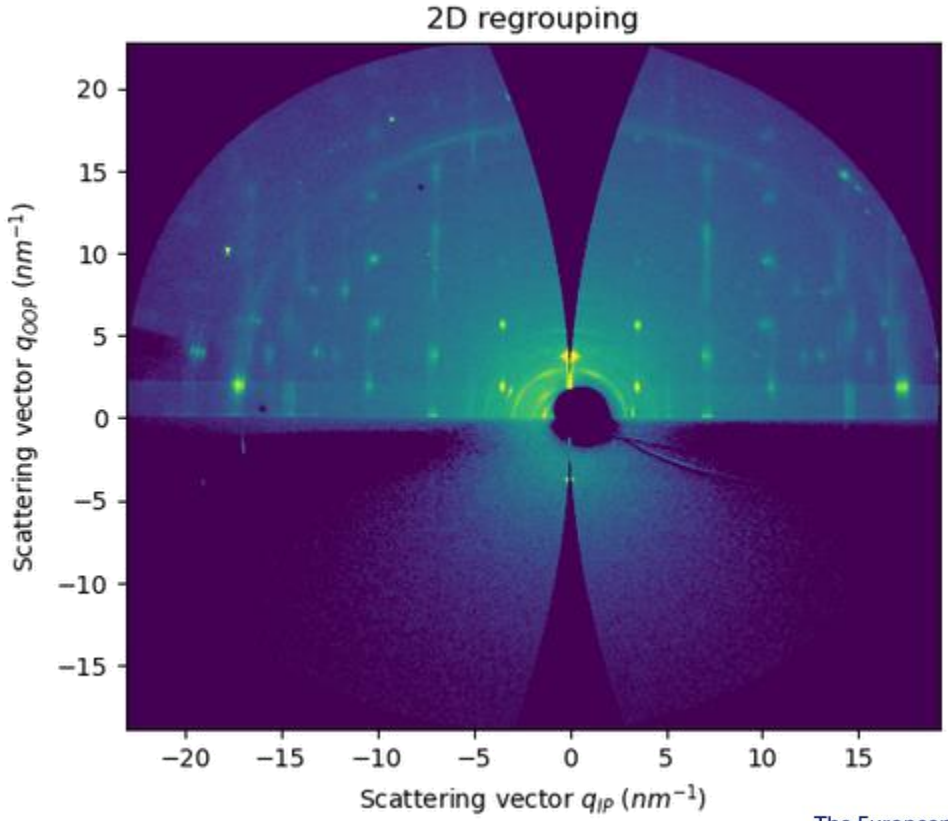


pyFAI geometry: components of the q vector

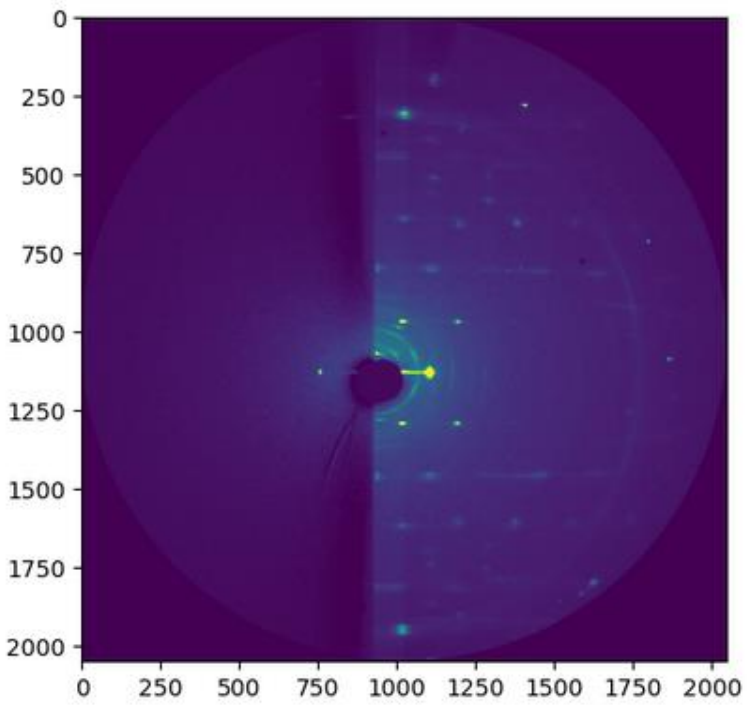
$$data(pix_x, pix_y) \rightarrow data(\chi(°), q(nm^{-1})) \rightarrow data\left(\sqrt{{q_x}^2 + {q_y}^2}, q_z\right) = data(q_{IP}, q_{OOP})$$

$$q_{IP} = \sqrt{{q_x}^2 + {q_y}^2} \qquad q_{OOP} = q_z$$





GrazingIncidence / Fiber integrate2d (caking)



1) Instantiate the units

- ***from pyFAI.units import get_unit_fiber***
- ***qip = get_unit_fiber(name='qip_nm^-1', incident_angle=0.12, tilt_angle=0.0, sample_orientation=2)***
- ***qoop = get_unit_fiber(name='qoop_nm^-1', incident_angle=0.12, tilt_angle=0.0, sample_orientation=2)***

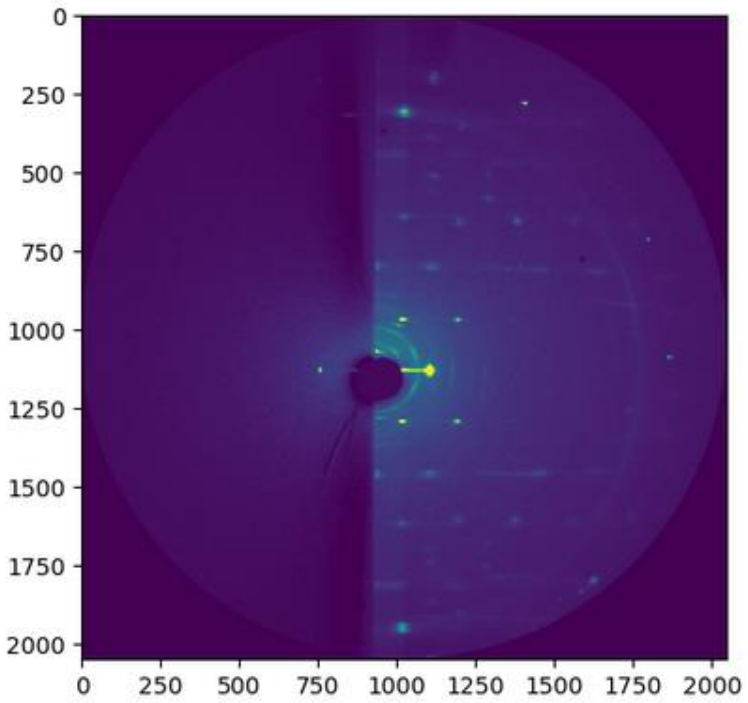
2) Integrate using different units and a method without pixel splitting

- ***units = (qip,qoop)***
- ***method = ("no", "csr", "cython")***
- ***res2d_gi = ai.integrate2d(data=data, npt_rad=1000, npt_azim=1000, unit=units, method=method)***

3) Plot the results:

- ***from pyFAI.gui.jupyter import plot2d***
- ***plot2d(res2d_gi)***

GrazingIncidence / Fiber integrate2d (caking)



1) Instantiate the units

- ***from pyFAI.units import get_unit_fiber***
- ***qip = get_unit_fiber(name='qip_nm^-1', incident_angle=0.12, tilt_angle=0.0, sample_orientation=2)***
- ***qoop = get_unit_fiber(name='qoop_nm^-1', incident_angle=0.12, tilt_angle=0.0, sample_orientation=2)***

2) Integrate using different units and a method without pixel splitting

- ***units = (qip,qoop)***
- ***method = ("no", "csr", "cython")***
- ***res2d_gi = ai.integrate2d(data=data, npt_rad=1000, npt_azim=1000, unit=units, method=method)***

3) Plot the results:

- ***from pyFAI.gui.jupyter import plot2d***
- ***plot2d(res2d_gi)***