ESRF user meeting tutorial 2026 T10
Advanced session on pyFAI

ESRF | The European Synchrotron

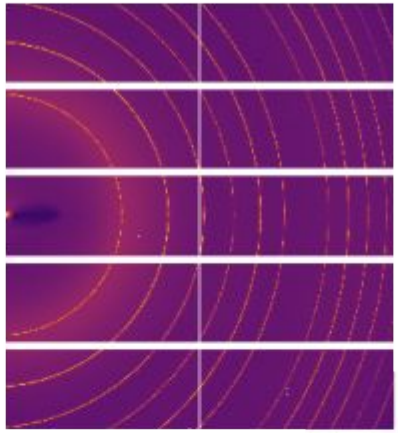
Powder diffraction with spotty data

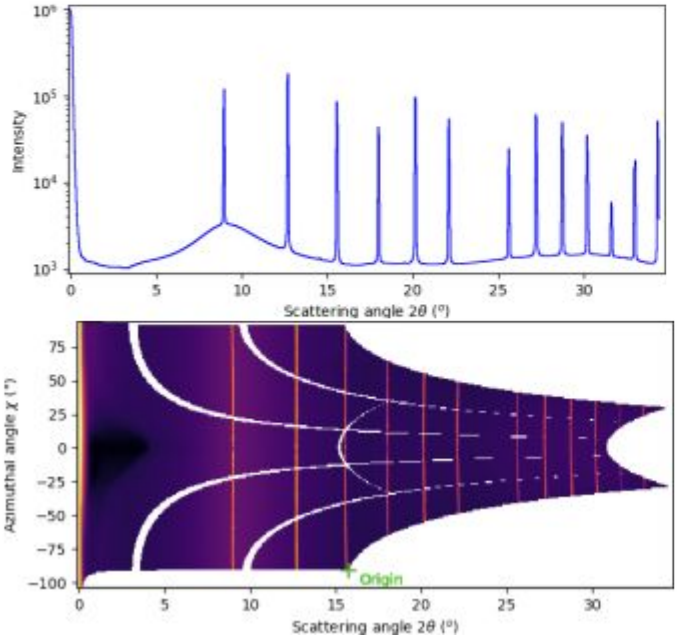
- Why are my diffraction frames spotty ?
- How to separate the powder signal from larger crystallites ?
- Assessment of the spottiness
- Advanced re-binning tricks.

Fast Azimuthal Integration using Python

PyFAI

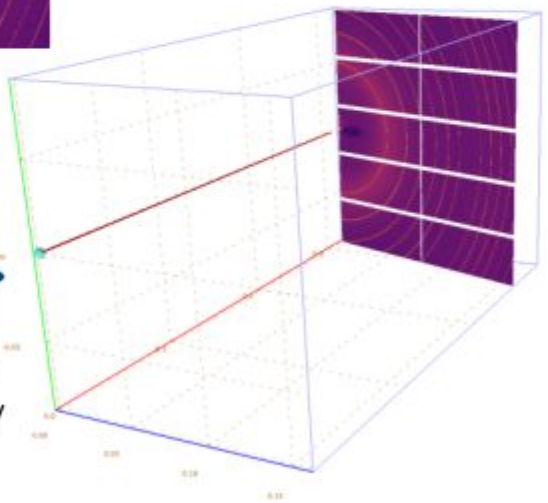
Fast Azimuthal Integration





Average
1d and 2d

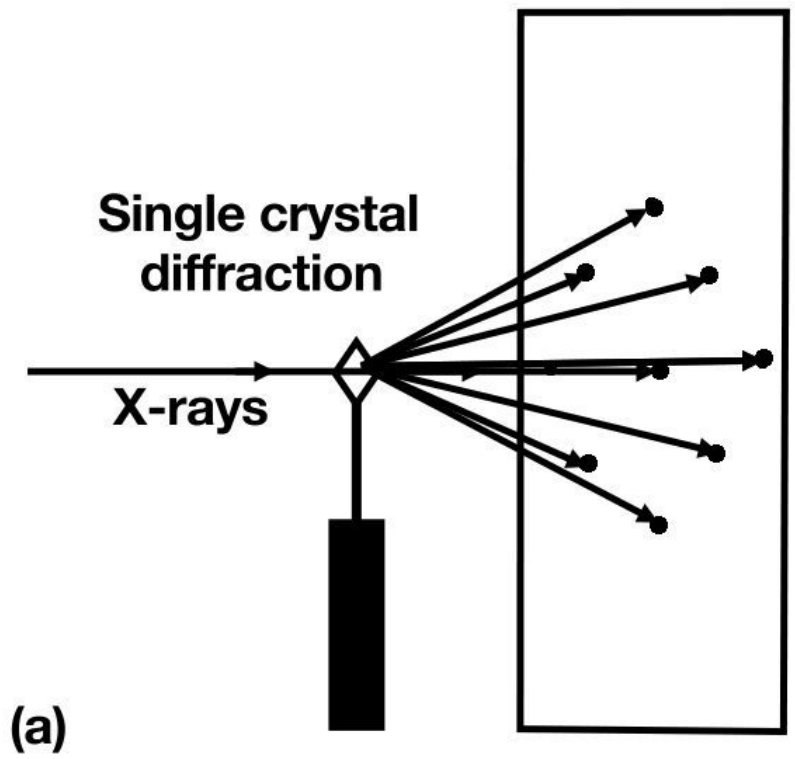
Calibrate
3d geometry

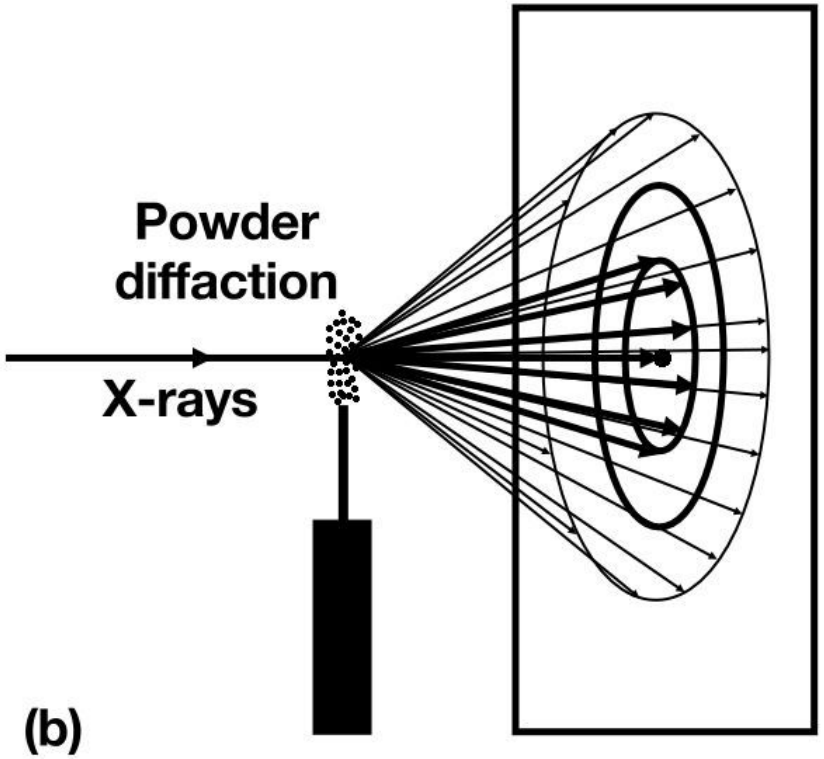


Kieffer, Jerome; Valls, Valentin; Gutierrez-Fernandez; Edgar, Ashiotis; Giannis, Karkoulis, Dimitris; Nawaz, Zubair; Deschildre, Aurore; Vincent, Thomas; Picca, Frederic Emmanuel; Massahud, Emily; Payno, Henri; Huder, Loïc; Wright, Johnathan Paul; Pandolfi, Ronald; Jankowski, Maciej; Paleo, Pierre; Faure, Bertrand; Storm, Malte; De Nolf, Wout; Wright, Christopher J.; Hopkins, Jesse B,; Pascal, Elena; Weninger, Clemens; Detlefs, Carsten; Plaswig, Florian; Lavanchy, Aurelien; gbenecke; zxs-un; iltommi11; dodogerstlin; harshal301002; Lotze, Gudrun

https://zenodo.org/records/17984195

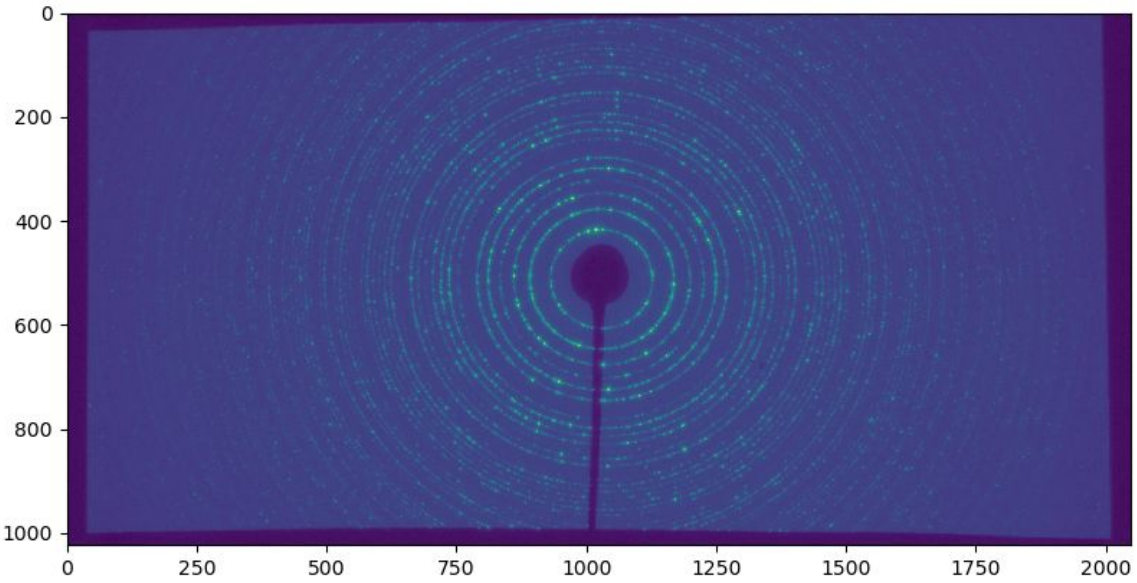
Single crystal diffraction vs powder diffraction





Dealing with spotty data:

- **Too few crystallites in the beam**
  - Expose longer
  - Shake the sample
  - Use larger beam
  - Focus on detector, not on the sample
  - Grind your sample
  - …



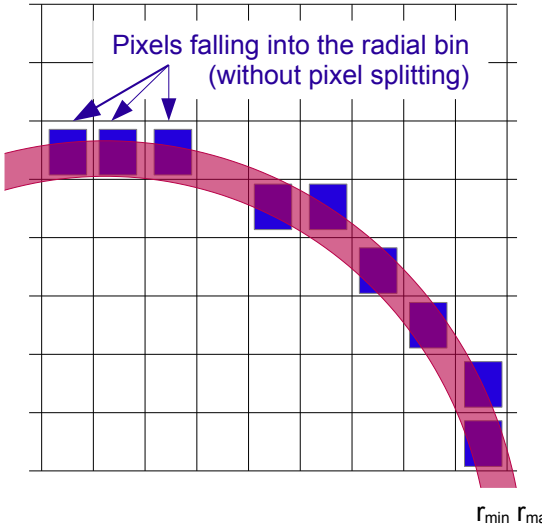
How azimuthal integration works

- **Pixel-wise corrections:**

$$I_{cor} = \frac{I_{raw} - I_{dark}}{F \cdot \Omega \cdot P \cdot A \cdot I_0} = \frac{signal}{normalization}$$

Where: I_0 is the incoming flux (pixel independent)

  - I_{raw} and I_{dark} are the signal measured from the detector
  - F is the flat-field correction
  - Ω is the solid angle for this pixel
  - P is the polarization factor
  - A is the parallax correction factor



- **Weighted average of I_{cor} over a ring:**

$$\langle I \rangle_r = \frac{\sum \omega \cdot I_{cor}}{\sum \omega}$$

  - Fraction of area $c_{i,r}$ of a pixel (i) in the bin (r)
    - **fraction of the pixel i contributing to bin_r**
    - **accounts for pixel splitting**
  - Normalization factor

Stored as sparse matrix

$$\langle I \rangle_r = \frac{\sum_{i \in bin_r} (c_i \cdot normalization_i) \cdot \left\{ \frac{signal_i}{normalization_i} \right\}}{\sum_{i \in bin_r} c_i \cdot normalization_i}$$

$$\langle I \rangle_r = \frac{\sum_{i \in bin_r} c_i \cdot signal_i}{\sum_{i \in bin_r} c_i \cdot normalization_i}$$

- **Use accumulator to simplify calculations:**

$$V = \sum \omega \cdot v = \sum c \cdot signal$$

$$\Omega = \sum \omega = \sum c \cdot normalization$$

$$\langle I \rangle = \frac{V}{\Omega}$$

Math from Kieffer et al.; *J. Appl. Cryst,* (2025)
https://doi.org/10.1107/S1600576724011038

Uncertainties in azimuthal integration

- **Poisson-like uncertainty:**
  - $\sigma^2 = I$ or $f(I)$
- **Std:** $$\sigma(\langle I \rangle_r) = \frac{\sqrt{\sum_{i \in bin_r} c_i^2 \cdot \sigma_i^2}}{\sum_{i \in bin_r} c_i \cdot normalization_i}$$
- **Sem:** $$\sigma(I_r) = \sqrt{\frac{\sum_{i \in bin_r} c_i^2 \cdot \sigma_i^2}{\sum_{i \in bin_r} c_i^2 \cdot normalization_i^2}}$$

- **Variance in the ring:**
  - $\sigma^2 = (I - \langle I \rangle)^2$
- **Std:** $$\sigma(\langle I \rangle_r) = \frac{\sqrt{\sum_{i \in bin_r} c_i^2 (I_i - \langle I \rangle)^2}}{\sum_{i \in bin_r} c_i \cdot normalization_i}$$
- **Sem:** $$\sigma(I_r) = \sqrt{\frac{\sum_{i \in bin_r} c_i^2 (I_i - \langle I \rangle)^2}{\sum_{i \in bin_r} c_i^2 \cdot normalization_i^2}}$$

Using accumulators:

$$VV = \sum_{i \in bin_r} c_i^2 \cdot \sigma_i^2$$

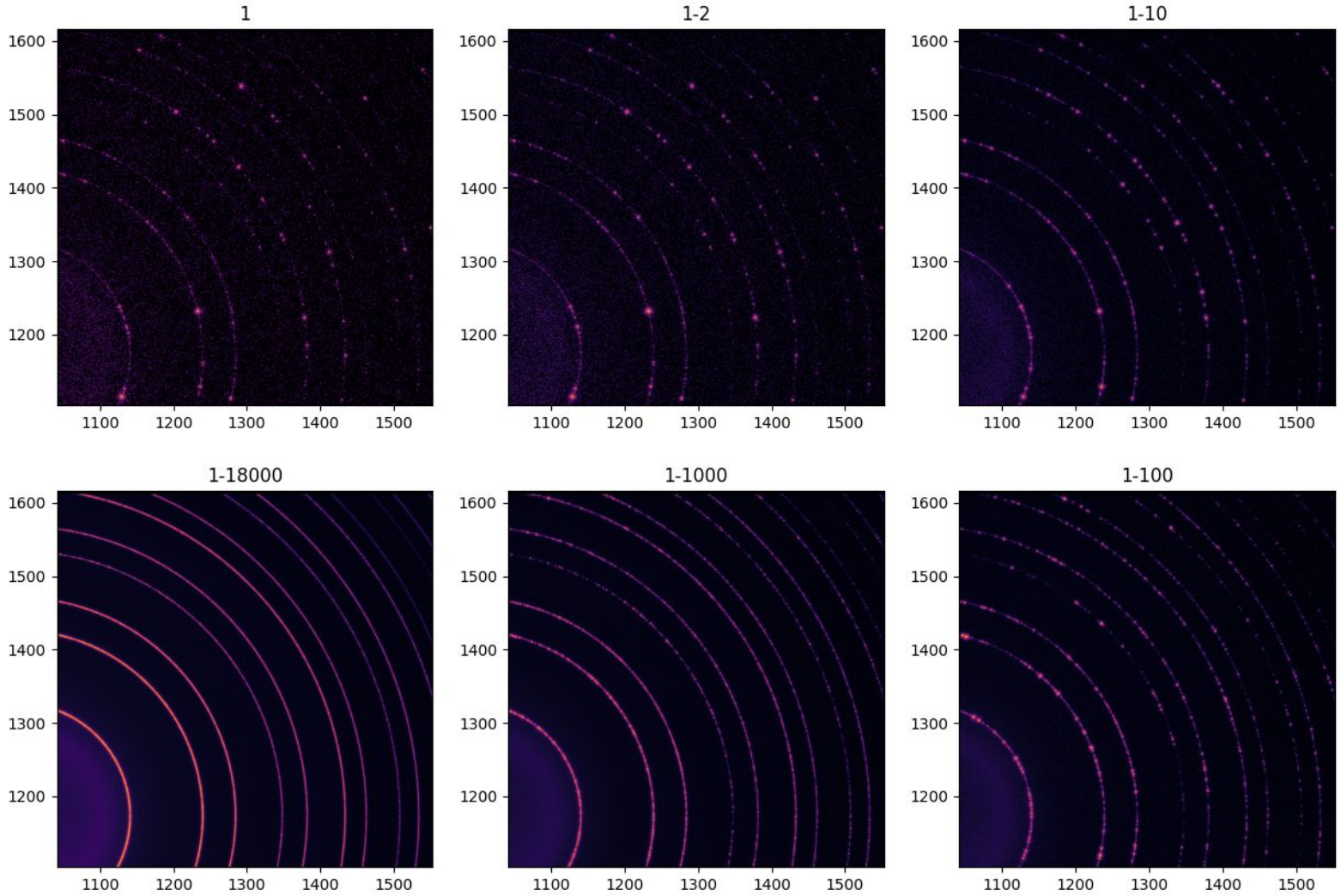
$$VV = \sum_{i \in bin_r} c_i^2 \left(I_i - \frac{V}{\Omega} \right)^2$$

$$\Omega\Omega = \sum \omega^2 = \sum_{i \in bin_r} c_i^2 \cdot normalization_i^2$$

$$\sigma(I) = \sqrt{\frac{VV}{\Omega\Omega}}$$

$$\sigma(\langle I \rangle) = \frac{\sqrt{VV}}{\Omega}$$

Accumulate more frames:



ESRF-ID11 @40keV / Eiger2 CdTe 4M detector @200Hz / SRM640 Silicon

Exercise

- **Download the data-file from**
  - https://www.silx.org/pub/pyFAI/pyFAI_UM_2026/eiger_0000.h5
  - Contains 200 frames
- **Accumulate the frames to reproduce the effect observed in the previous slide**
- **Nota:**
  - The dummy value is set to 65535 (dynamic masking from Eiger)
  - Despite the dataset is stored as `uint32`

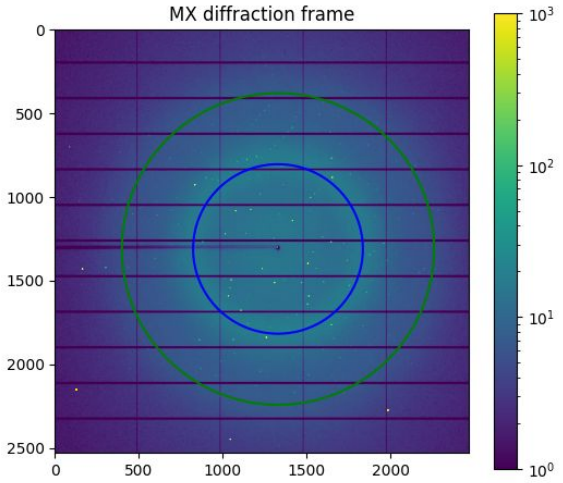
Outlier rejection

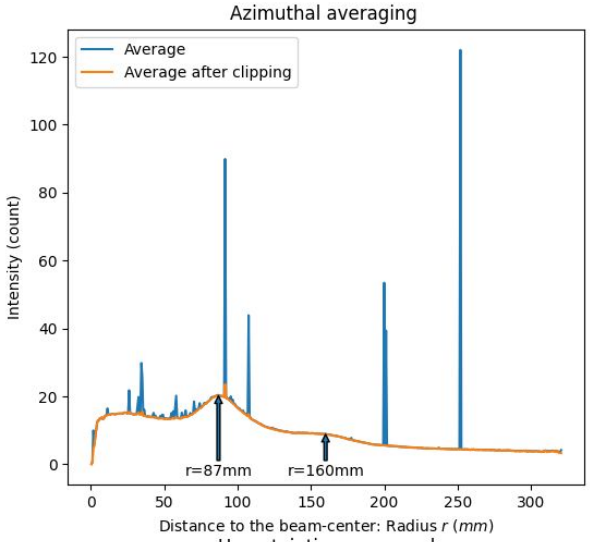
Outlier rejection algorithms:

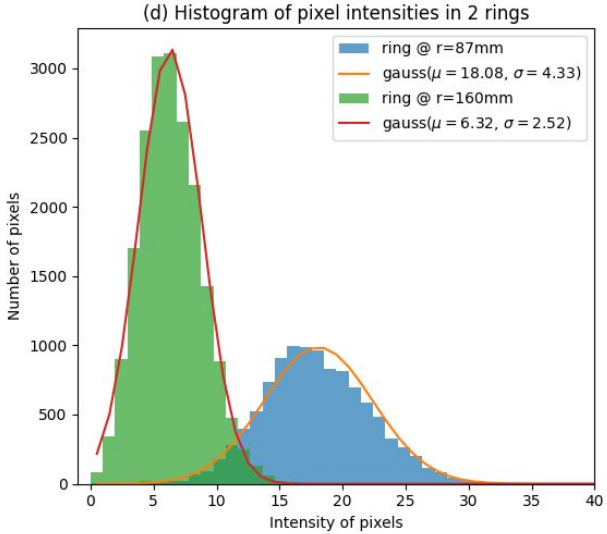
Sigma-clipping

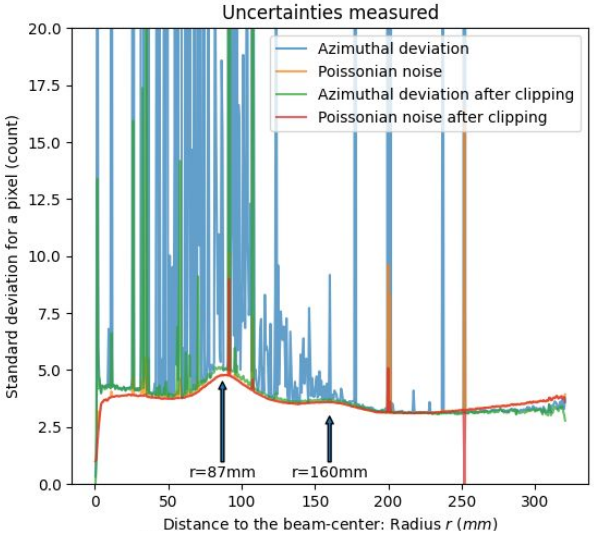
Median filtering

Sigma clipping to remove Bragg-peaks







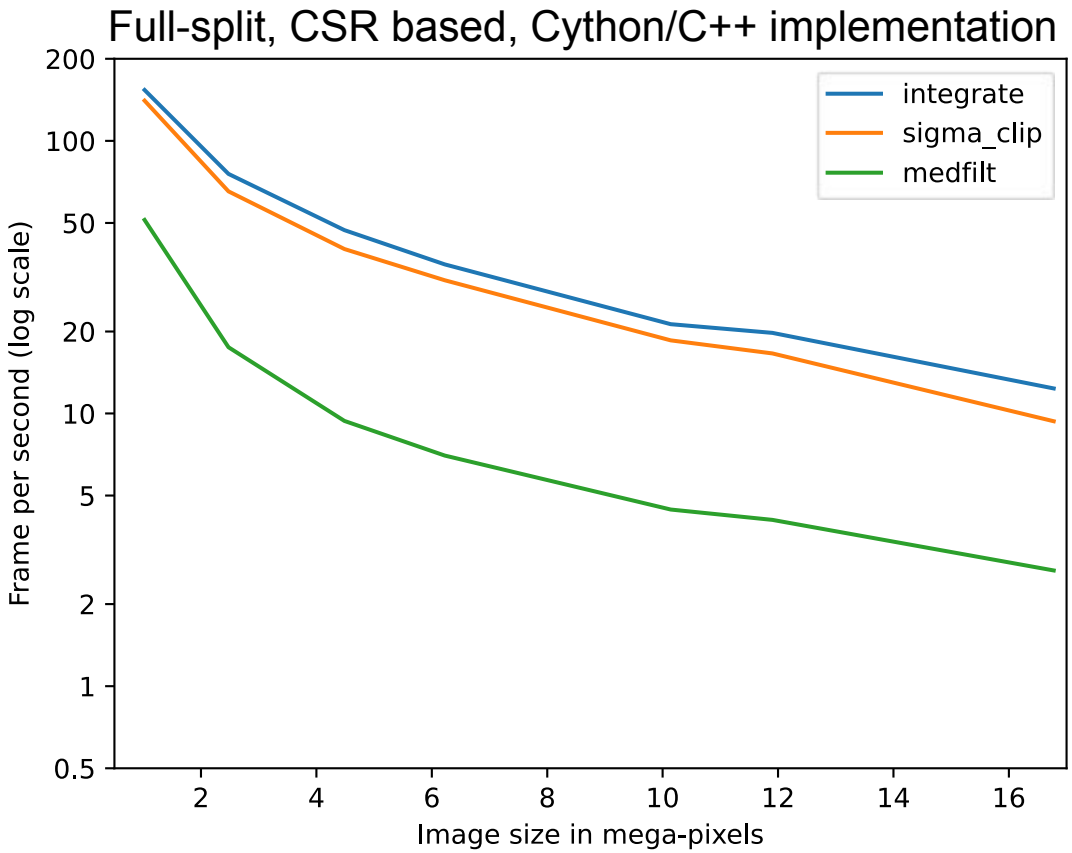


⚠ No pixel splitting ⚠

Kieffer et al. J. Appl. Cryst (2025)
https://doi.org/10.1107/S1600576724011038

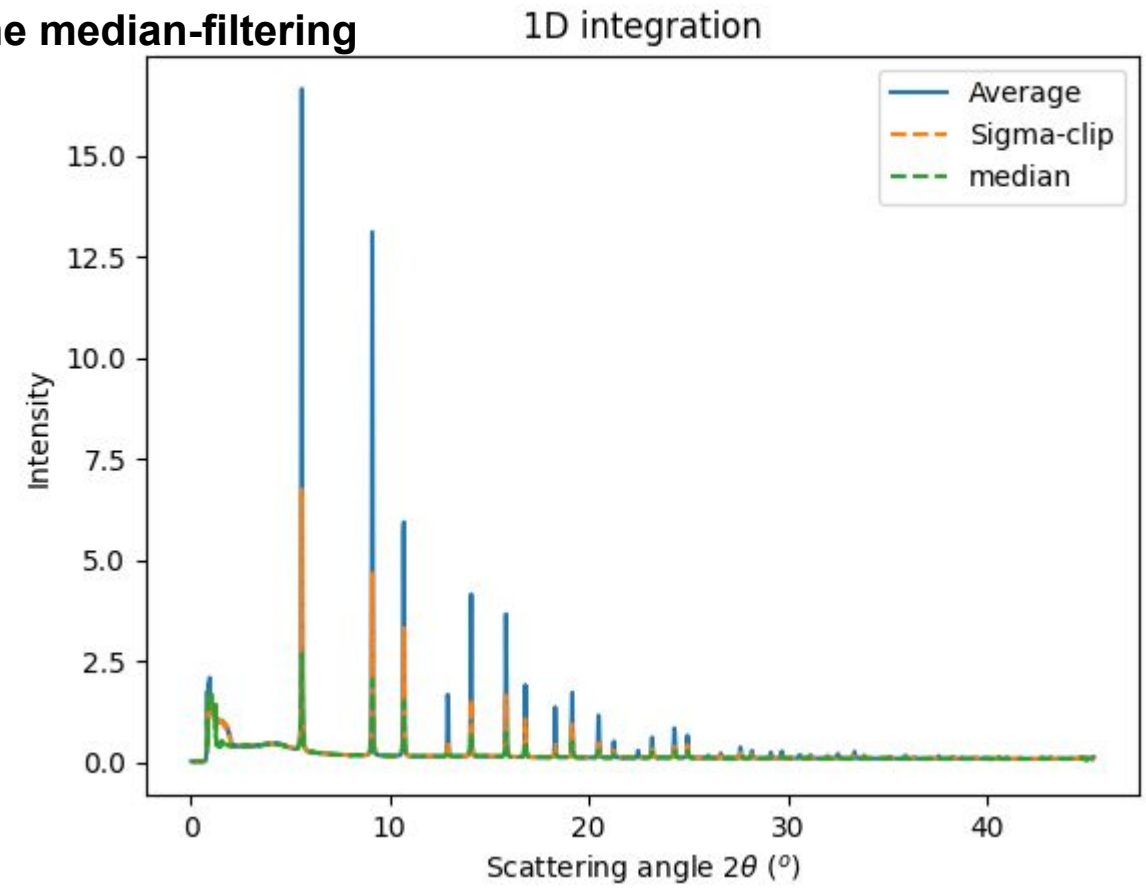
Median filtering / Quantile averaging in azimuthal space

- **Outlier rejection based on extrema rejection**
  - Oversampling for Rietveld refinement with large pixel size
- **Requires one sort per azimuthal bin, → expensive !**



Exercise:

- **Considering the average of 200 frames (one file)**
  - Overlay the plot of:
    - **The azimuthal integration**
    - **The sigma-clipping**
    - **The median-filtering**



Global Spottiness index:

- **Requirements:**
  - Integration performed with **azimuthal** error model
- **Formula:**

$$S = \sqrt{\left\langle \left\{ \frac{\sigma}{\bar{I}} \right\}^2 \right\rangle_{rings}} = \sqrt{\frac{1}{\mathrm{card}(rings)} \sum_{rings} \frac{VV}{V^2}}$$

- **Numerical values (unweighted):**
  - S < 0.05: Smooth powder pattern
  - 0.05 < S < 0.15: Mild spottiness / texture
  - S > 0.15 : strongly spotty, likely with large grain size
- **Alternative:**
  - Weighted average by ring intensity

$$S_I = \sqrt{\frac{\sum_{rings} I \frac{VV}{V^2}}{\sum_{rings} I}}$$

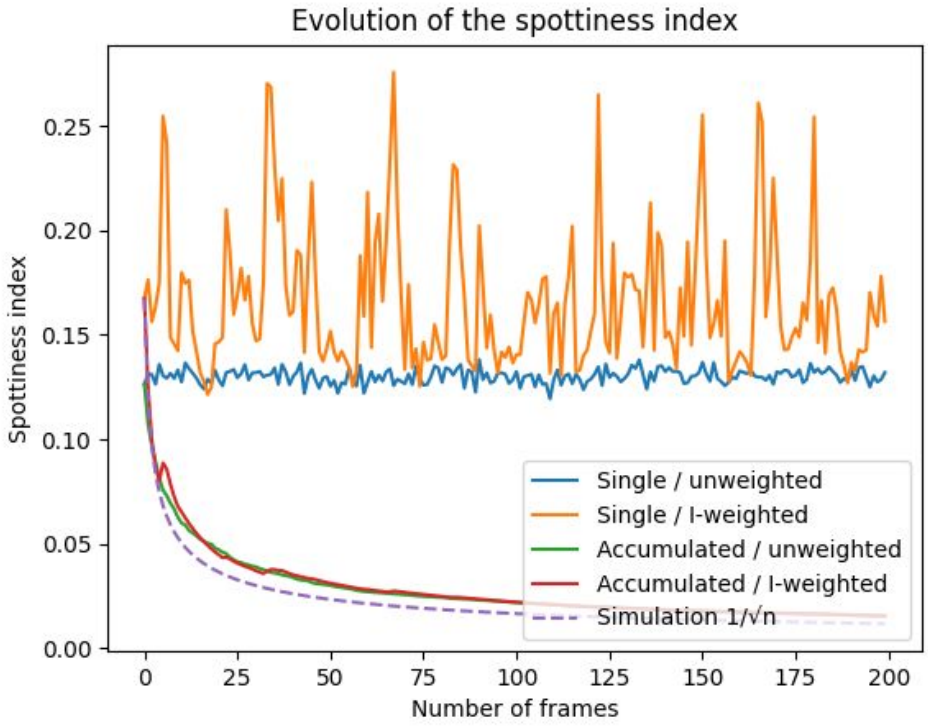
- **Availability:**
  - `res.calc_spottiness(weighted=True/False)`

Exercise:

- **Calculate the spottiness index for:**
  - The first frame
  - The averaged frame (200 frames)
- **Consider the weighting by the scattering intensity**
- **Hint: look at the documentation**
  - https://www.silx.org/doc/pyFAI/latest/api/pyFAI.html#pyFAI.containers.Integrate1dResult.calc_spottiness
- **Result**
  - 0.16 and 0.2 for the first
  - 0.03 and 0.06 for the average

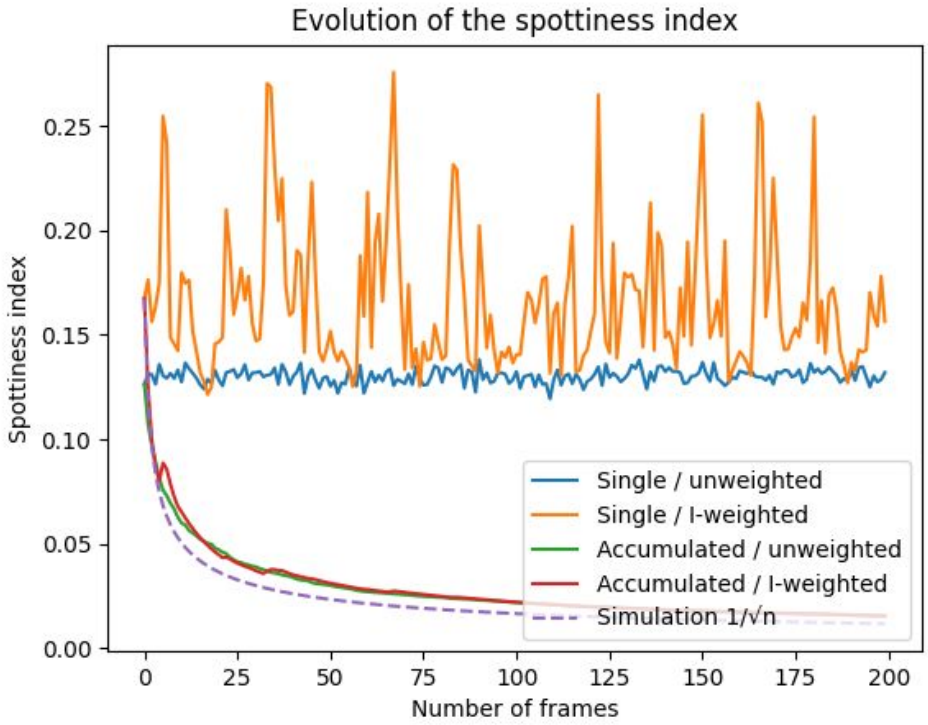
Accumulation of integrated data

- **Did you know one can further process integrated data:**
  - Rebin data from a finer grid to a coarser one
    - **Example rebin 2D integrated data to 1D**
  - Merge several datasets (using the `union` operator)
    - **Example follow online the reduction of the spottiness**



Exercise:

- **Perform the azimuthal integration of each of the 200 frames and store the result**
- **Accumulate the result of the two first frames**
- **Build the plot of the former slide:**



Thanks for your attention