

# Using pyFAI-calib2 GUI for area-detector calibration

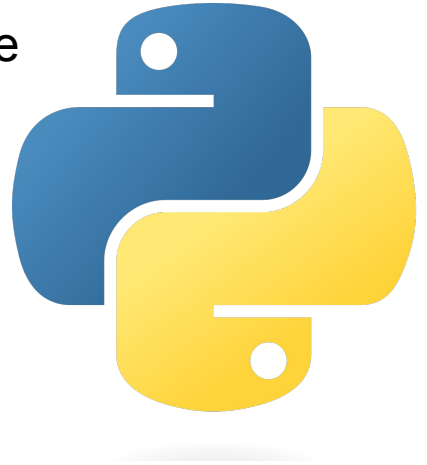
By Kevin Morell and Peter Meshkov

# What is pyFAI?

- pyFAI is a Python library which helps convert XRD detector images to an Intensity vs 2Theta CSV file
- You will need to have Python installed on your machine
  - <https://www.python.org/downloads/>
- You will also need the pyFAI package installed in a Python environment
  - <https://pypi.org/project/pyFAI/>
- Otherwise, no other installation is necessary

# Installing Python

- Python installation is different based on your OS
- Windows: go to the Microsoft Store and find the most recent version of python. It will install to a hidden folder and again will not have to be used beyond setup wizard.
- MacOS: download from python site and follow setup guide  
<https://www.python.org/downloads/macos/>
- Linux: run “sudo apt install python3” in terminal



# Conda

- Package manager for Python
  - Anaconda is heavier version, with almost every package you could possibly need preinstalled
  - Miniconda is lighter
  - Neither version comes with pyFAI
- Download for any OS here: <https://www.anaconda.com/download/success>
- Follow setup guide
  - It may ask if you want to open terminal in your python environment by default; select yes



# Your Python Environment

- Activate your Python environment in terminal
  - This is easy to do with conda, just enter the command “conda activate” in terminal
  - To deactivate, enter “conda deactivate”
  - If you are using Python venv instead of conda, follow these instructions  
<https://docs.python.org/3/library/venv.html> (I recommend using conda)
- You should have (base) in front of your username if the environment is active
  - If using Python virtual environment, it may be (venv) or something else
- Now you can call system level Python commands in terminal (ex: “pip install”)

```
petermeshkov@130-199-210-250 ~ %  
petermeshkov@130-199-210-250 ~ % conda activate  
(base) petermeshkov@130-199-210-250 ~ %
```

# pip Installer



- pip is a tool that installs python packages
  - Python versions  $\geq 3.4$  come with pip preinstalled.
- Ensure pip is installed with the command “pip --version”

```
(base) petermeshkov@130-199-210-250 ~ % pip --version
pip 23.3.1 from /opt/anaconda3/lib/python3.11/site-packages/pip (python 3.11)
```

- If it is missing, try the following commands in terminal:
  - Windows: “py -m ensurepip --upgrade”
  - MacOS: “python -m ensurepip --upgrade”
  - Linux: “python -m ensurepip --upgrade”
- Called through terminal
  - Can be called in Python shell (in a Python program) by including an exclamation mark before the command
- Conda has an equivalent command “conda install”



# Install pyFAI

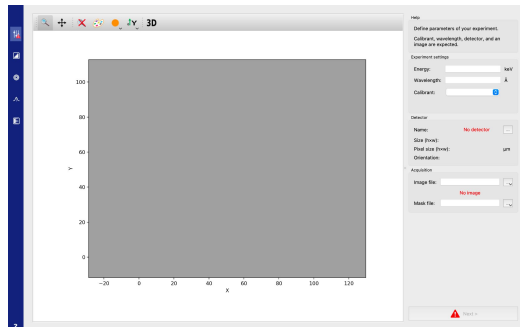
```
>S C:\Users\Kev> pip install pyFAI
```

```
Requirement already satisfied: pyFAI in c:\users\kev\anaconda3\lib\site-packages (2024.5.0)
Requirement already satisfied: numpy>=1.10 in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (1.26.4)
Requirement already satisfied: h5py in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (3.11.0)
Requirement already satisfied: fabio in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (2024.4.0)
Requirement already satisfied: silx>=2 in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (2.1.0)
Requirement already satisfied: numexpr!=2.8.6 in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (2.8.7)
Requirement already satisfied: scipy in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (1.13.1)
Requirement already satisfied: matplotlib in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (3.8.4)
Requirement already satisfied: packaging in c:\users\kev\anaconda3\lib\site-packages (from pyFAI) (23.2)
Requirement already satisfied: hdf5plugin in c:\users\kev\anaconda3\lib\site-packages (from fabio->pyFAI) (4.4.0)
Requirement already satisfied: lxml in c:\users\kev\anaconda3\lib\site-packages (from fabio->pyFAI) (5.2.1)
Requirement already satisfied: pillow in c:\users\kev\anaconda3\lib\site-packages (from fabio->pyFAI) (10.3.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\kev\anaconda3\lib\site-packages (from matplotlib->pyFAI) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\kev\anaconda3\lib\site-packages (from matplotlib->pyFAI) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kev\anaconda3\lib\site-packages (from matplotlib->pyFAI) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\kev\anaconda3\lib\site-packages (from matplotlib->pyFAI) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\kev\anaconda3\lib\site-packages (from matplotlib->pyFAI) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kev\anaconda3\lib\site-packages (from matplotlib->pyFAI) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\kev\anaconda3\lib\site-packages (from python-dateutil->matplotlib->pyFAI) (1.16.0)
```

- pip should automatically install all of the required modules for pyFAI

# Install pyFAI cont.

- If “pip install pyFAI” doesn’t work, ensure your terminal is opened in the Python environment
  - I.e. you see (base) or (venv) or some custom made environment name in front of your name
- Now you can run the pyFAI graphical user interface using the command “pyFAI-calib2” in terminal; the interface shown below should pop up
  - Your directory location in terminal doesn’t matter, the GUI allows you to select files from anywhere





# PONI File

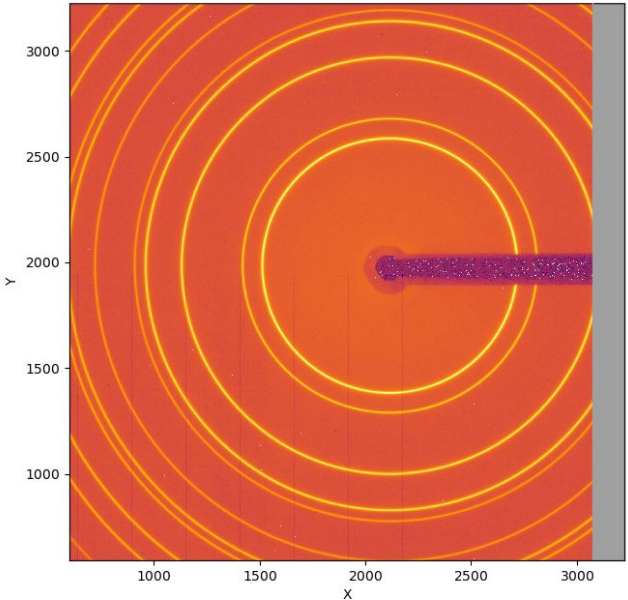
In order to create the calibration (PONI) file you need a few key things:

- The data .tiff file (detector image) saved locally
- The wavelength or energy your data was taken with
- The name of the detector used to collect your data
- The species of the sample or a .cif file for that species (if it is not present in the drop down in pyFAI)

# Calibration data entry

PyFAI Calibration

3D



Help

Define parameters of your experiment.  
Calibrant, wavelength, detector, and an image are expected.

Experiment settings

Energy: 64.67615985039137 keV

Wavelength: 0.1917 Å

Calibrant: CeO2

Detector

Name: Dexela 2923

Size (h×w): 3888 × 3072 px

Pixel size (h×w): 75.0 × 75.0 μm

Orientation: BottomRight (3)

Acquisition

Image file: 723\_detector/CeO2.tiff

Image size: 3888 × 3072 px

Binning: 1 × 1

Mask file:

Dark file:

Flat-field file:

Next >

Enter wavelength or energy here

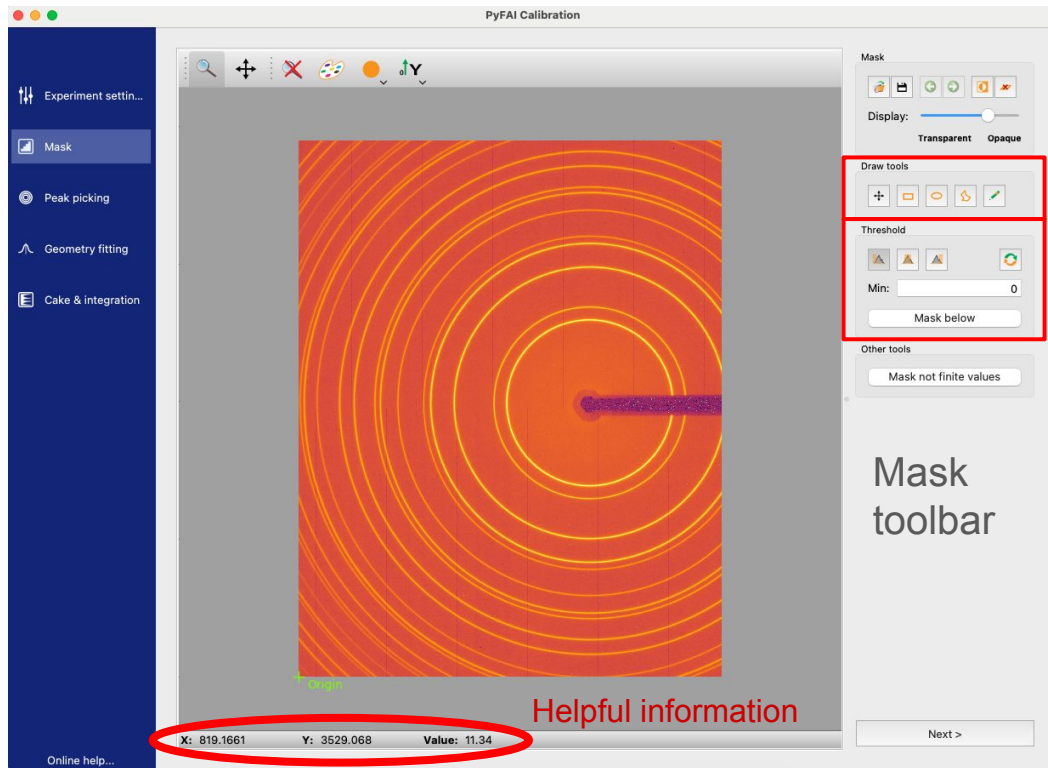
Enter calibrant species or .cif here

Enter detector here

Enter image file here

Once all fields are filled, press next

# Mask intro



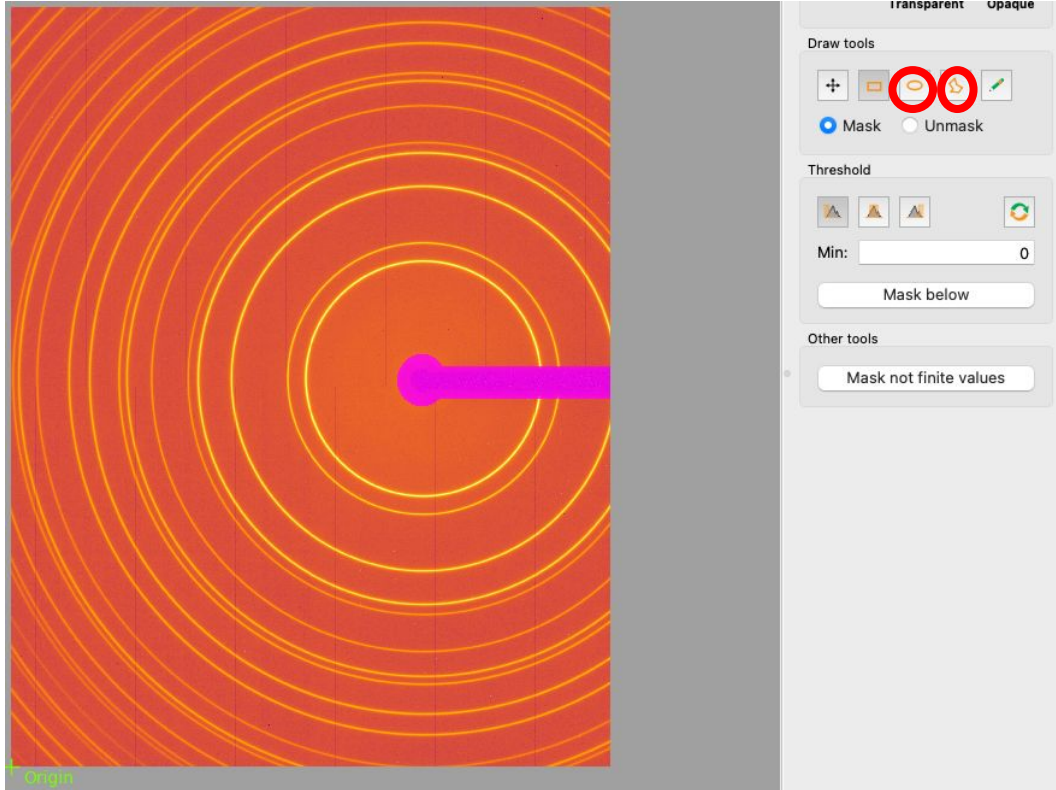
Masks are files that filter out unwanted data from your detector readings

For example, in the figure on the left, you can see where our beam stop and arm were blocking light

However, some imperfections like dead pixels are harder to spot

Calib2 mask provides a few tools for these situations

# Drawing a mask



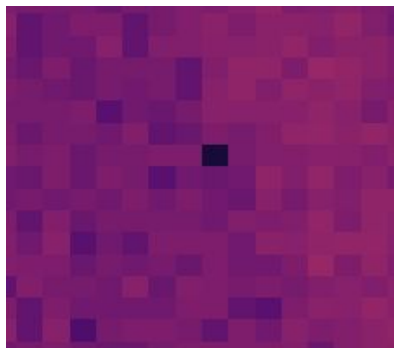
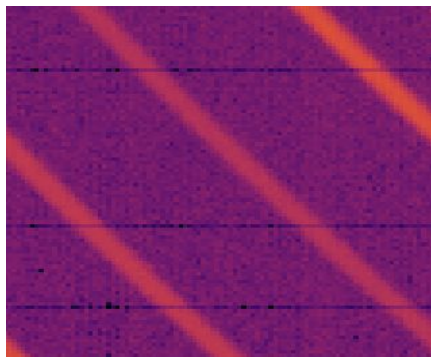
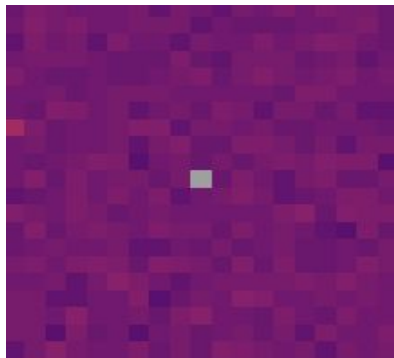
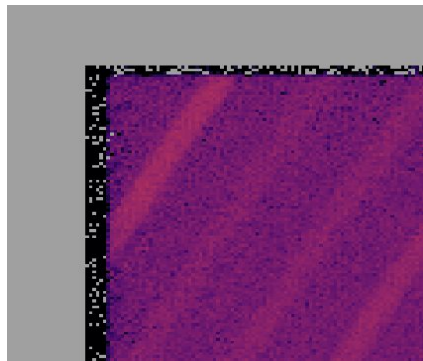
First we handle the beamstop

The two circled Draw Tools are the most helpful, one draws ellipses and the other polygons

Use the ellipse tool to circle the beamstop and the polygon for the arm.

Once selected, the masked areas should be pink as shown.

# Pixel problems



Next we mask irregular pixels

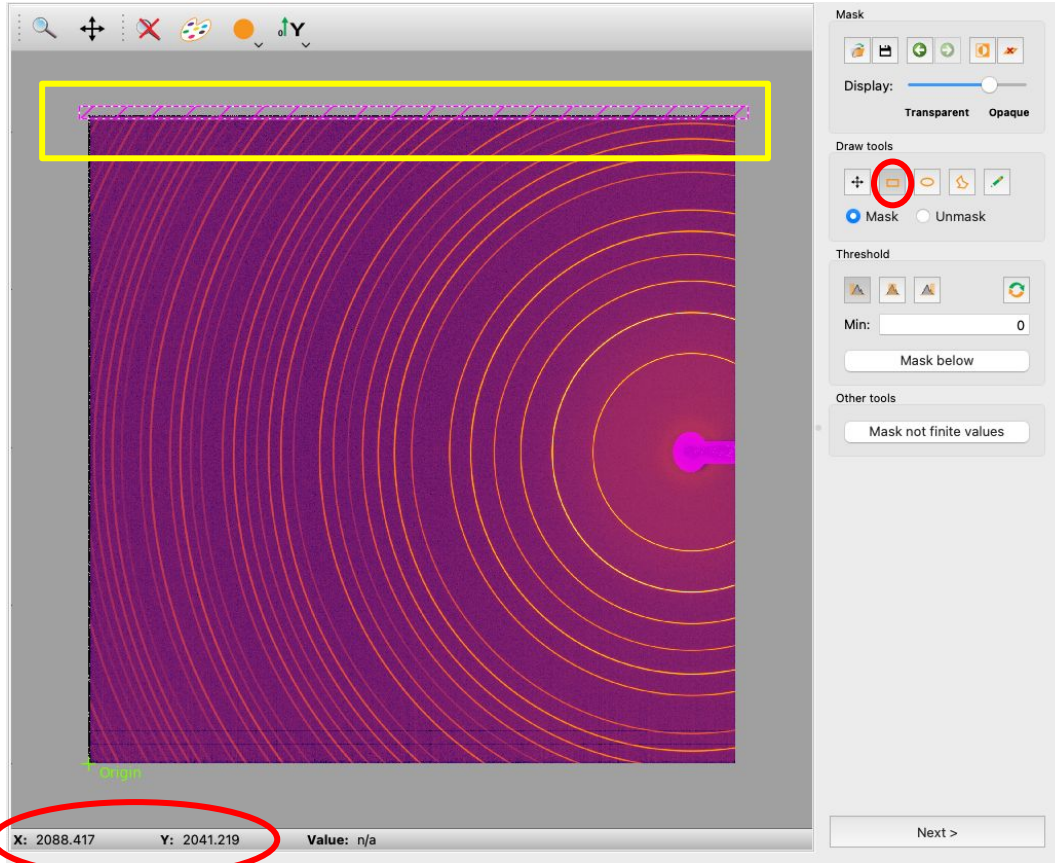
Depending on the condition of your detector, you may experience different irregularities.

As as seen in the top two images, border defects and negative values are easy to handle with `calib2`.

As seen in the bottom two, defective lines and non-negative outliers are better handled by the median filter described in notebook 003.

(note: this is a new example file, as the other one didn't have many irregularities)

# Masking edges

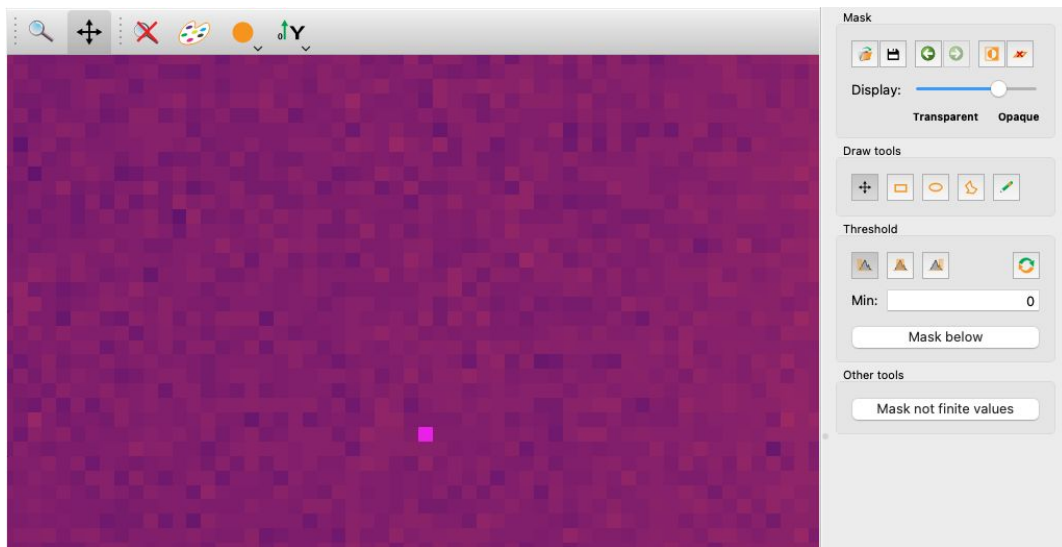


For irregularities on the boundary:

Use the rectangle tool to mask the defective section. The coordinates on the bottom may be helpful as you need to be completely zoomed out to mask the entire edge.



# Masking negatives



For the negative pixel case:

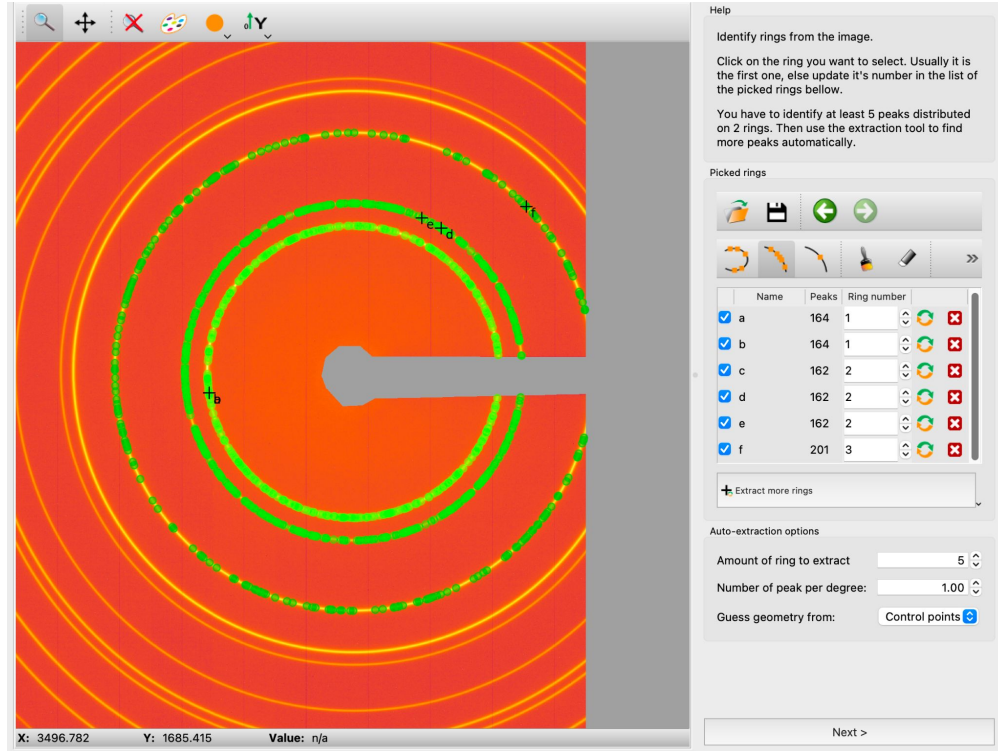
Under Threshold select the left “Mask... below given threshold” option.

Enter a minimum of zero.

Press Mask below.

Once you have created your mask, press next.

# Ring Selection for Calibration: Manual



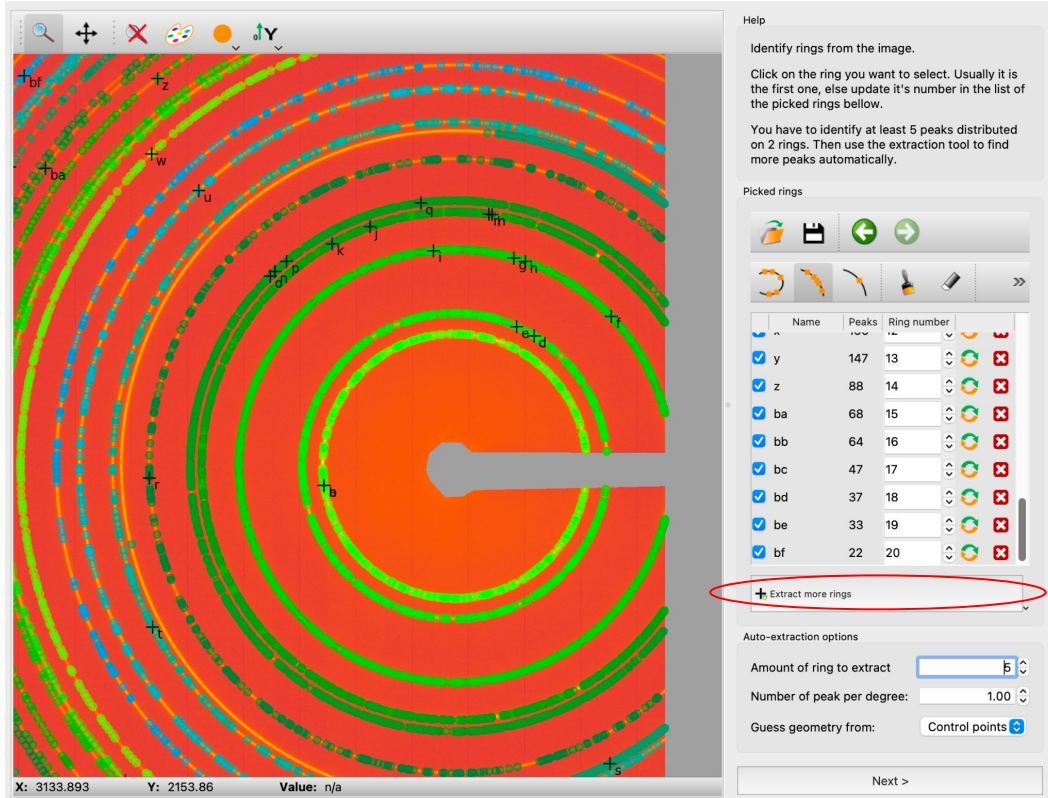
Once masking is complete, start the ring selection process. Choose the number of rings to extract (5 is usually fine), and left click on a ring to select.

A table of labeled rings should appear; make sure the ring number matches the number you selected.

Try to select each ring multiple times.

(We are back to the original example file)

# Ring Selection for Calibration: Automated

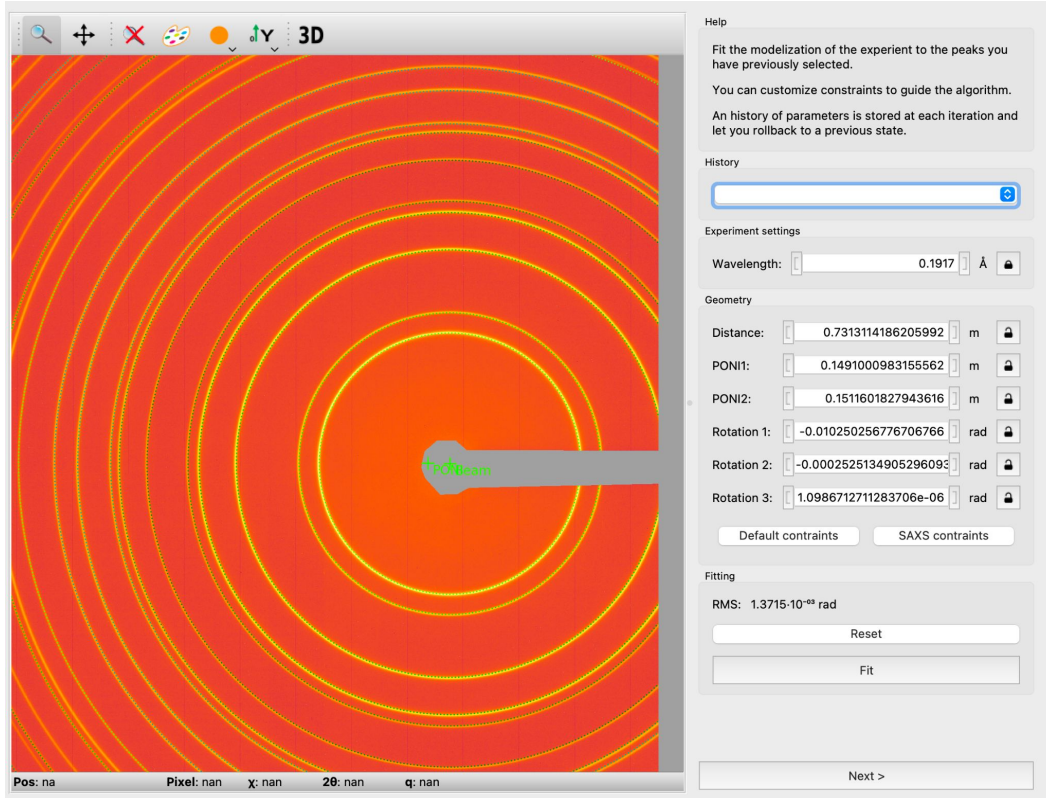


After 5 manual rings, the “Extract more rings” feature should be able to extract the rest of the rings for you. Ratatouille!

Make sure to double check the rings it extracts, and also if it seems necessary, select some of the outer rings multiple times.

Hit next to continue to calibration.

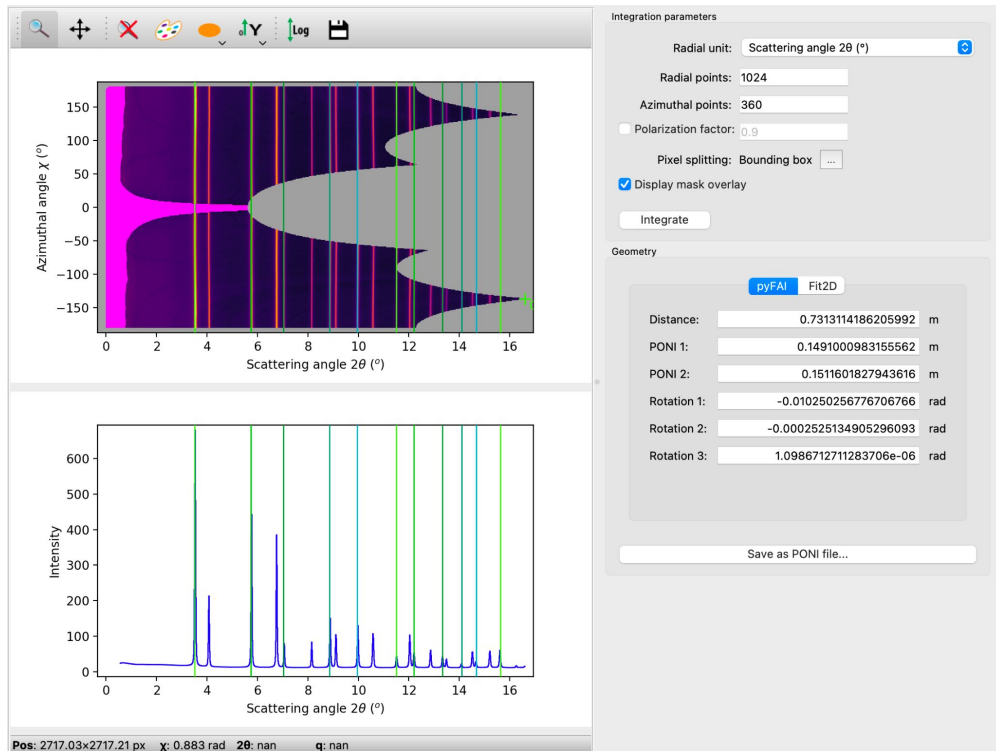
# Automatic Calibration



The PONI (point of normal incidence) and the Beam location, should be automatically located based on the rings, as well as some additional parameters.

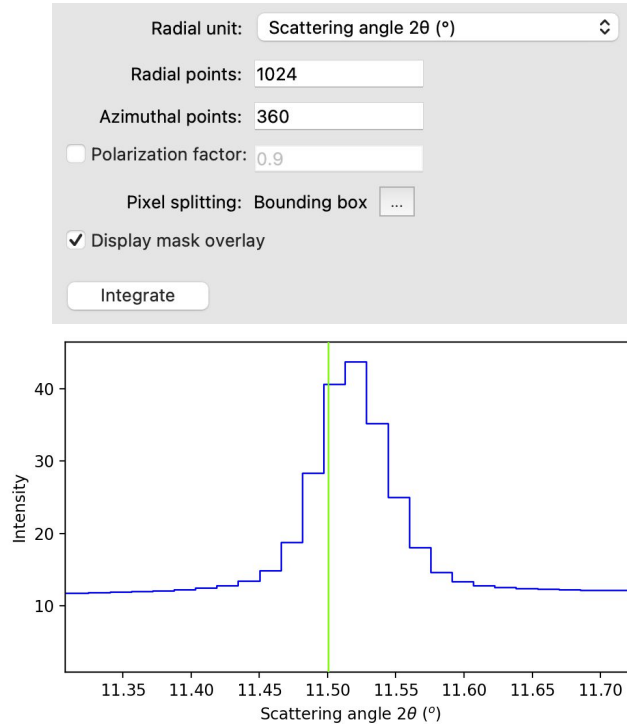
Now we are ready to integrate.

# Integration



Once you continue from the calibration, the 2D cake pattern, and the azimuthal integration of the intensity should be automatically created.

# Integration: Parameter Adjustment




The graphical cells are interactive; zoom in by selecting a rectangular region on either plot (to exit zoom, press the looking glass with the red X over it).

Upon zooming one of the further peaks, we see that it is very jagged; we can deal with this by increasing the number of radial points (essentially increasing the bin count).



# Integration: Parameter Adjustment

Radial unit: Scattering angle  $2\theta$  (°) 

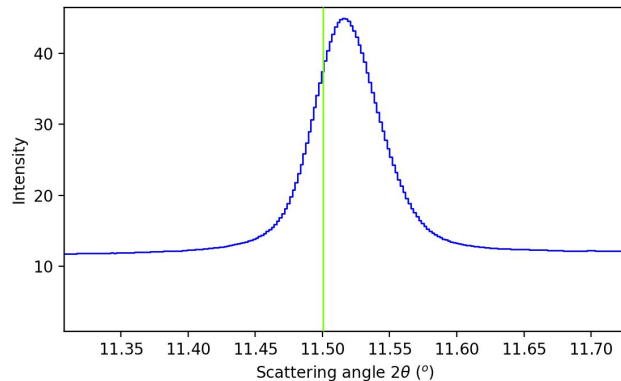
Radial points:

Azimuthal points:

☐ Polarization factor:

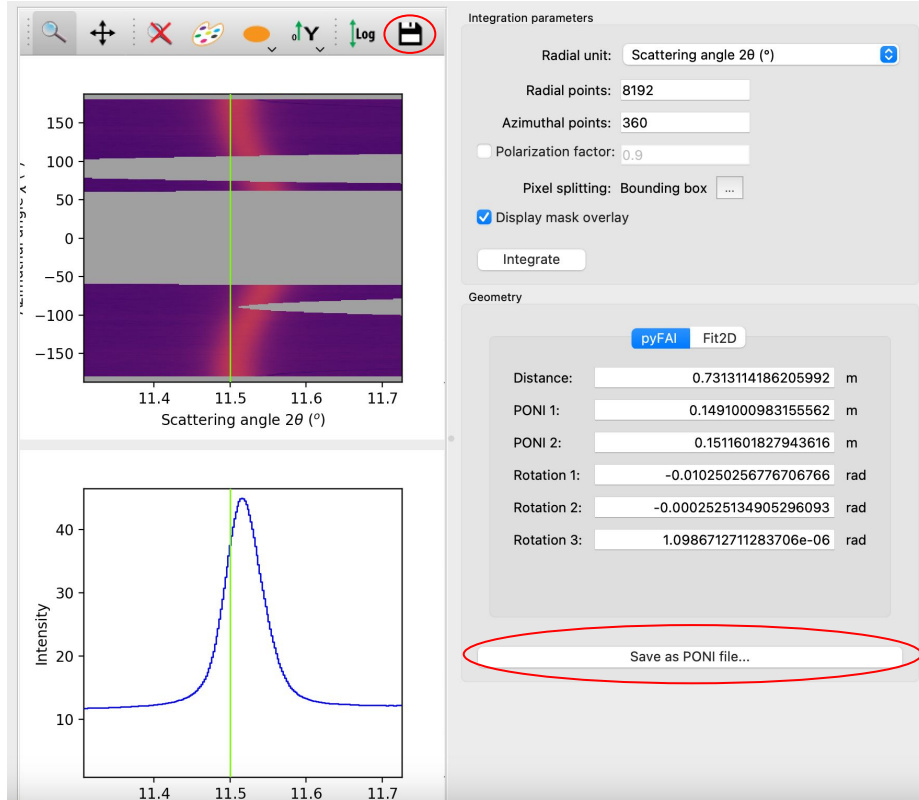
Pixel splitting: Bounding box

☒ Display mask overlay



Increasing from 1024 by a factor of 8 yields a much nicer peak with little computational increase!

# Integration: Getting your Result



Select the Save icon at the the top to save Intensity vs 2Theta data as a CSV.

Select “Save as PONI file...” to save your calibration as a PONI file.

Later, if you prefer to redo integration with the pyFAI python tool instead of the GUI, you can import this PONI file.

**(\*) The peak in this page is a bit off. This will be addressed later.**

# Riding your PONI into Colab

```
[21] 1 # Create a azimuthal integrator (without fitting capabilities from the geometry-refinement object)
      2 ai_recalibrated = pyFAI.load(gr)
      3 ai_recalibrated
```



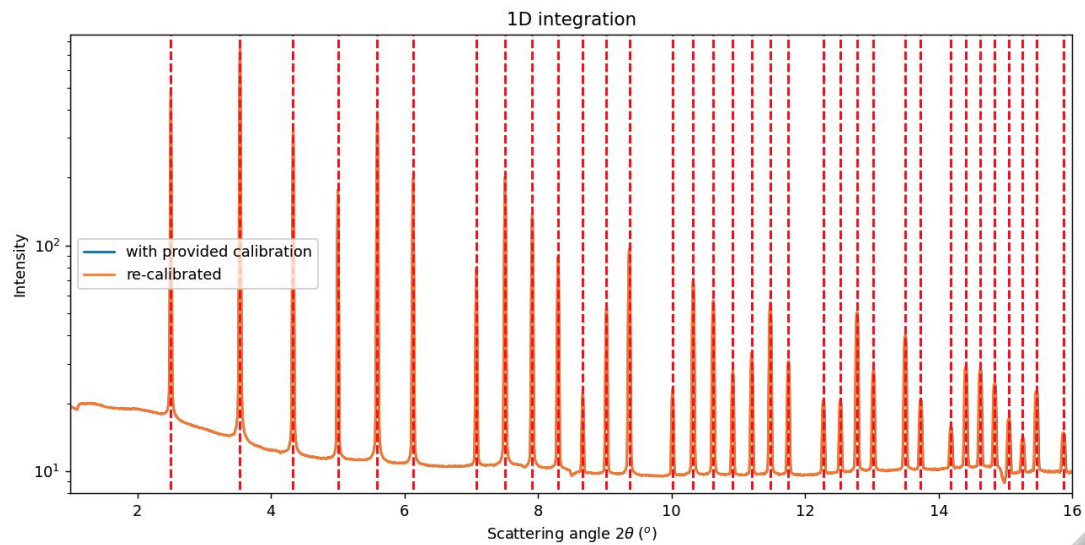
```
[21] 1 # Create a azimuthal integrator (without fitting capabilities from the geometry-refinement object)
      2 ai_recalibrated = pyFAI.load('from_PerkinElmer_detector/calibtest.poni')
      3 ai_recalibrated
```

```
Detector Perkin detector      PixelSize= 2.000e-04, 2.000e-04 m
Wavelength= 1.814000e-11 m
SampleDetDist= 1.427826e+00 m  PONI= 2.343461e-01, 3.799538e-01 m    rot1=-0.001487  rot2=-0.026384  rot3=-0.000135 rad
DirectBeamDist= 1428.325 mm    Center: x=1910.382, y=983.329 pix    Tilt= 1.514° tiltPlanRotation= -86.776° λ= 0.181Å
```

In the pySULI folder on Colab, upload your PONI file somewhere notebook 003 can easily access it.

Edit the cell on the left so that ai\_recalibrated loads from YOUR PONI file, and not the one created earlier in the notebook.

PONI can be read with  
pyFAI.load()



The output shows our calibration was successful.