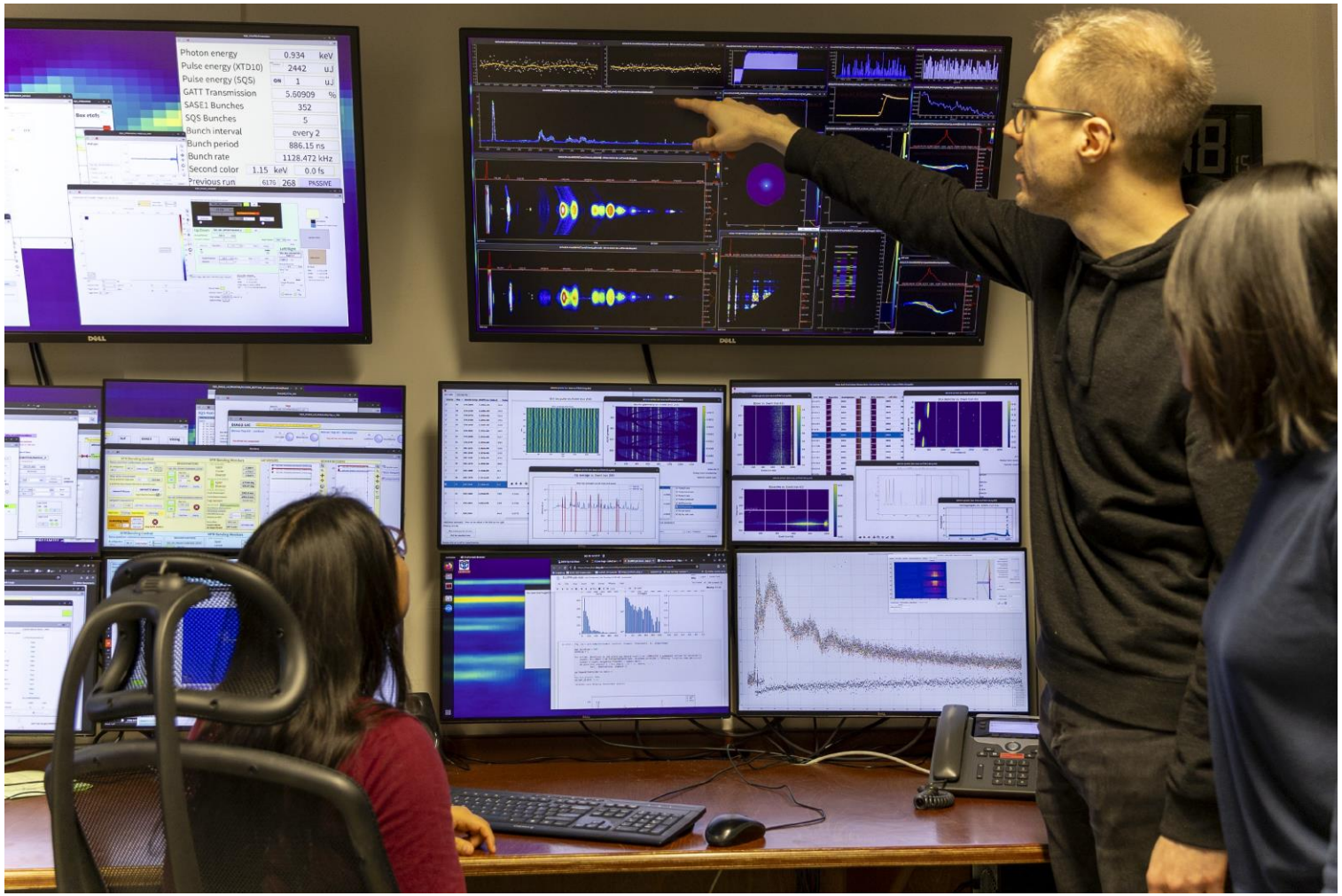
Introduction to Online Analysis

Cammille Carinan
David Hammer
Data Analysis Group

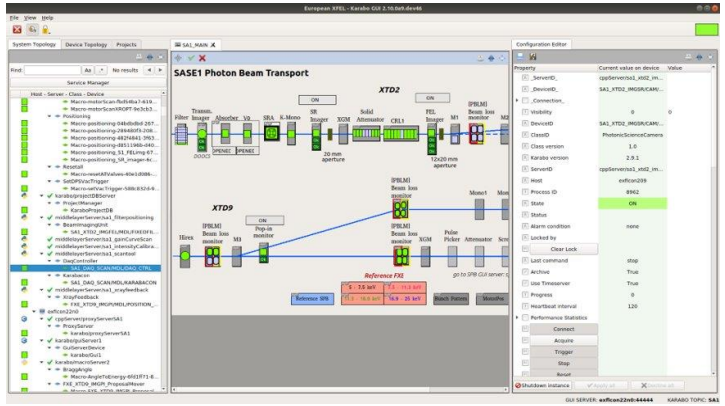
European XFEL Users' Meeting 2024
26 January 2024

Online analysis is data analysis that is performed in near-realtime, to guide an experiment.



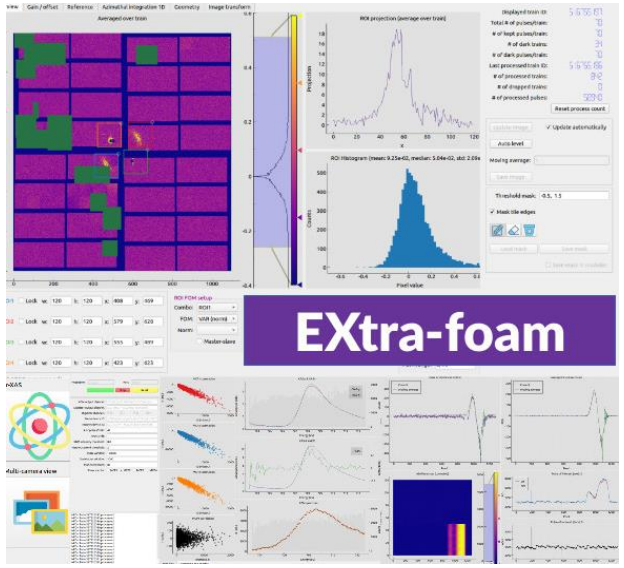
There are currently four ways to do online analysis at the facility.

Karabo devices



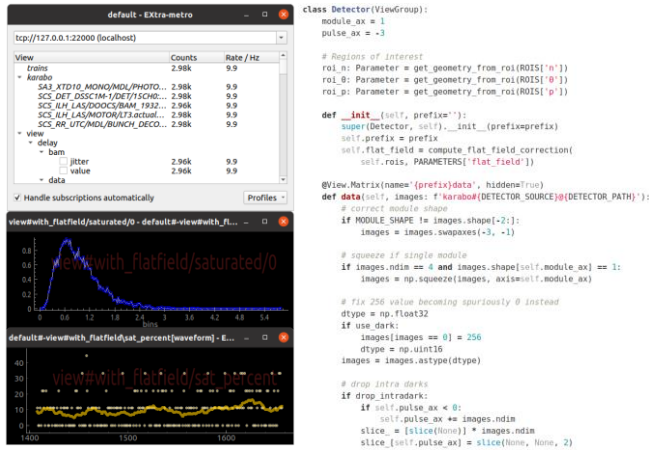
https://rtd.xfel.eu/docs/karabo/en/latest/index.html

EXtra-foam



https://extra-foam.readthedocs.io

EXtra-metro

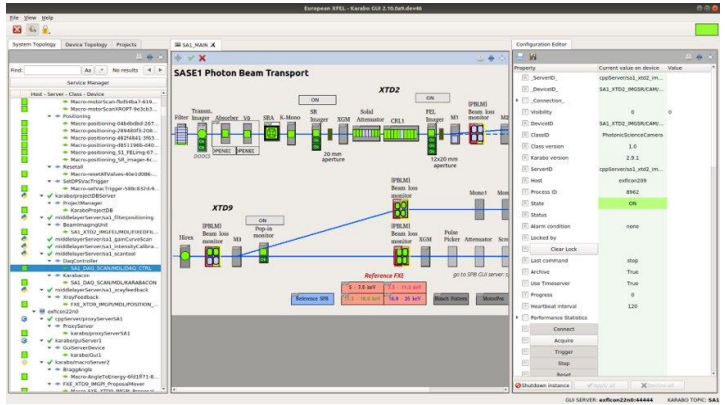


https://rtd.xfel.eu/docs/metropc/en/latest/index.html

Custom tools

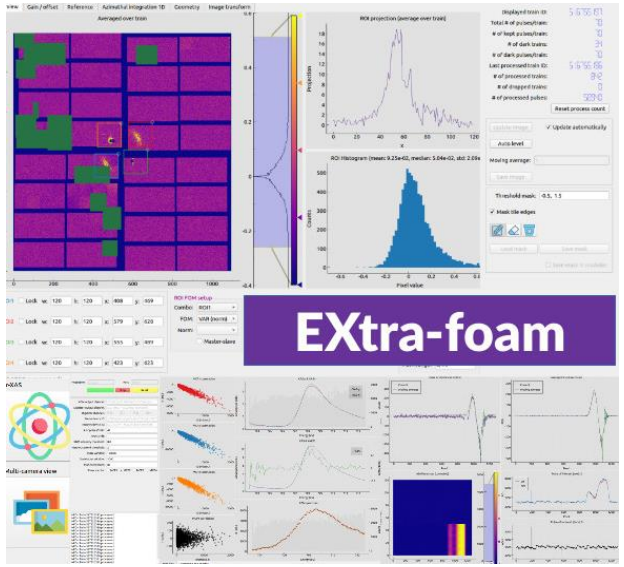
Further questions and requirements for your upcoming beamtime? Let us help you!
Contact us at da@xfel.eu
Or visit https://www.xfel.eu/data_analysis for more information.

Karabo devices



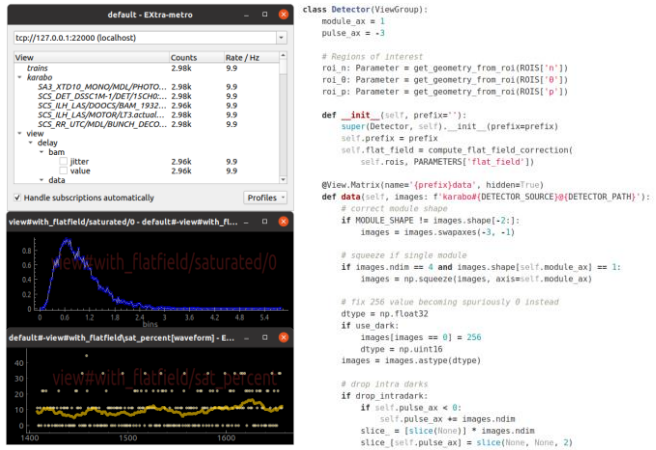
https://rtd.xfel.eu/docs/karabo/en/latest/index.html

EXtra-foam



https://extra-foam.readthedocs.io

EXtra-metro



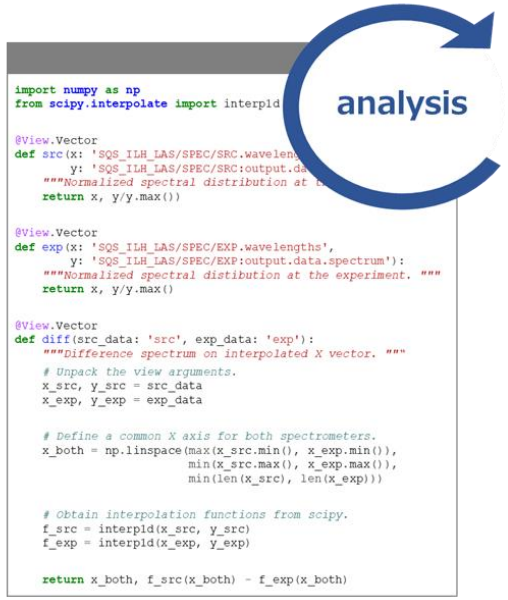
https://rtd.xfel.eu/docs/metropc/en/latest/index.html

Custom tools

Karabo: the distributed control system is composed of small pluggable units (devices) that represent various components

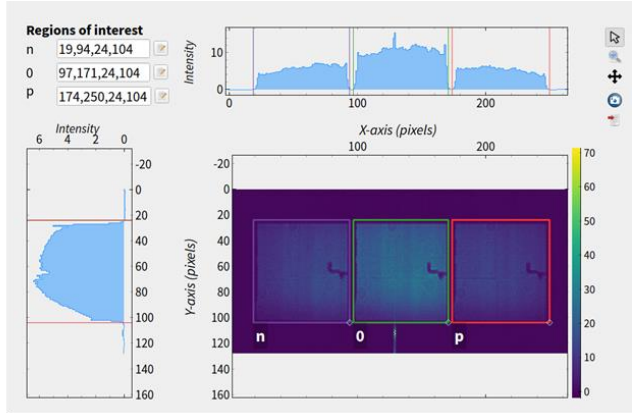
detector

(calibrated) big data

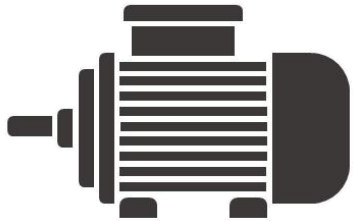


analysis device

parameters



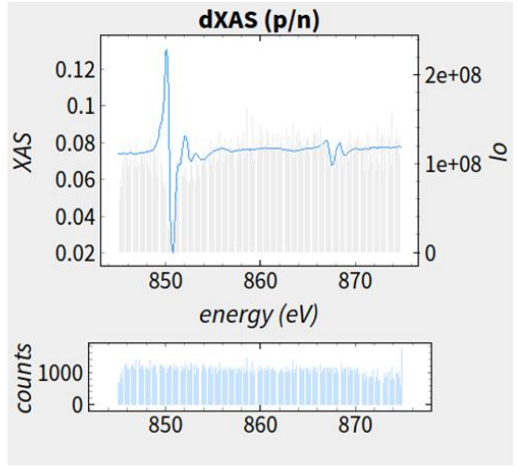
output with user input



scanning device

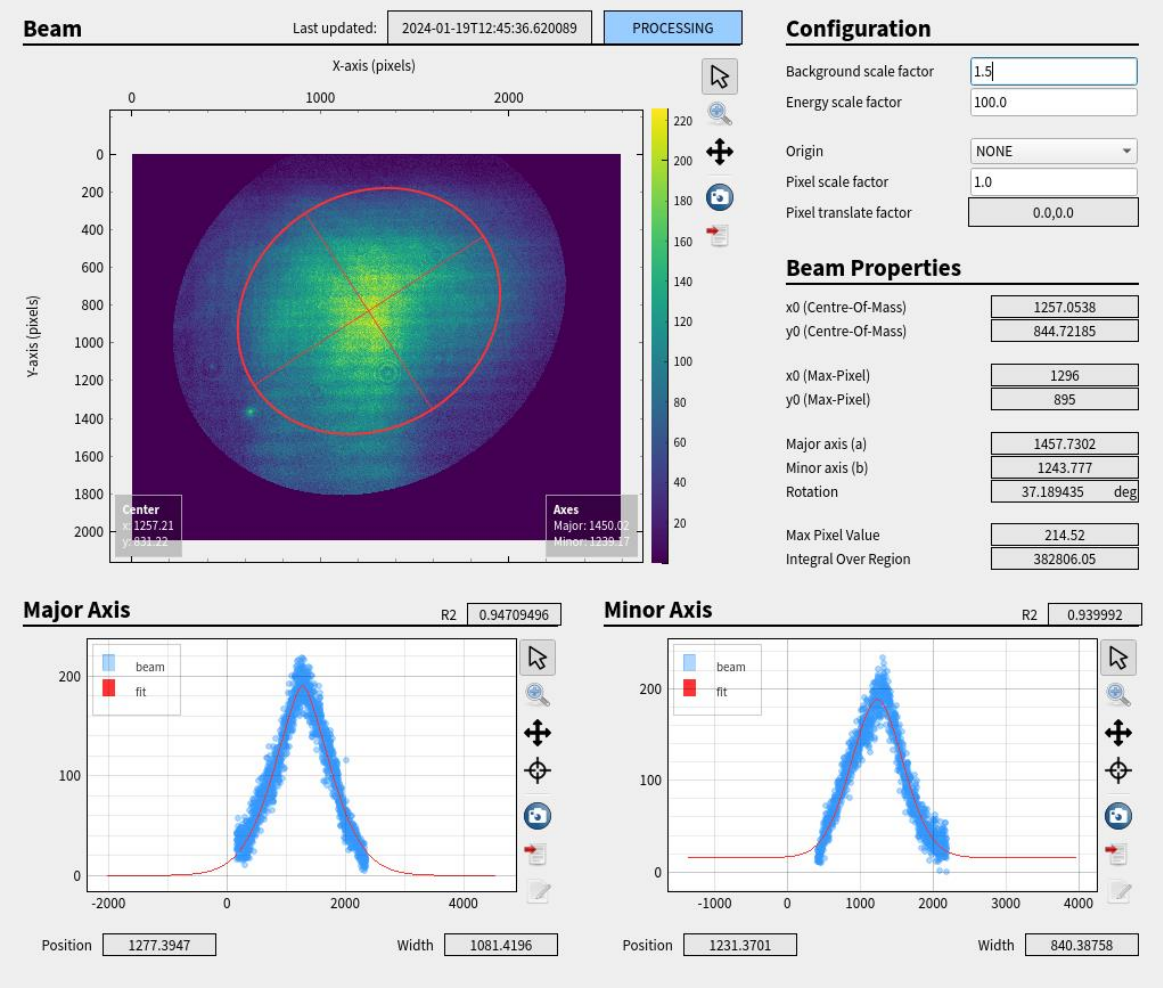
control data

settings change

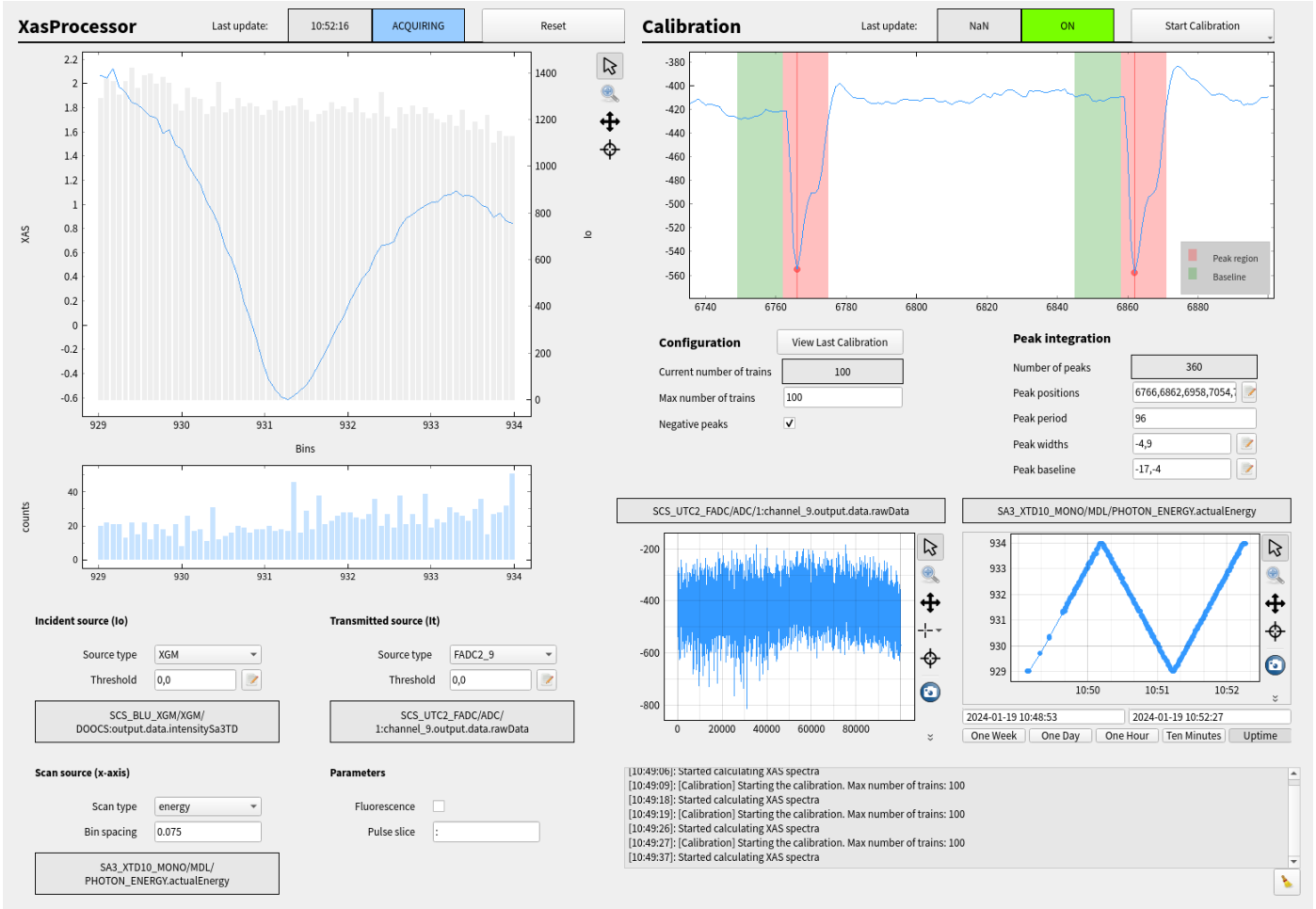


visualization of resulting data

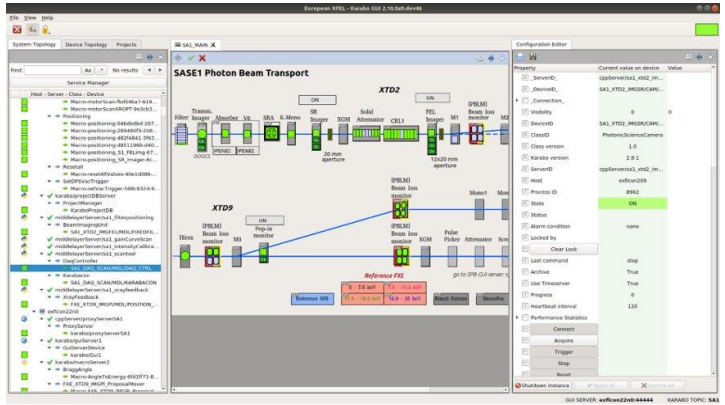
BeamProcessor device: the quality of a laser beam is characterized by calculating its properties such as its centroid and widths



XasProcessor device: the binned X-ray absorption spectrum of a digitizer is calculated by integrating its peaks and normalizing with an XGM

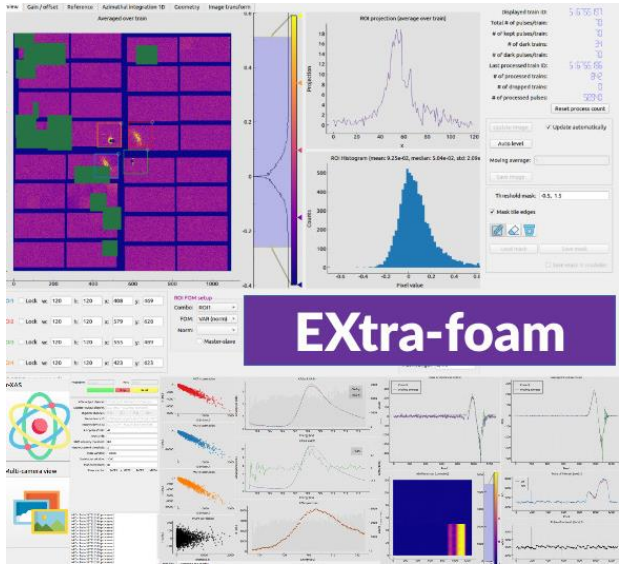


Karabo devices



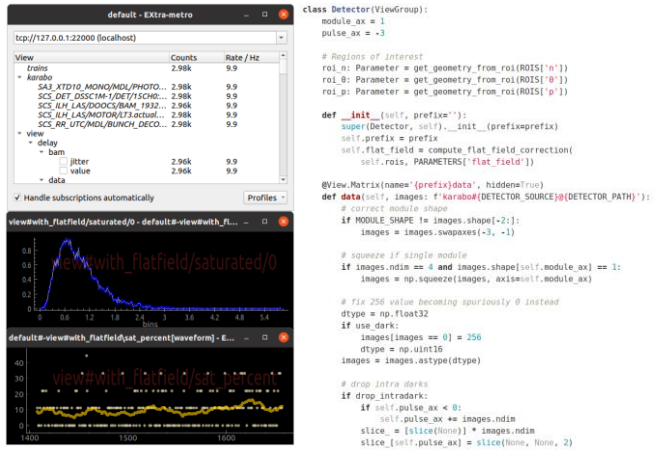
https://rtd.xfel.eu/docs/karabo/en/latest/index.html

EXtra-foam



https://extra-foam.readthedocs.io

EXtra-metro

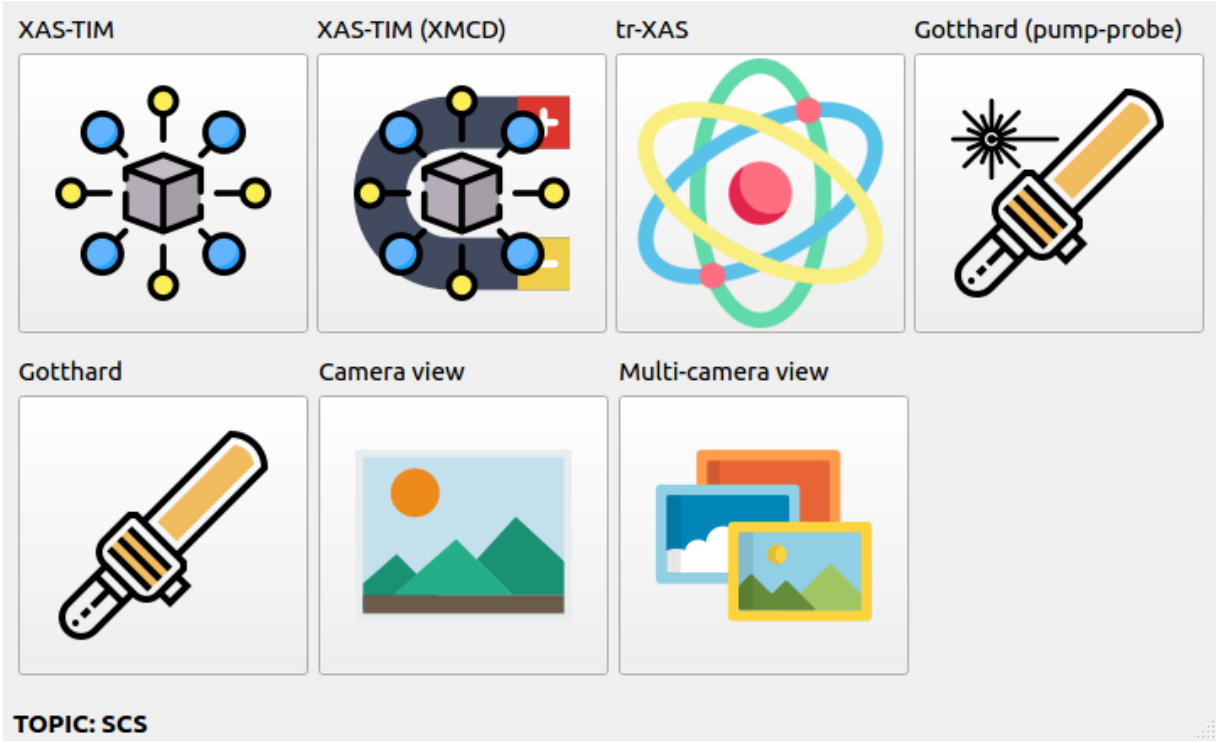


https://rtd.xfel.eu/docs/metropc/en/latest/index.html

Custom tools

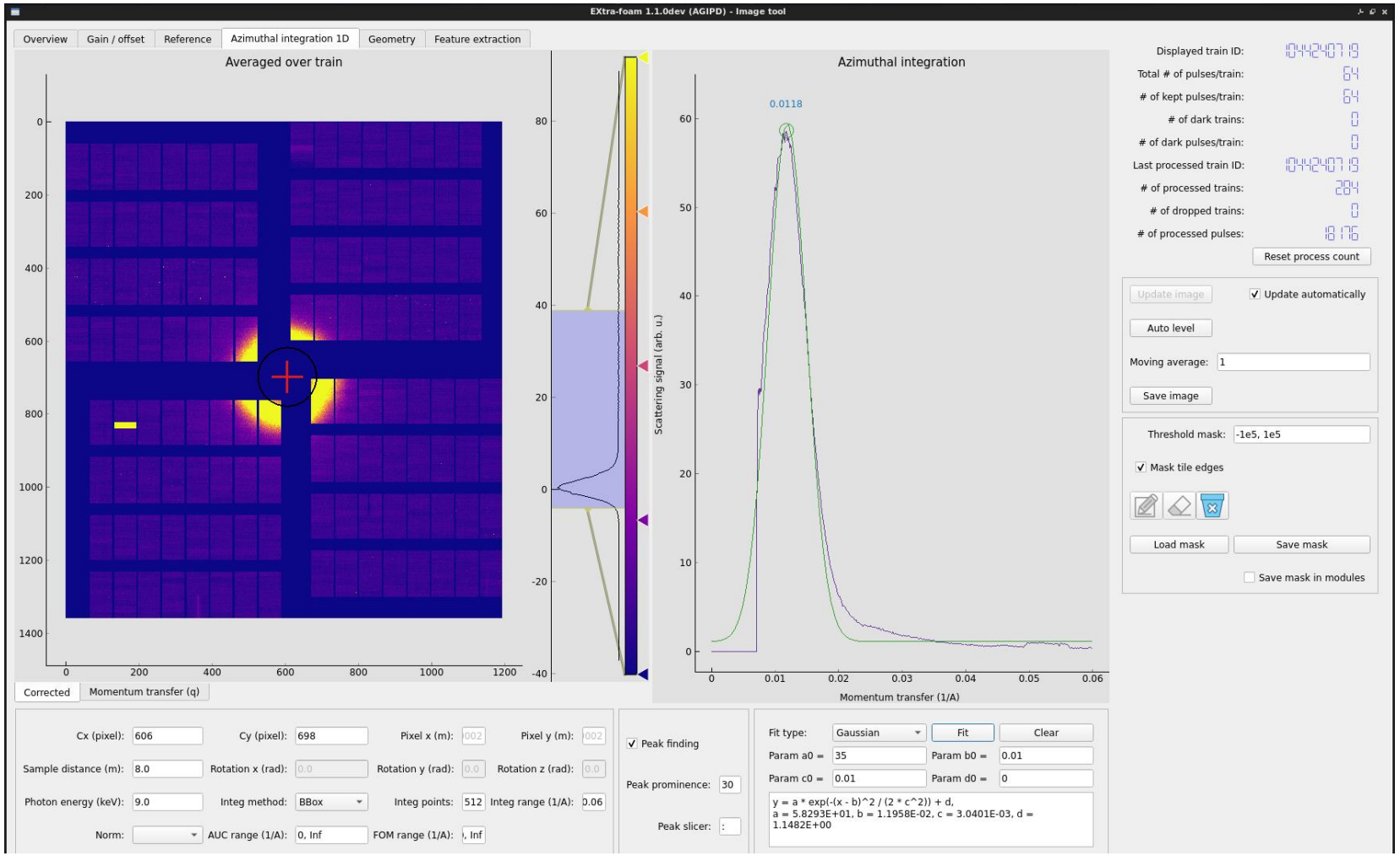
EXtra-foam: an in-house developed framework that processes and visualizes data from experiments that uses 1D and 2D detectors

- **F**ast **O**nline **A**nalysis **M**onitor
- Graphical program for online analysis
- Supports features such as:
  - Detector preview from user-specified geometry (e.g. CrystFEL geometry file)
  - ROI analysis
  - Azimuthal integration
  - Correlating figures-of-merit with other data
  - Masking

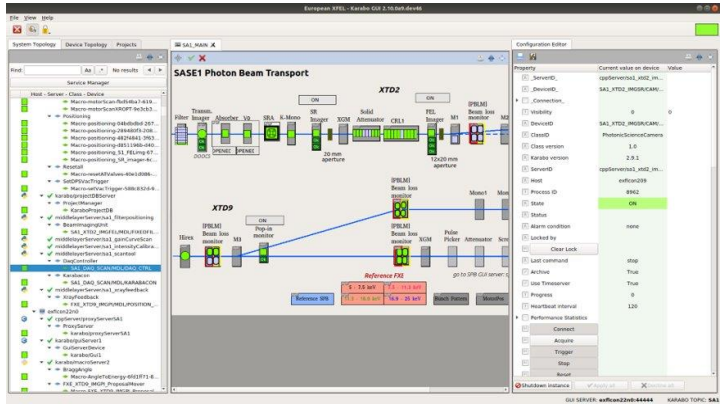


Special suites are small applications dedicated to specific online analysis.

EXtra-foam: analysis procedures for experiments such as azimuthal integration from a scattering experiment using AGIPD are provided

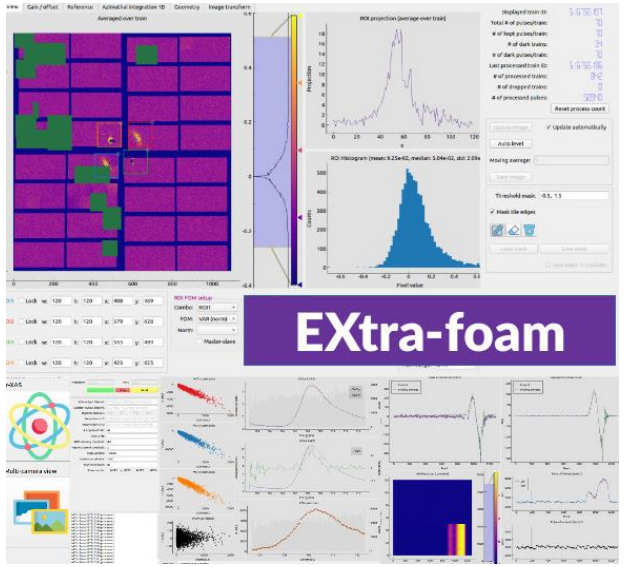


Karabo devices



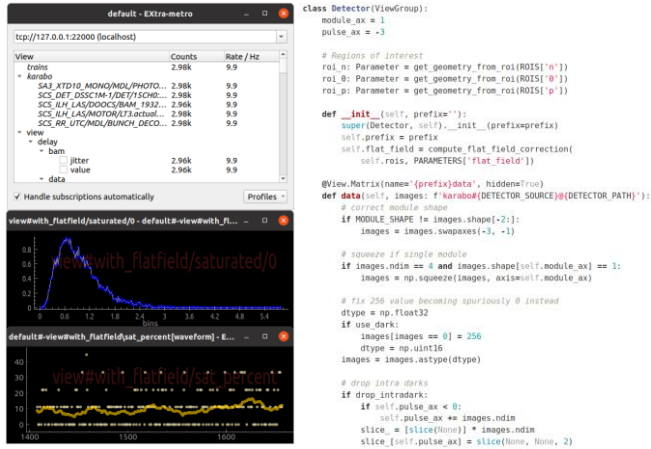
https://rtd.xfel.eu/docs/karabo/en/latest/index.html

EXtra-foam



https://extra-foam.readthedocs.io

EXtra-metro



https://rtd.xfel.eu/docs/metropc/en/latest/index.html

Custom tools

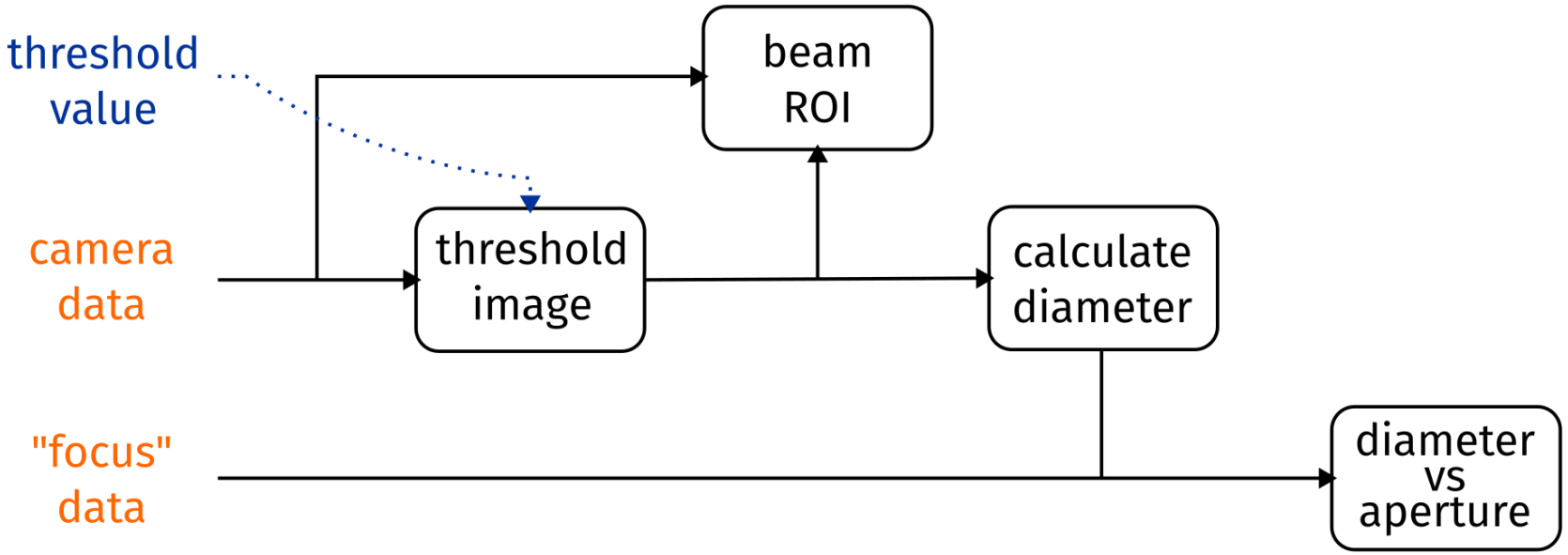
EXtra-metro

- Express analysis as small connected steps
- Each step is a simple Python function
- Code can be changed at runtime
- Allows exploratory data analysis with live data

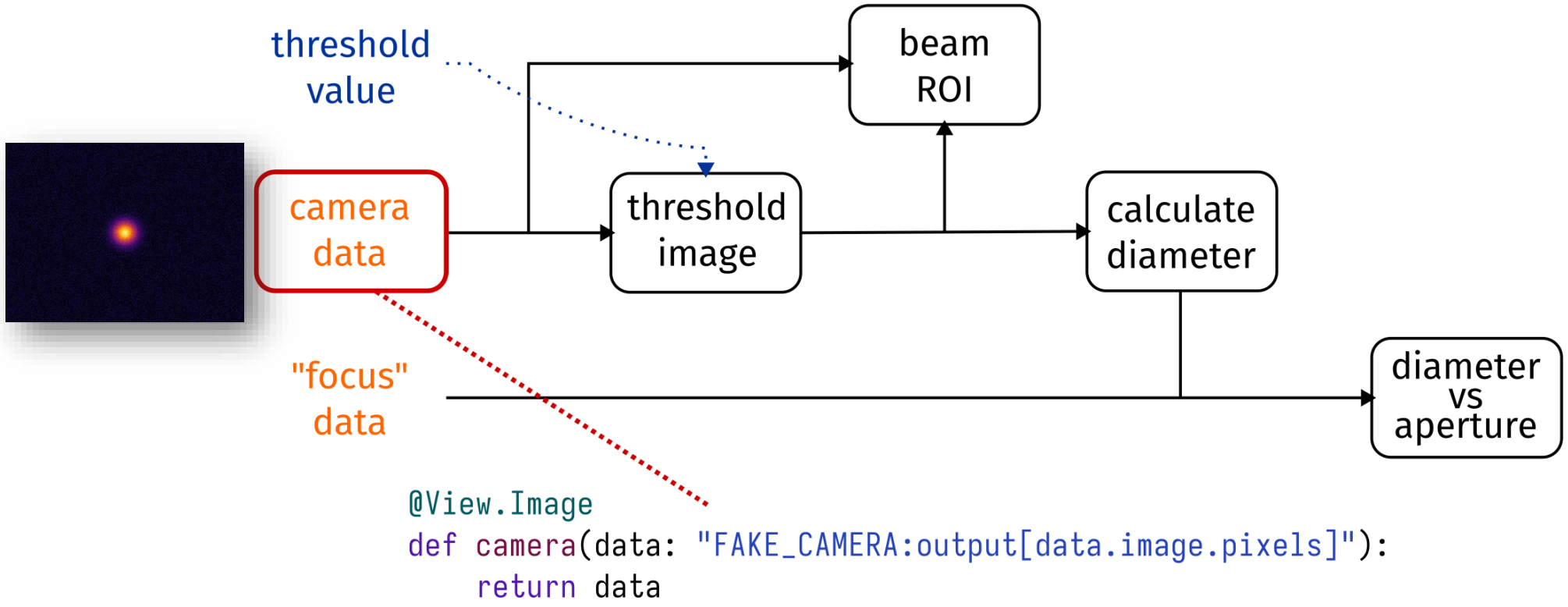
The framework handles:

- Getting data from the control system
- Executing your analysis
- Visualizing (intermediate) results

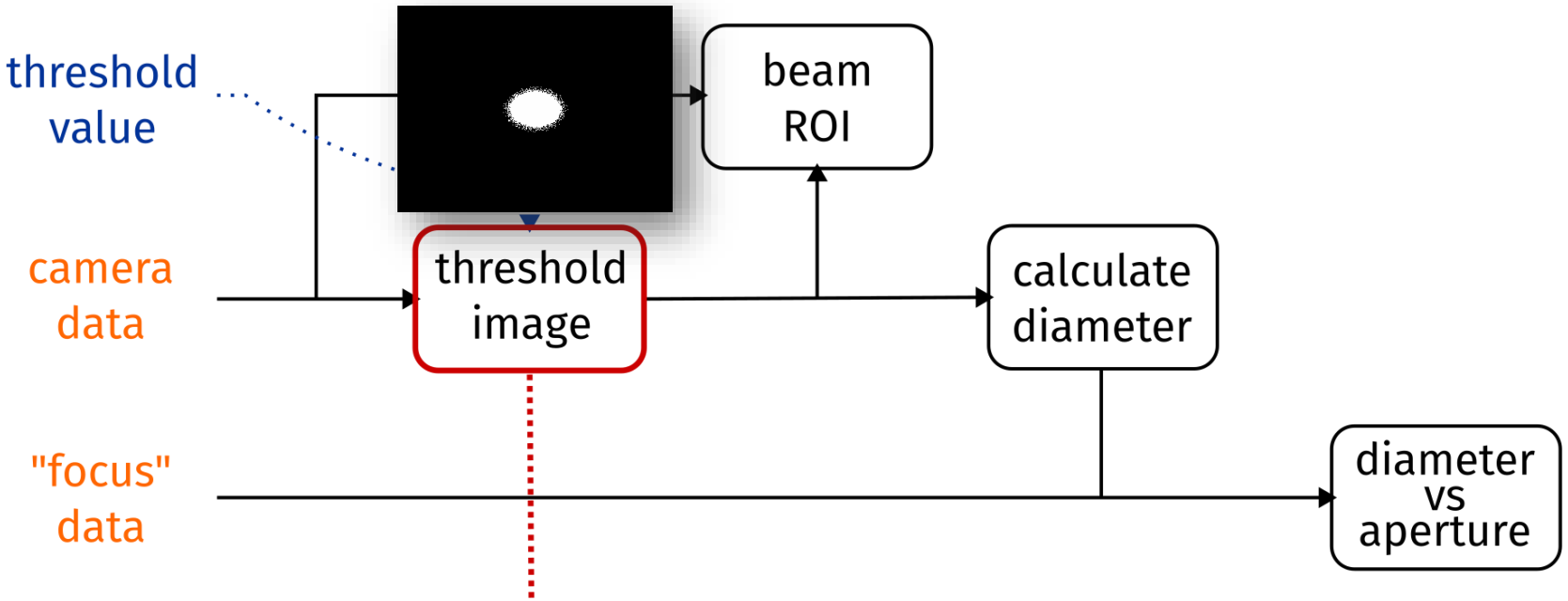
Toy example: analyze camera data



Simple view: access data from Karabo channel



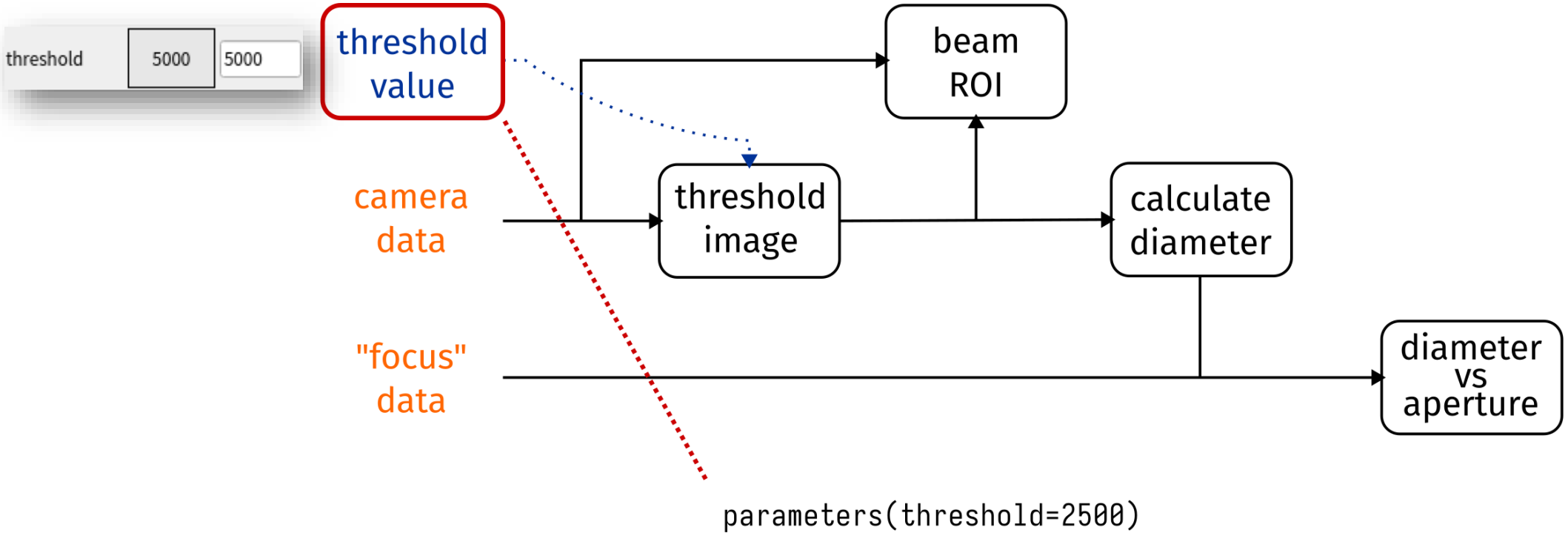
Use regular NumPy operations




```
@View.Image
def threshold_mask(data: "camera"):
    return (data >= threshold).astype(np.uint8)
```

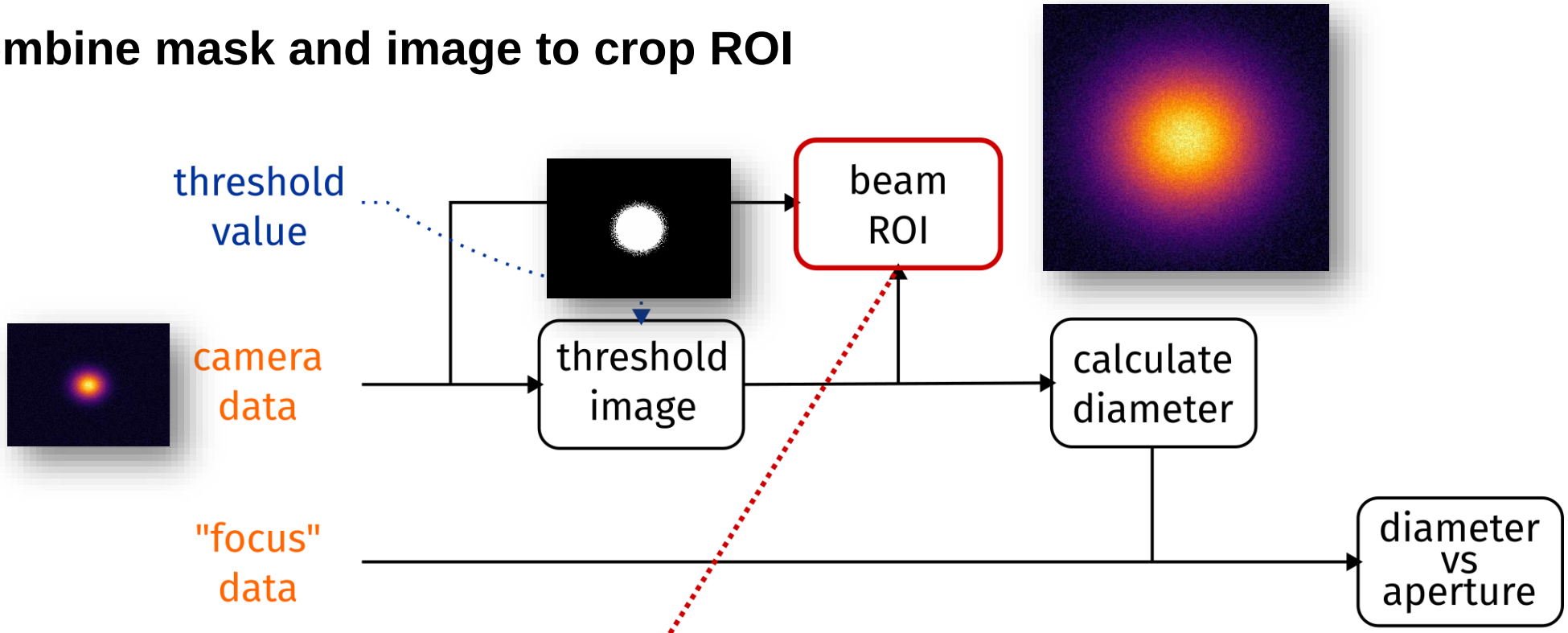
Note: `threshold value` not yet defined, don't want to hardcode in context file

Specify parameters to tweak at runtime



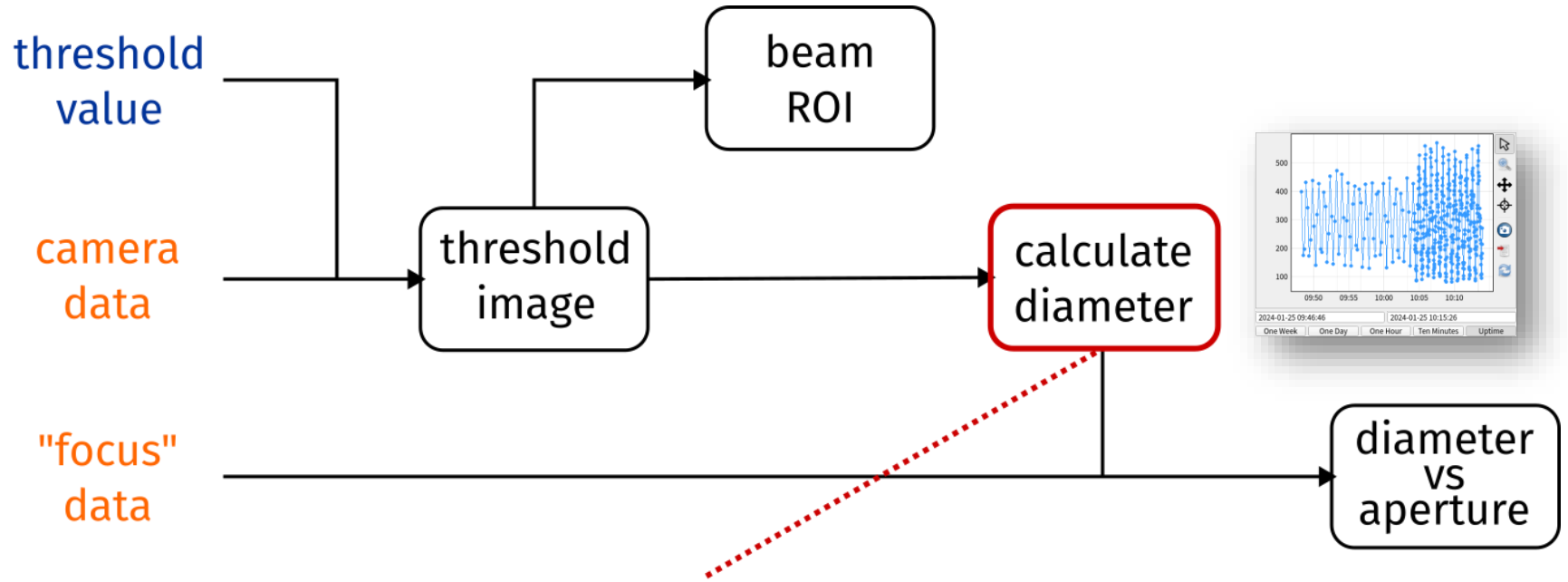
User-supplied value automatically used in previous view

Combine mask and image to crop ROI



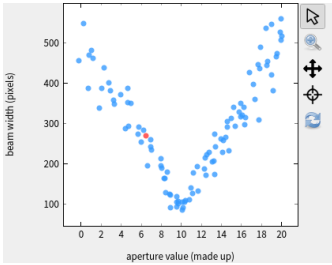

```
@View.Image
def beam_roi(image: "camera", mask: "threshold_mask"):
    slice_y, slice_x = ndimage.find_objects(mask, max_label=1)[0]
    return image[slice_y, slice_x]
```

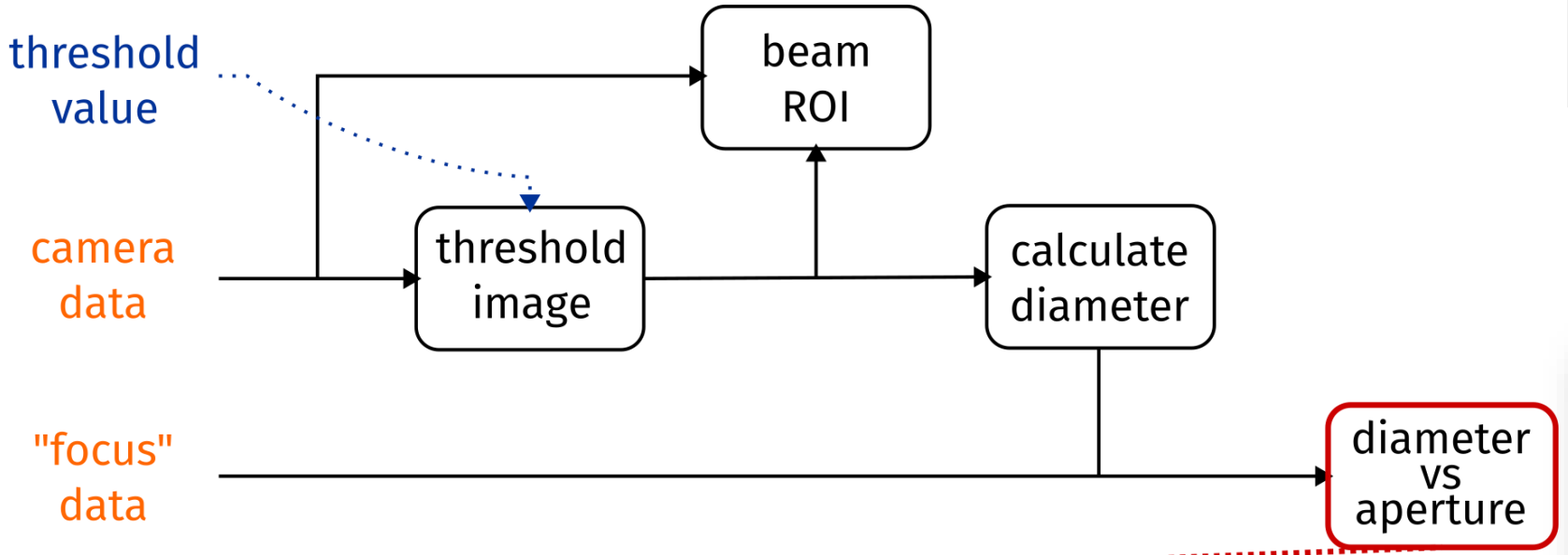
Use standard Python scientific libraries to compute metrics

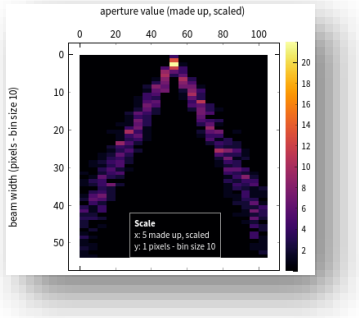



```
@View.Scalar
def beam_diameter(mask: "threshold_mask"):
    return measure.regionprops(label_image=mask)[0].equivalent_diameter_area
```

Plot variables against each other

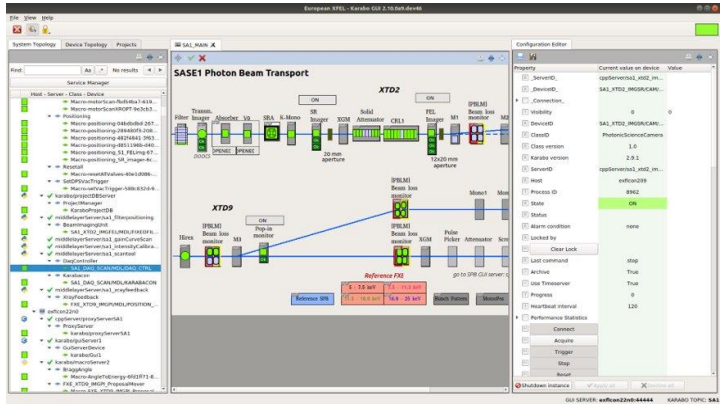






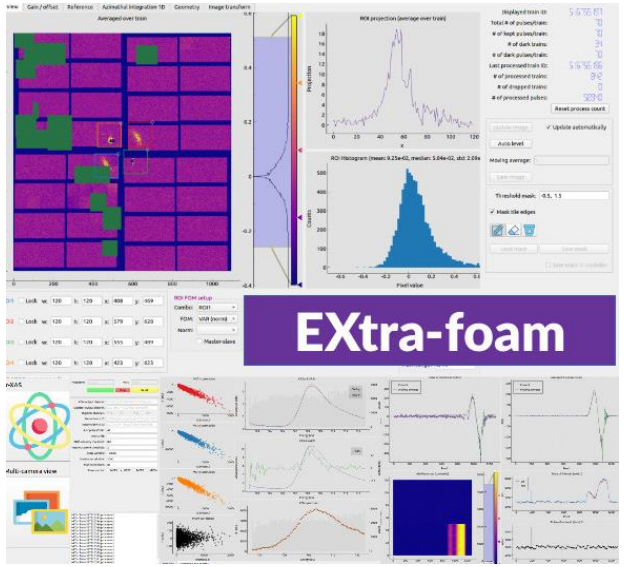

```
@View.Matrix_Histogram(y_step=10)
def diameter_vs_aperture(diameter: "beam_diameter", aperture: "aperture_value"):
    return aperture, diameter
```

Karabo devices



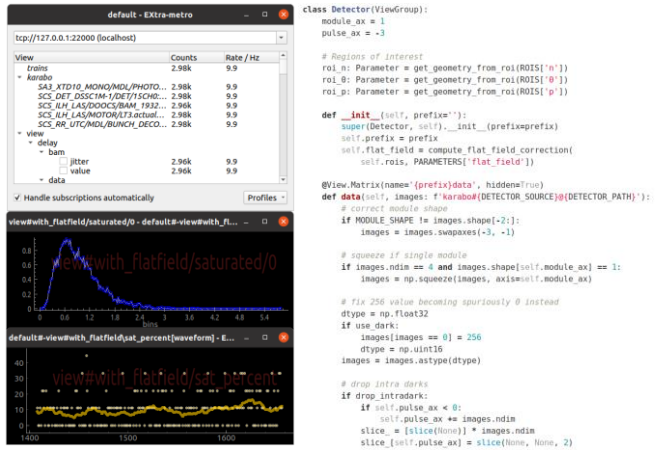
https://rtd.xfel.eu/docs/karabo/en/latest/index.html

EXtra-foam



https://extra-foam.readthedocs.io

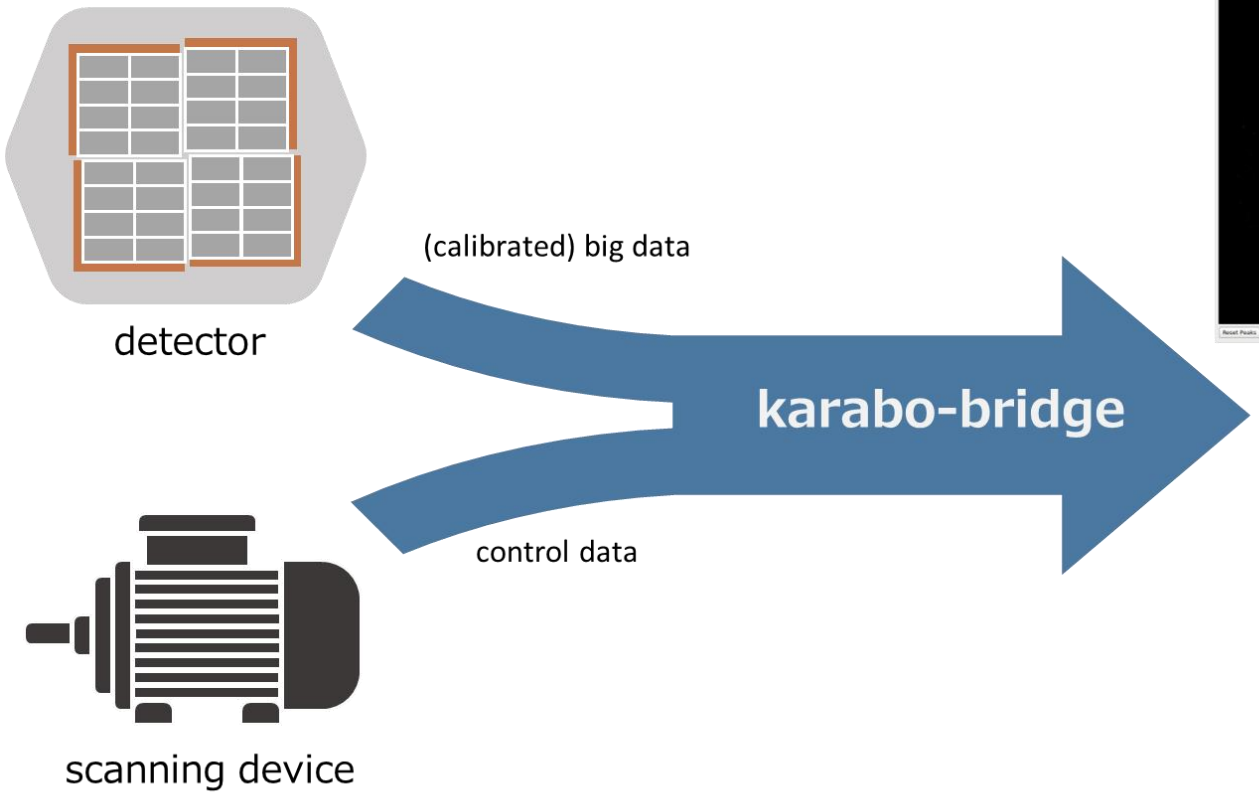
EXtra-metro

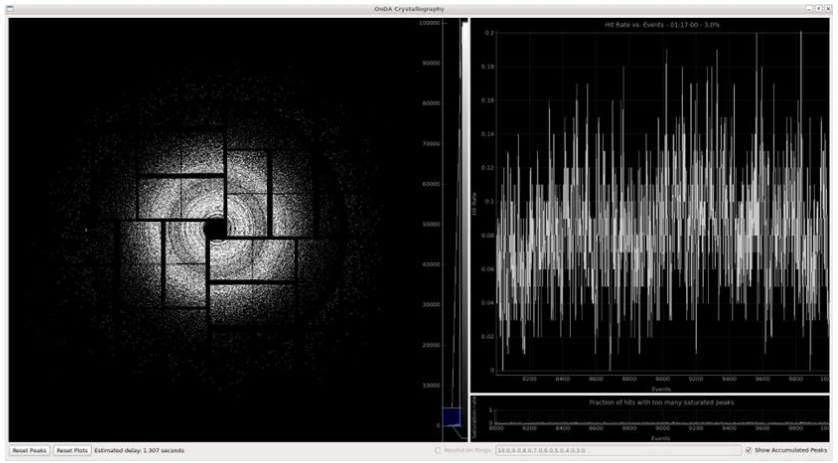


https://rtd.xfel.eu/docs/metropc/en/latest/index.html

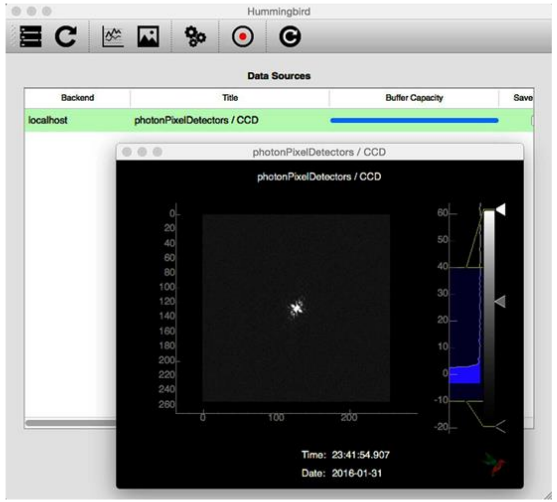
Custom tools

karabo-bridge: enables external applications to access data from the control system





OnDA



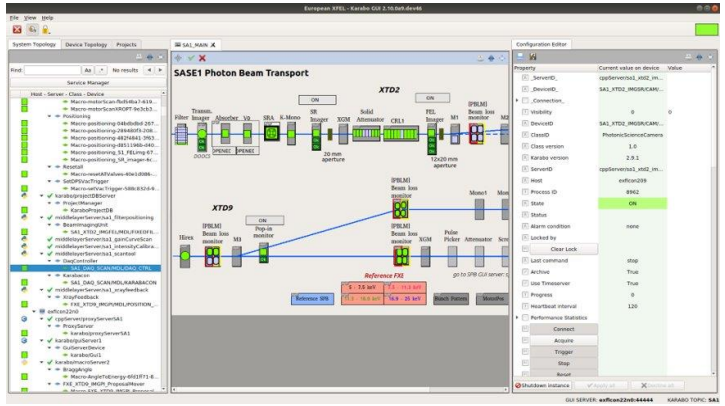
Hummingbird

karabo-bridge: the data can be retrieved from a bridge server by using the in-house developed bridge client

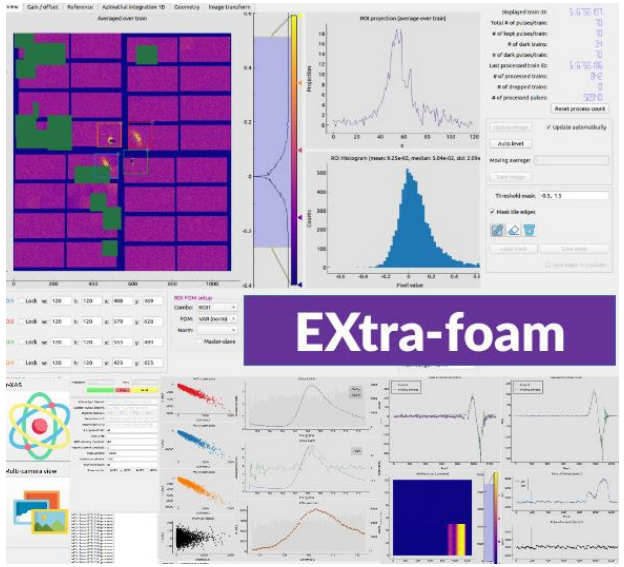
```
>>> from karabo_bridge import Client
>>> krb_client = Client('tcp://server-host-name:12345')
>>> data, metadata = krb_client.next()
>>> data.keys()
dict_keys(['source1', 'source2', 'source3'])
>>> data['source1'].keys()
dict_keys(['param1', 'param2'])
>>> metadata['source1']
{'source1': {'source': 'source1',
  'timestamp': 1528476983.744877,
  'timestamp.frac': '744877000000000000',
  'timestamp.sec': '1528476983',
  'timestamp.tid': 10000000073}}
```

Client libraries:
https://github.com/European-XFEL/karabo-bridge-py
https://github.com/European-XFEL/karabo-bridge-cpp

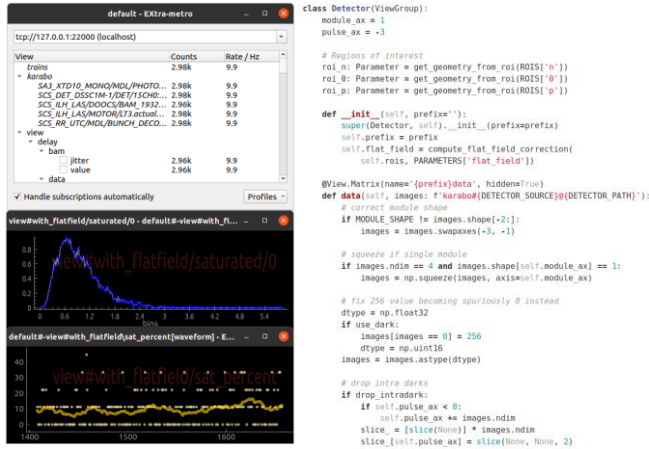
Summary: different levels of flexibility



Karabo devices



EXtra-foam

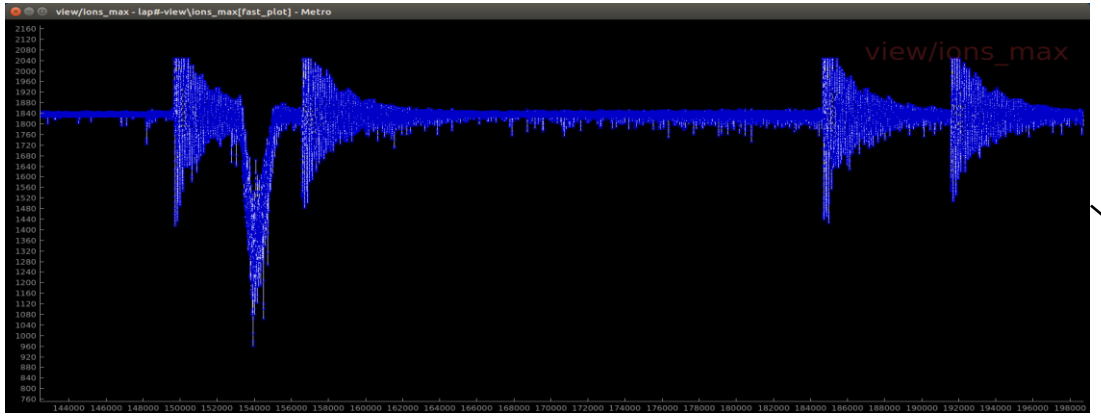


EXtra-metro

Custom tools
(using Karabo-bridge)

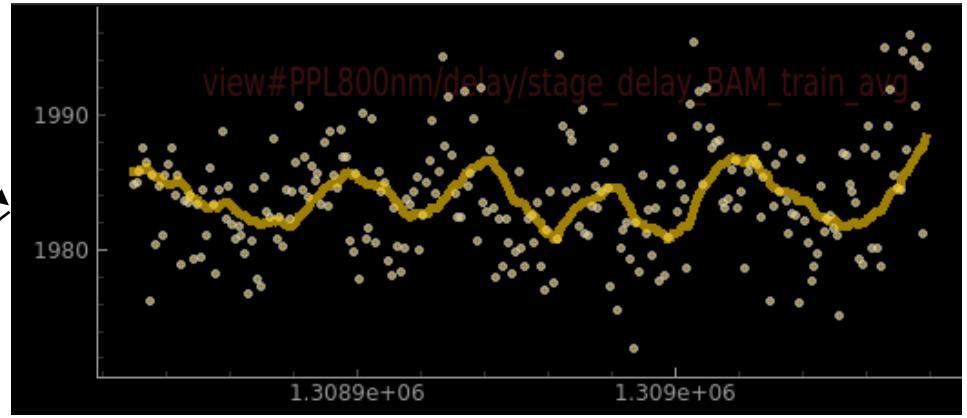
Bonus slides

EXtra-metro: programmable custom online analysis framework

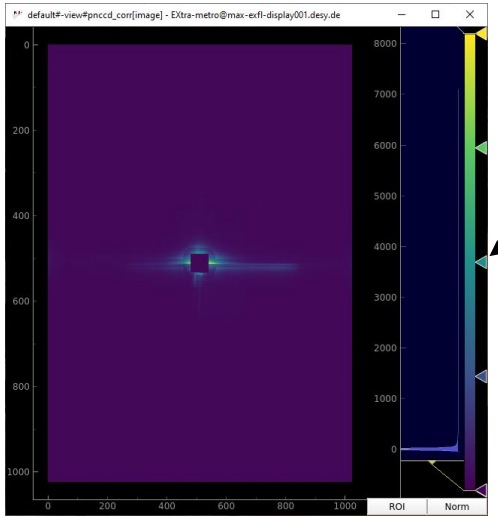



```
spectra_by_pulse  # (n_pulses, samples) array with spectra.
hits = 0  # Number of hits per pulse train
    for pulse_spectrum in spectra_by_pulse:
    # Check integrated intensity in specific region.
    if -pulse_spectrum[180:220].sum() > 2400:
        hits += 1
```

Count hits in spectrum



Filter images for hits




```
detector_frame  # (n_pulses, samples) array with spectra.
hits  # Number of hits per pulse train.
if hits > 0:
    return detector_frame
```

Conveniently run Python functions on data pipelines

Annotate the kind of output for automatic plotting

Specify data sources

```
@View.Scalar
def hits_per_trains(trace: 'SQS_DIGITIZER_UTC1/ADC/1:network'
                           '[digitizers.channel_2_C.raw.samples]'):
    spectra_by_pulse = separate_pulses(trace)
    hits = 0  # Number of hits per pulse train.

    for pulse_spectrum in spectra_by_pulse:
        # Check integrated intensity in specific region.
        if -pulse_spectrum[peak_roi].sum() > peak_value:
            hits += 1

    return hits
```

Expose configurable parameters

Return value for visualization or reuse

Build analysis pipelines by combining small functions