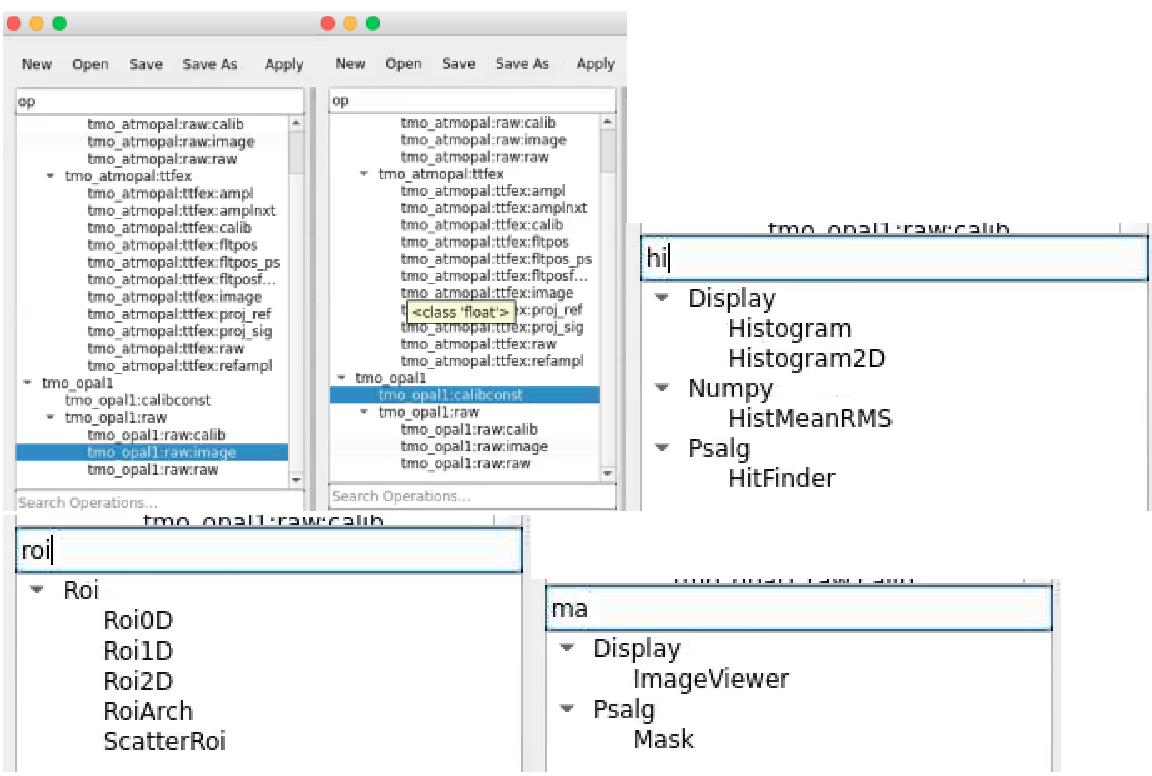
# AMI Examples for Detector Geometry Mask and RoiArch



## Command to start

```
ami-local -b 1 -f interval=1 psana://exp=tmoc00118,run=222,dir=/cds/data/psdm/prj/public01/xtc
```

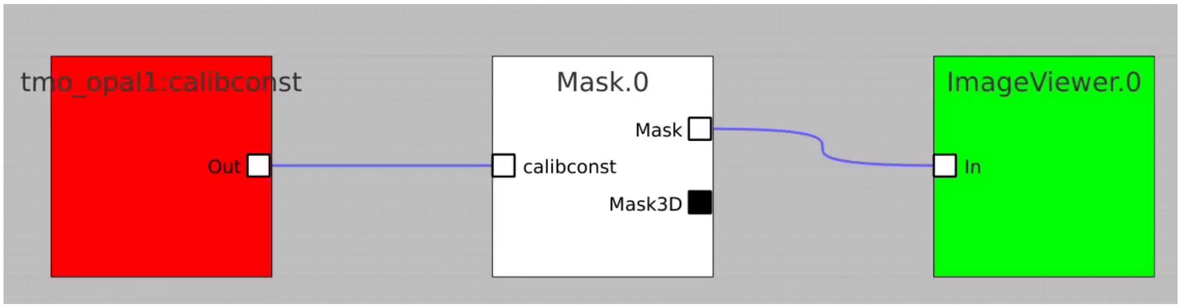
## Finding control nodes for this example



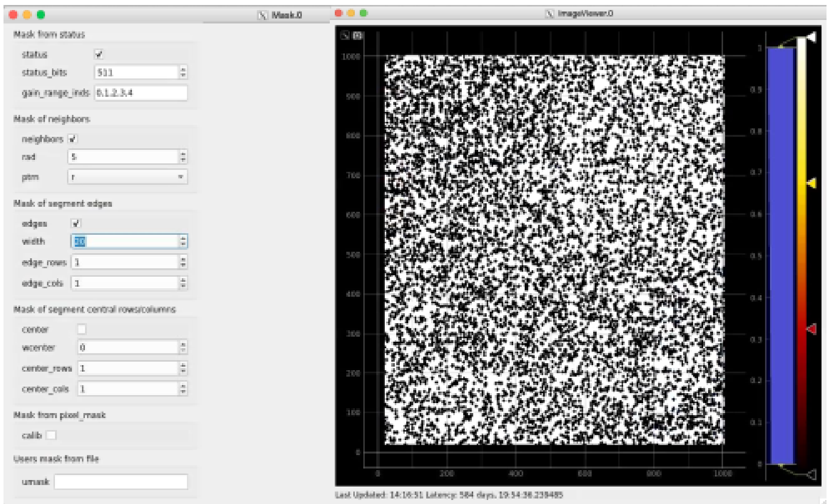
# Mask

Mask (2D or 3D array) is created mainly from calibration constants of particular detector.

For example, bring control nodes to the flowchart and connect their terminals as shown below.




Click on "Apply" button, then click on Mask.0 and ImageViewer.0 control nodes to open editor for mask parameters and mask image viewer.



Adjust the mask parameters using editor window and click on "Apply" button again. Mask image will have changed according to set parameters.

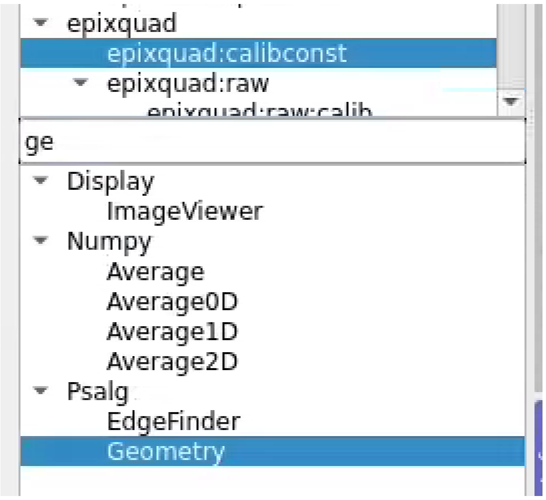
# Geometry and Detector Image

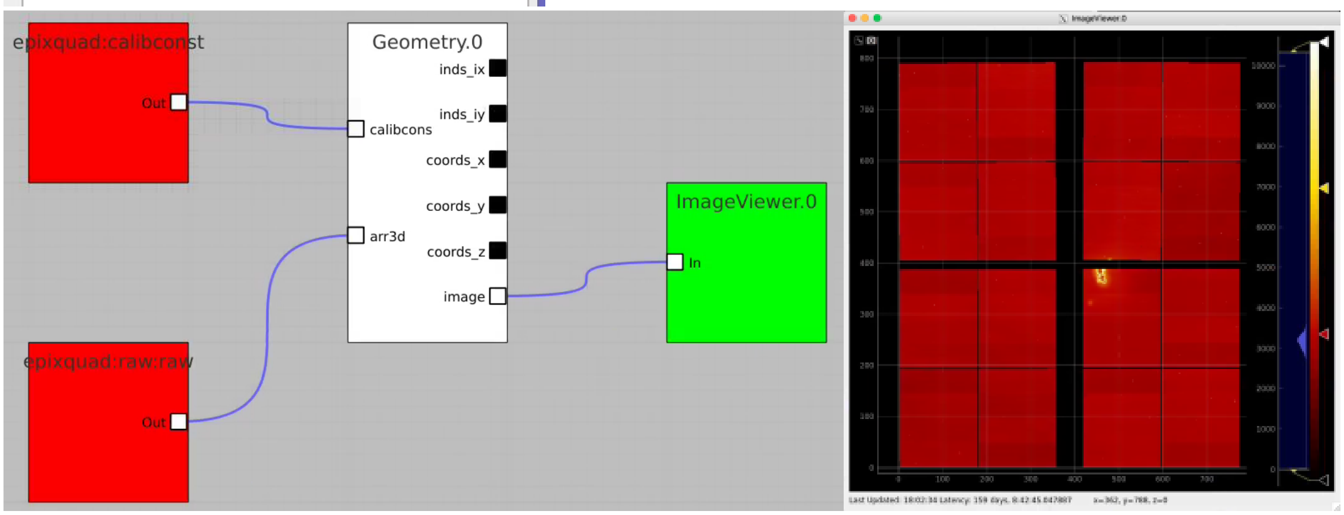
## Command to test

**Command to test Geometry**

```
ami-local -b 1 -f interval=1 psana://exp=uedcom103,run=7,dir=/cds/data/psdm/prj/public01/xtc
```

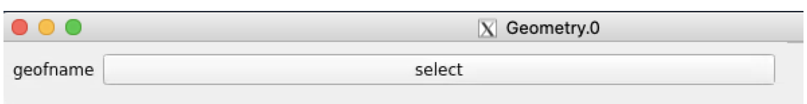
## Use geometry with image



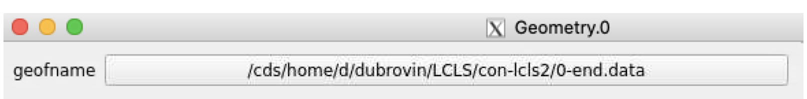




## Select optional geometry file

By default geometry data comes from DB associated with experiment or detector. Optional geometry file can be specified through the click on Geometry Control Node to deal with parameters window.

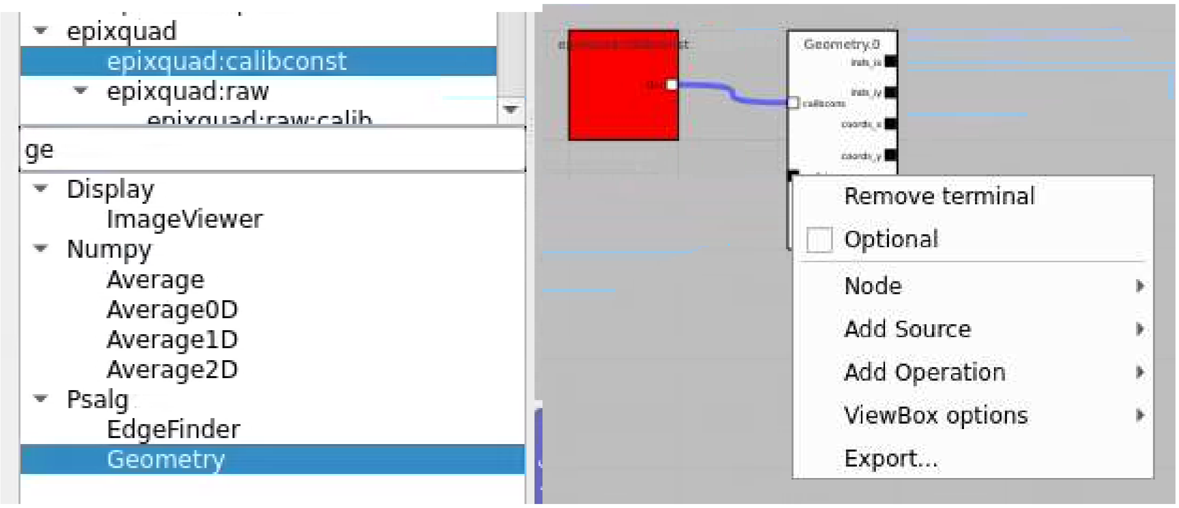


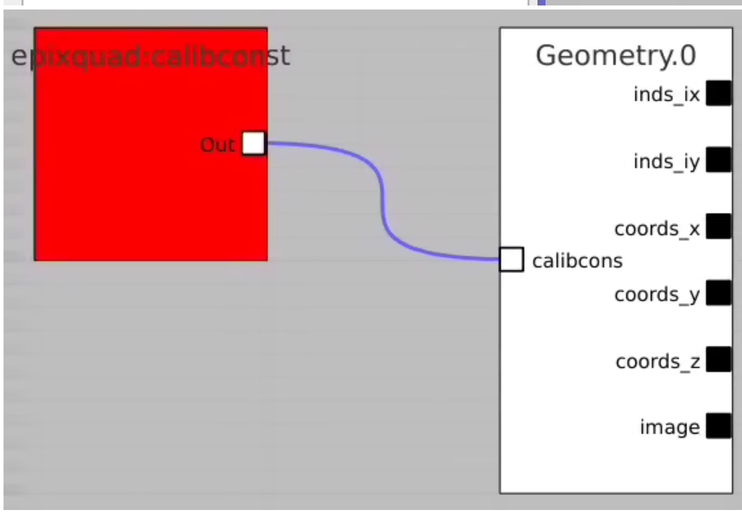

Click on "select" button and select desired geometry file, e.g.




## Use Geometry without input array

Add the Geometry Control Node as usually and remove input terminal for arr3d. Click on "Apply" and use any geometry output parameters except "image".



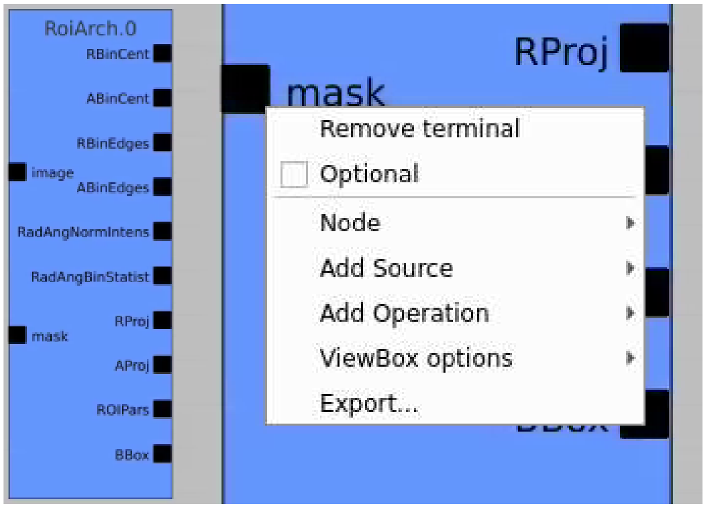




# RoiArch

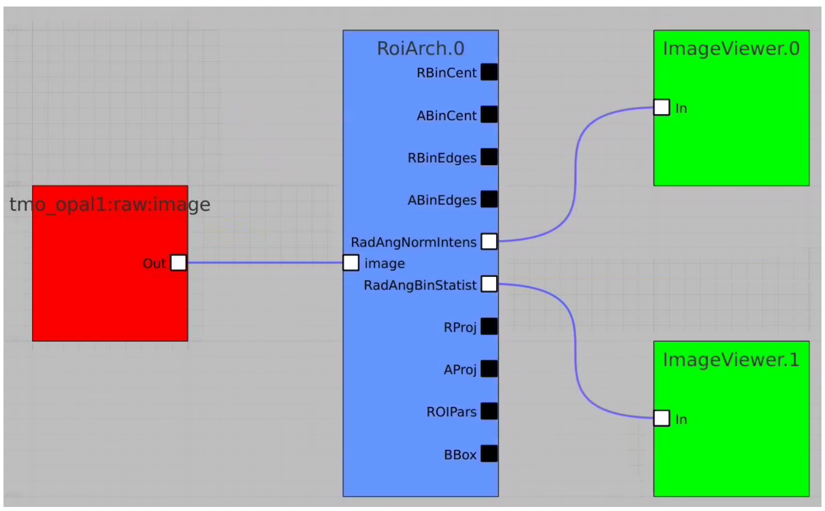
## Simple example

In simple case we use 2D image as input array without mask and ImageViewer with bin numbers in stead of actual scales for radial and angular dimensions.

First, bring RoiArch to the flowchart, right-click on input "mask" terminal and select "Remove terminal" on pop-up window. (Currently ami2 does not work with optional terminals).




Bring other control nodes to the flowchart and connect them as shown below and click on "Apply" button.




Click on RoiArch control node and adjust desired ROI region on pop-up window.

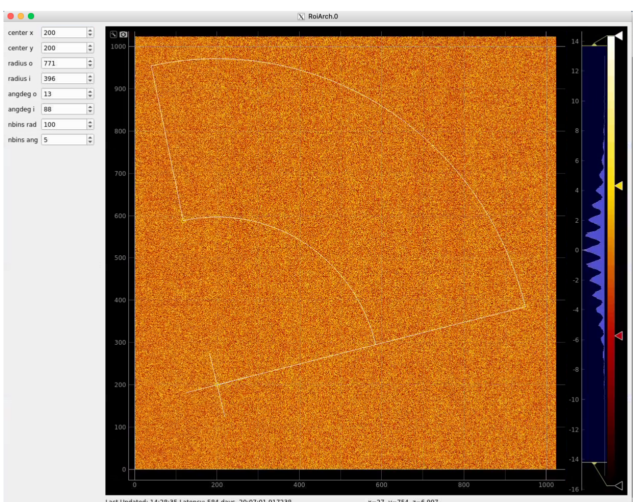
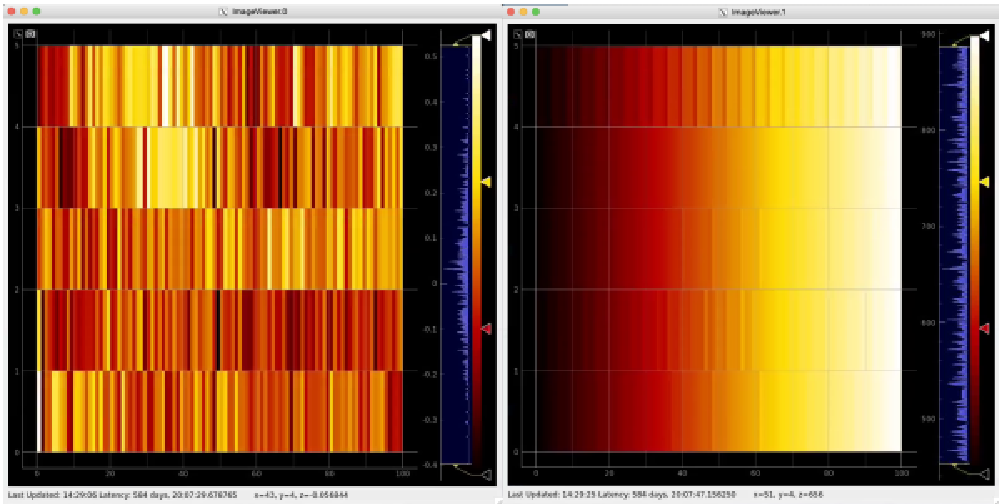



Image can be moved and zoomed by click-and-drag and scrolling mouse, respectively.

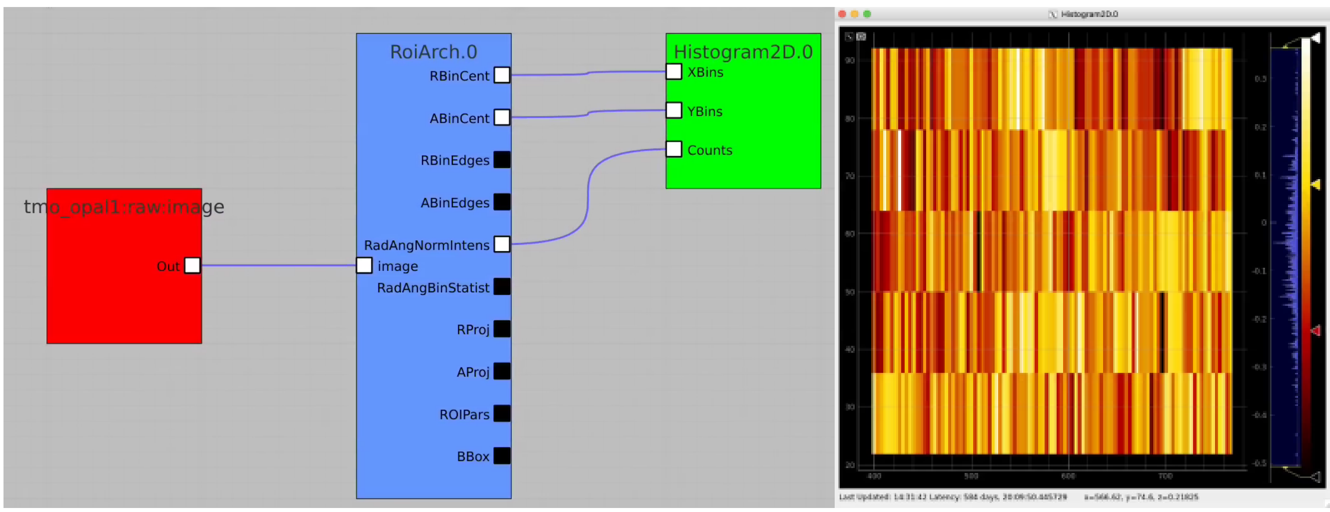
ArchROI can be adjusted using 3 control points for center, external radius and entire rotation, internal radius and arch angular size. Alternatively, ArchRoi parameters can be adjusted using editor for parameters in the left side of the window. Click on "Apply" button after mask parameters are set.

Then click on ImageViewer.0 and ImageViewer.1 control nodes to see images for r-angle per pixel normalized intensity and associated pixel per bin statistics.

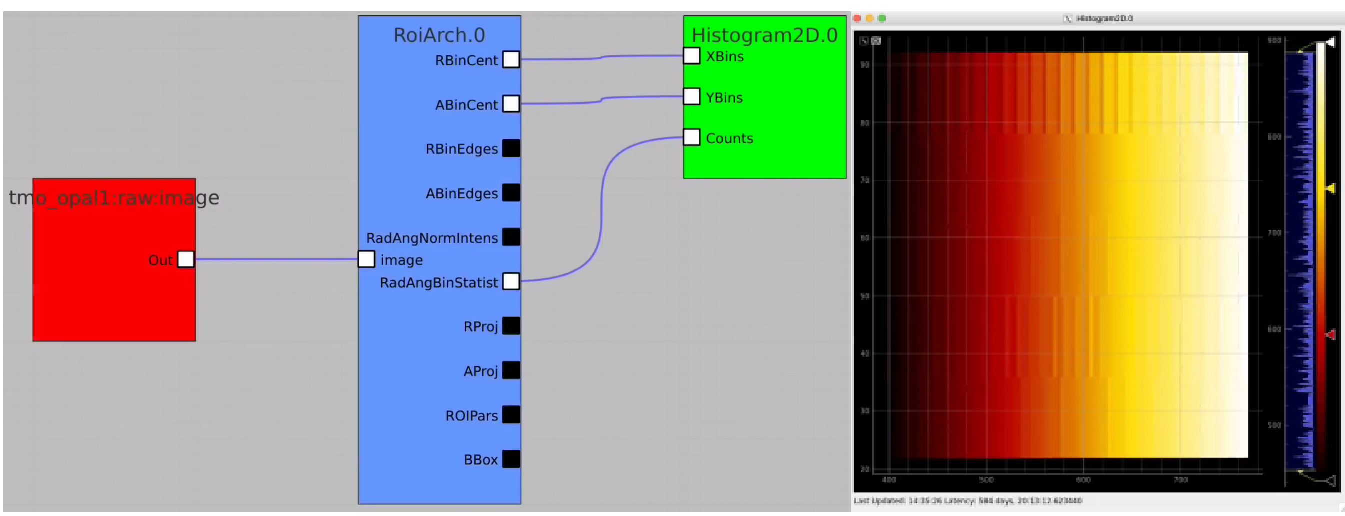


Plot radial-angular distribution with scales scale

**R-angular plot with scales for normalized intensity**

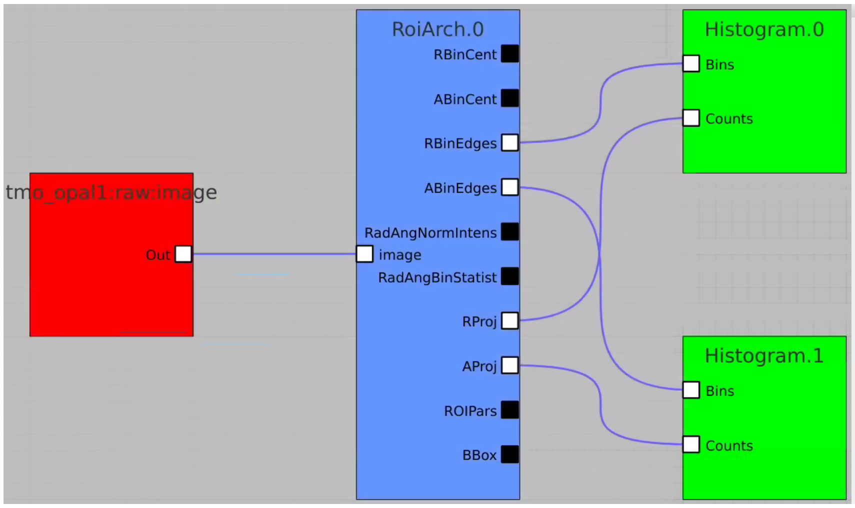


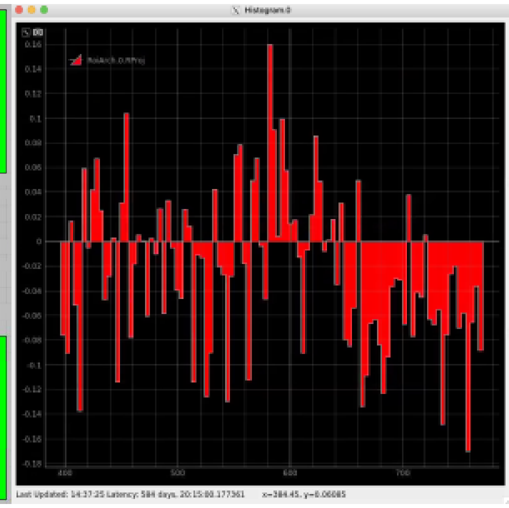

**R-angular plot with scales for per-bin pixel statistics**

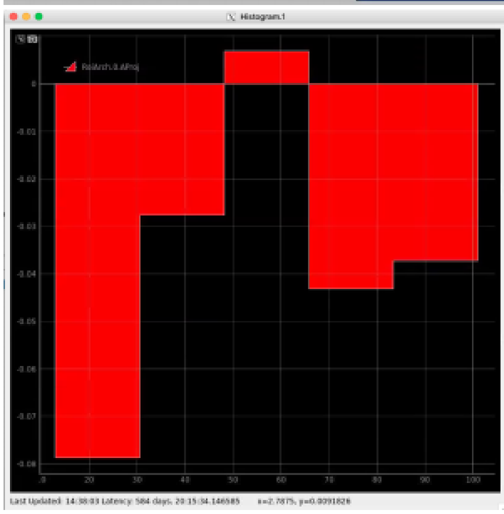



**Radial and angular projections with normalized per pixel intensity**

Arrays of N+1 bin edges should be used for 1D histograms



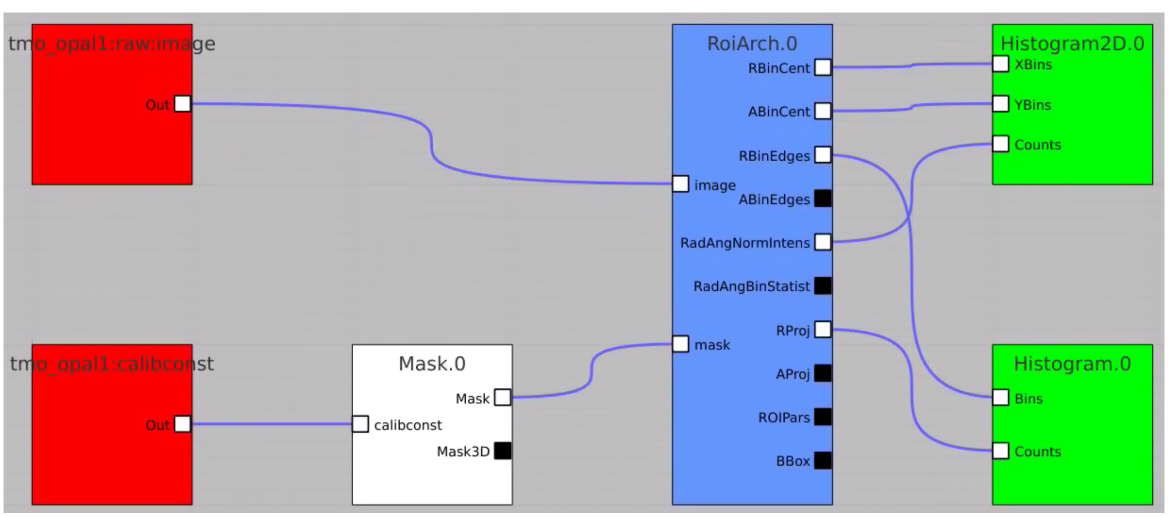





## Apply mask

Bring RoiArch to the flowchart, connect th Mask.0 control node output (2D) Mask with RoiArch "mask" input terminal.

Then build other control nodes as in previous examples, click on "Apply" button, etc.

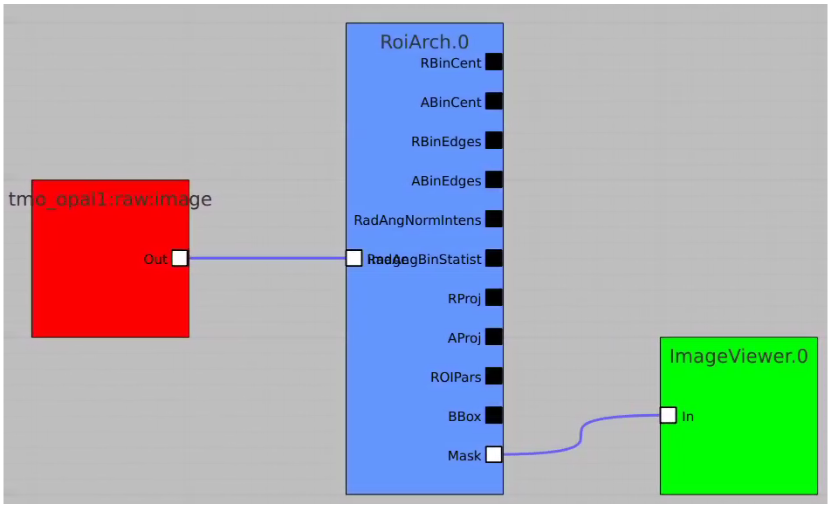


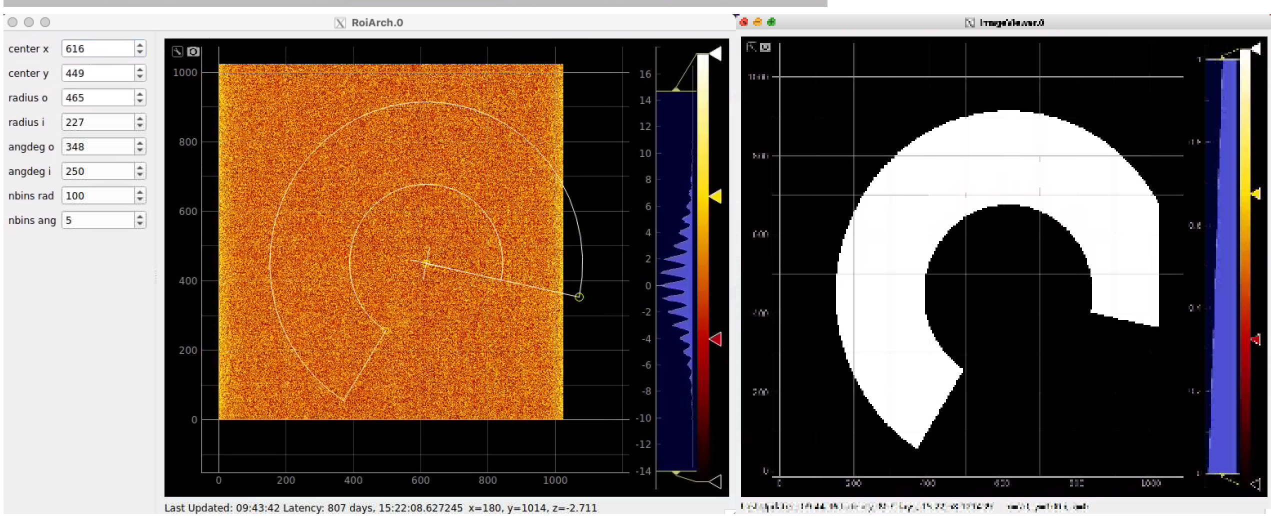

## RoiArch output for 2-d mask

**Test command**

```
ami-local -b 1 -f interval=1 psana://exp=tmoc00118,run=222,dir=/cds/data/psdm/prj/public01/xtc
```

- NEW: Node RoiArch got new output terminal Mask (Array2d)
- Use tmo_opal1:raw:image (Array2d) as input to RoiArch, draw/edit arch, get Mask (Array2d) shaped as input image.




# Usage of RoiArch for multi-panel detector and make 2-d and 3-d masks

**Test command**

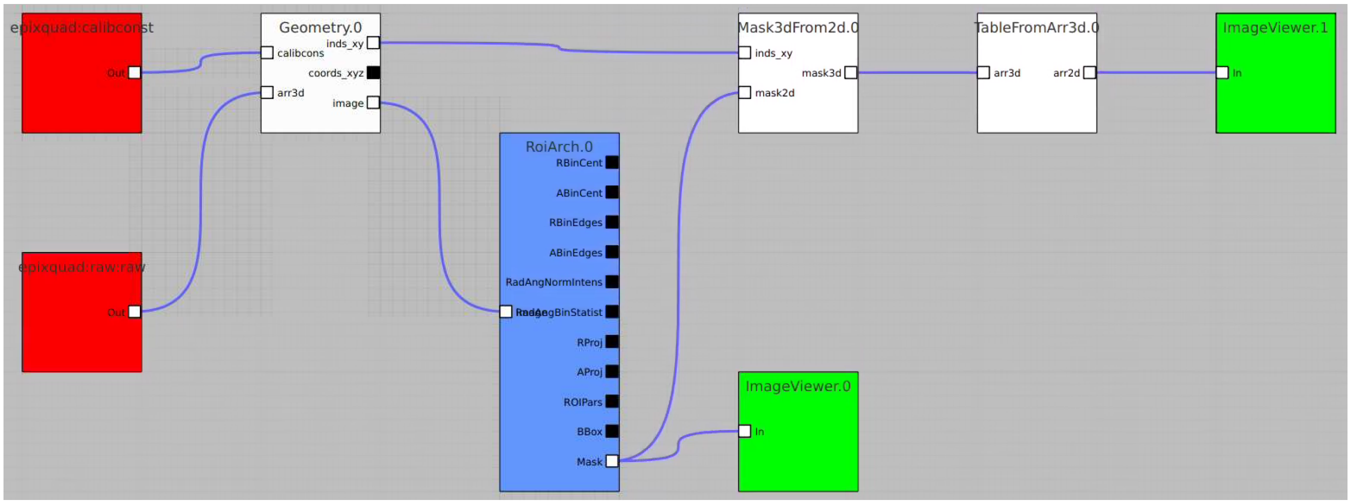
```
ami-local -b 1 -f interval=1 psana://exp=uedcom103,run=7,dir=/cds/data/psdm/prj/public01/xtc
```

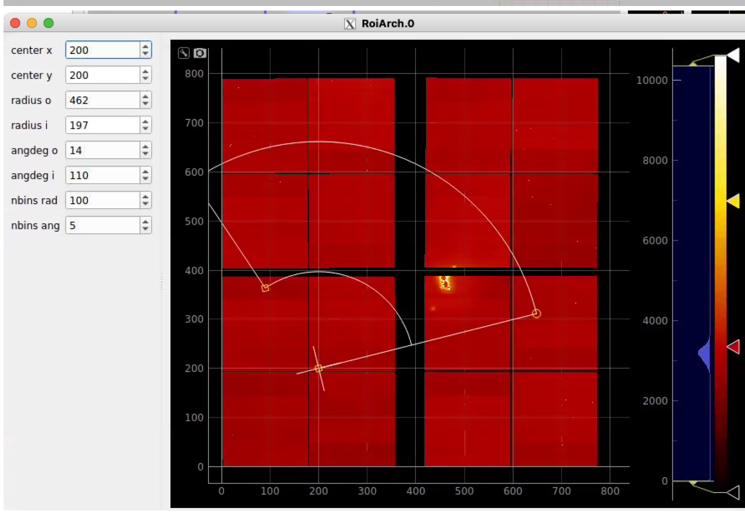
NEW:

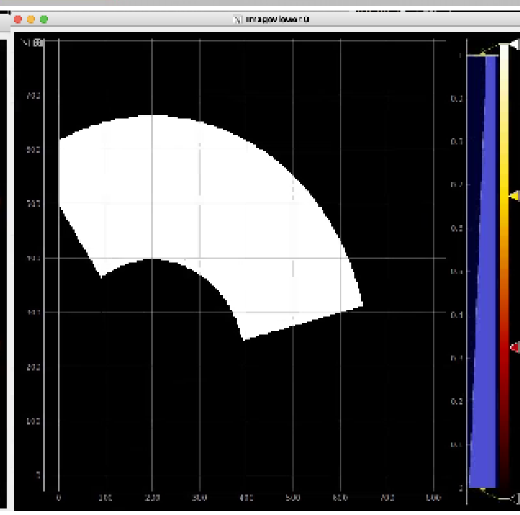
- in node Geometry separate arrays are combined in lists of index and coordinate arrays
- new nodes Mask3dFrom2d and TableFromArr3d

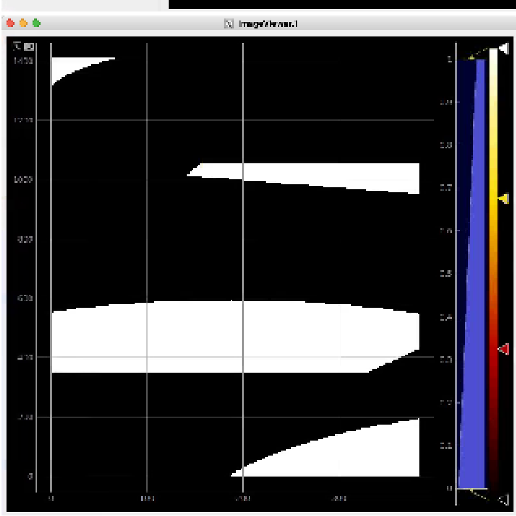
Example shows how to

- use epixquad:raw:raw (Array3d) and epixquad:calibconst (dict) as input to Geometry to generate image (Array2d),
- pass this image to RoiArch input, draw/edit arch, get Mask (Array2d) shaped as input image,
- use node Mask3dFrom2d with input mask2d from RoiArch and inds_xy from Geometry to generate 3-d mask array shaped as raw data,
- use nodes TableFromArr3d to convert 3-d mask to 2-d table of segments (without Geometry) to plot with ImageViewer.1



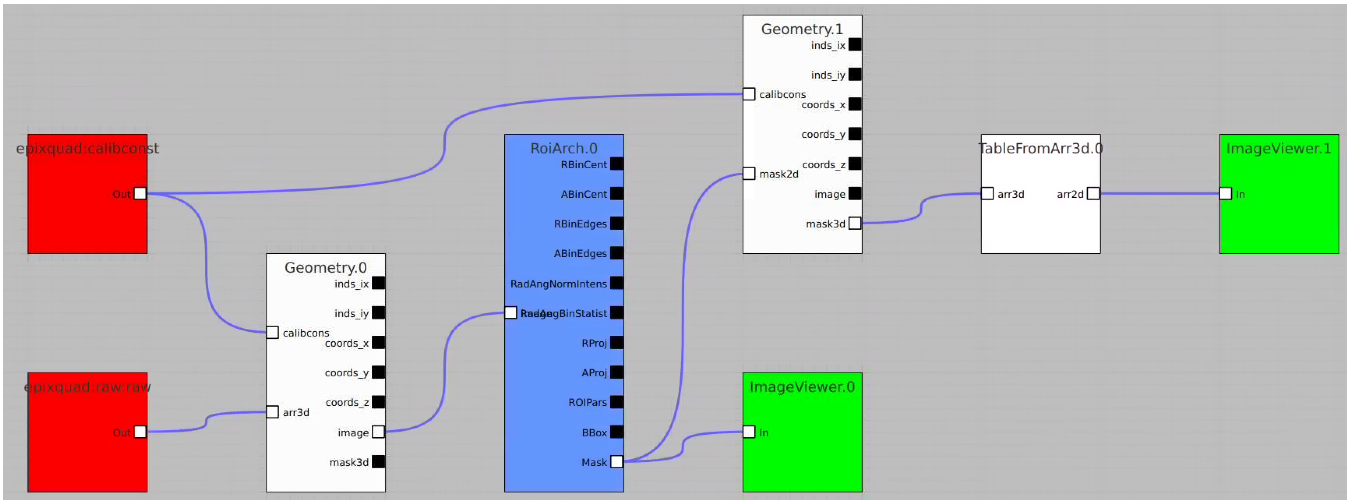


# Potential extension of the node Geometry for 3-d mask output (TEMPORARY FOR PRESENTATION ONLY)

### AMI issues

- Mask 3-d can be converted directly in the node Geometry (Mask3dFrom2d may not be required...),
  - it needs in optional input parameters
- AMI does not work correctly with default input terminals
  - unused terminals must be remover interactively
  - if intermediate terminal removed input is moved to previous one...
- Also need in a few parameters in TableFromArr3d, but

- AMI fails to pickle qt modules (ComboBox, etc) and crashes...

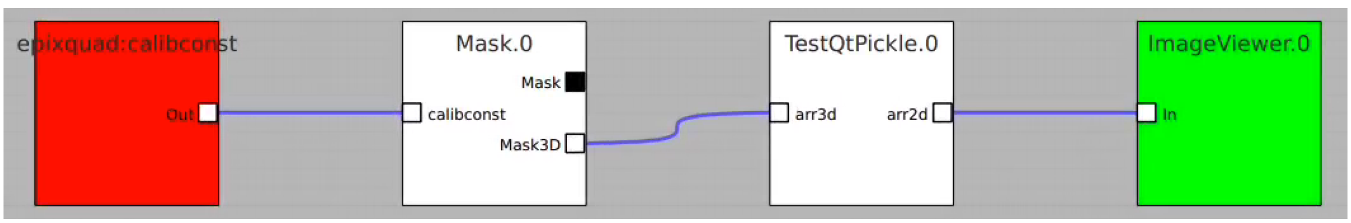


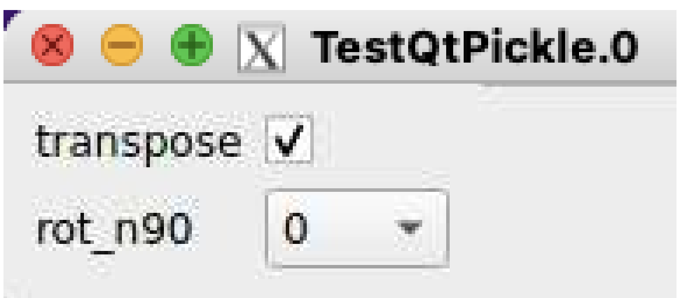

## Example of issue with pickle of qt object

**Test code and command**

```
Essential code is in ami/flowchart/library/Psalg.py

ami-local -b 1 -f interval=1 psana://exp=uedcom103,run=7,dir=/cds/data/psdm/prj/public01/xtc
```




Crash:

```
[ 2023-03-01 16:27:41,647 | ami.flowchart.library.Psalg | INFO    ] TestQtPickleProdProd.__init__ kwa: {'transpose': <PyQt5.QtWidgets.QCheckBox object at 0x7f5cd24c4310>, 'rot_n90': <PyQt5.QtWidgets.QComboBox o
ject at 0x7f5cd24c4430>}
[ 2023-03-01 16:27:41,648 | asyncqt._QEventLoop | ERROR   ] Task exception was never retrieved
future: <Task finished name='Task-5' coro=<Flowchart.loadFile() done, defined at /cds/home/d/dubrovin/LCLS/con-lcls2/ami/ami/flowchart/Flowchart.py:427> exception=TypeError("cannot pickle 'QCheckBox' object")>
Traceback (most recent call last):
  File "/cds/home/d/dubrovin/LCLS/con-lcls2/ami/ami/flowchart/Flowchart.py", line 446, in loadFile
    await ctrl.applyClicked(build_views=False)
  File "/cds/home/d/dubrovin/LCLS/con-lcls2/ami/ami/flowchart/Flowchart.py", line 760, in applyClicked
    await self.graphCommHandler.add(graph_nodes)
  File "/cds/home/d/dubrovin/LCLS/con-lcls2/ami/ami/comm.py", line 2130, in _post_dill
    await self._sock.send(dill.dumps(payload))
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 263, in dumps
    dump(obj, file, protocol, byref, fmode, recurse, **kwds)#, strictio)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 235, in dump
    Pickler(file, protocol, **_kwds).dump(obj)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 394, in dump
    StockPickler.dump(self, obj)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 487, in dump
    self.save(obj)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 388, in save
    StockPickler.save(self, obj, save_persistent_id)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 560, in save
    f(self, obj)  # Call unbound method with explicit self
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 931, in save_list
    self._batch_appends(obj)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 955, in _batch_appends
    save(x)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 388, in save
    StockPickler.save(self, obj, save_persistent_id)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 603, in save
    self.save_reduce(obj=obj, *rv)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 717, in save_reduce
    save(state)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 388, in save
    StockPickler.save(self, obj, save_persistent_id)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 560, in save
```

...

```
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 388, in save
    StockPickler.save(self, obj, save_persistent_id)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 560, in save
    f(self, obj)  # Call unbound method with explicit self
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 1186, in save_module_dict
    StockPickler.save_dict(pickler, obj)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 971, in save_dict
    self._batch_setitems(obj.items())
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 1002, in _batch_setitems
    save(v)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 388, in save
    StockPickler.save(self, obj, save_persistent_id)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 560, in save
    f(self, obj)  # Call unbound method with explicit self
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 1186, in save_module_dict
    StockPickler.save_dict(pickler, obj)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 971, in save_dict
    self._batch_setitems(obj.items())
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 997, in _batch_setitems
    save(v)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/site-packages/dill/_dill.py", line 388, in save
    StockPickler.save(self, obj, save_persistent_id)
  File "/cds/sw/ds/ana/conda2/inst/envs/ps-4.5.26/lib/python3.9/pickle.py", line 578, in save
    rv = reduce(self.proto)
TypeError: cannot pickle 'QCheckBox' object
```

XQuartz Applications Edit Window Help zoo

# PythonEditor for calib components of epix10ka and epixhr detectors

This example shows how to access in ami all components of the det.raw.calib(...) method such as calibration constants, per-panel trbit and asicPixelConfig, per-event gain factors and pedestals, mask, common mode correction, etc.

## AMI test command for epixquad and epixhr

**Command to test**

```
# for epixquad - 4-panel epix10ka:
ami-local -b 1 -f interval=1 psana://exp=ueddaq02,run=569,dir=/cds/data/psdm/prj/public01/xtc

# for single panel epixhr:
ami-local -b 1 -f interval=1 psana://exp=rixx45619,run=121,dir=/cds/data/psdm/prj/public01/xtc
```

## Code example for PythonEditor

In this example the PythonEditor box receives input objects for raw (np.ndarray), config, calibconst, accesses and prints internal components of the det.raw.calib(...) method and returns 2-d image for one retrieved arrays. For imaging, the 3-d array of common mode correction is converted to 2-d table of segments.

**Code example for PythonEditor**

```
import psana.detector.utils_calib_components as ucc

np, info_ndarr, table_nxn_epix10ka_from_ndarr =\
  ucc.np, ucc.info_ndarr, ucc.psu.table_nxn_epix10ka_from_ndarr


class EventProcessor():

    def __init__(self):
        self.counter = 0
        self.cc = None

    def begin_run(self):
        pass

    def end_run(self):
        pass

    def begin_step(self, step):
        pass

    def end_step(self, step):
        pass

    def on_event(self, raw, config, calibconst, *args, **kwargs):
        self.counter += 1
        if self.cc is None:
            self.cc = ucc.calib_components_epix(calibconst, config)
        cc = self.cc

        kwa = {'status': True}
        cmpars = (0, 7, 300, 10)

        ctypes   = cc.calib_types() # list of calibration types
        npanels  = cc.number_of_panels() # number of panels/segments in the detector
        peds     = cc.pedestals() # OR cc.calib_constants('pedestals')
        gain     = cc.gain()      # OR cc.calib_constants('pixel_gain') # ADU/keV
        gfactor  = cc.gain_factor() # keV/ADU
        status   = cc.status() # 4-d array of pixel_status constants
        comode   = cc.common_mode()  # tuple of common mode correction parameters
        trbit_p0 = cc.trbit_for_panel(0)  # list of per-ASIC trbit for panel 0
        ascfg_p0 = cc.asicPixelConfig_for_panel(0) # matrix of asicPixelConfig for panel 0
        mask     = cc.mask(**kwa) # mask defined by kwa
        dettype  = cc.dettype() # detector type, e.g.  "epix10ka" or "epixhr"
        cbitscfg = cc.cbits_config_detector()
        cbitstot = cc.cbits_config_and_data_detector(raw, cbitscfg)
        gmaps    = cc.gain_maps_epix(raw) # gain map
        pedest   = cc.event_pedestals(raw) # per-pixel array of pedestals in the event
        factor   = cc.event_gain_factor(raw) # per-pixel array of gain factors in the event
        calib0   = cc.calib(raw, cmpars, **kwa) # method calib
        cmcorr   = cc.common_mode_correction(raw, cmpars=cmpars, **kwa) # common mode correction
        arrf = np.array(raw & cc.data_bit_mask(), dtype=np.float32) - pedest

        print('== Event %04d ==' % self.counter)
#        print('config', cc.config)
#        print('calib_metadata', cc.calib_metadata('pedestals'))
        print('calib_types', ctypes)
        print(info_ndarr(peds, 'pedestals'))
        print(info_ndarr(cc.gain(), 'gain'))
        print(info_ndarr(gfactor, 'gain_factor'))
        print(info_ndarr(status, 'status'))
        print('common_mode from caliconst', str(comode))
        print('number_of_panels', npanels)
        print('trbit_for_panel(0)', trbit_p0)
        print(info_ndarr(ascfg_p0, 'asicPixelConfig_for_panel(0)'))
        print(info_ndarr(raw, 'raw'))
        print(info_ndarr(mask, 'mask'))
        print('dettype', dettype)
        print(info_ndarr(cbitscfg, 'cbitscfg'))
        print(info_ndarr(cbitstot, 'cbitstot'))
        print(info_ndarr(gmaps, 'gmaps'))
```

```
    print(info_ndarr(pedest, 'pedest'))
    print(info_ndarr(factor, 'factor'))
    print(info_ndarr(cmcorr, 'cmcorr'))
    print(info_ndarr(arrf, 'raw(data bits)-peds'))
    print(info_ndarr(calib0, 'calib0'))

    # det.raw.calib(...) algorithm close reproduction
    calib1 = None
    if True:
        arrf1 = arrf.copy()
        cc.common_mode_apply(arrf1, gmaps, cmpars=cmpars, **kwa)
        calib1 = arrf1 * factor if mask is None else arrf1 * factor * mask
        print(info_ndarr(calib1, 'calib1'))

    # det.raw.calib(...) - effective algorithm
    calib2 = None
    if True:
        arrf2 = arrf.copy() + cmcorr
        calib2 = arrf2 * factor if mask is None else arrf2 * factor * mask
        print(info_ndarr(calib2, 'calib2'))

    # img = cmcorr[0, 144:, :192]
    # img = table_nxn_epix10ka_from_ndarr(cmcorr)
    img = table_nxn_epix10ka_from_ndarr(calib1)
    print(info_ndarr(img, 'img'))
    return img
```

## Composition of Control Nodes in the example

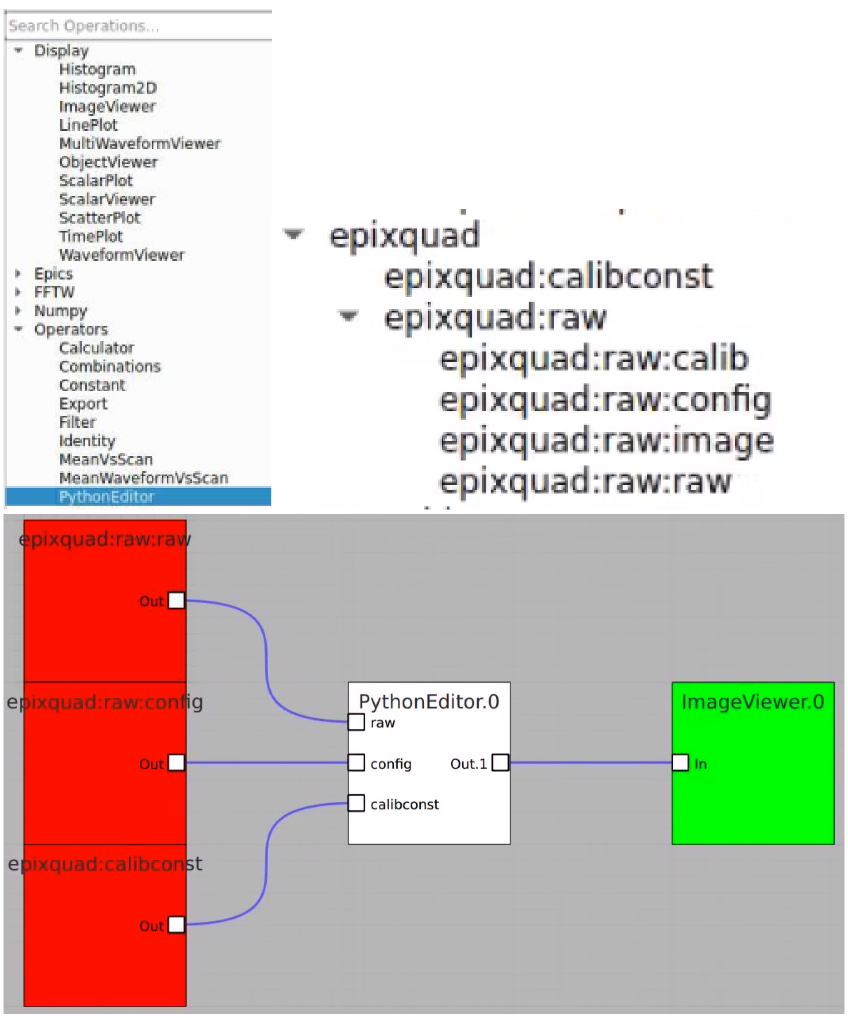
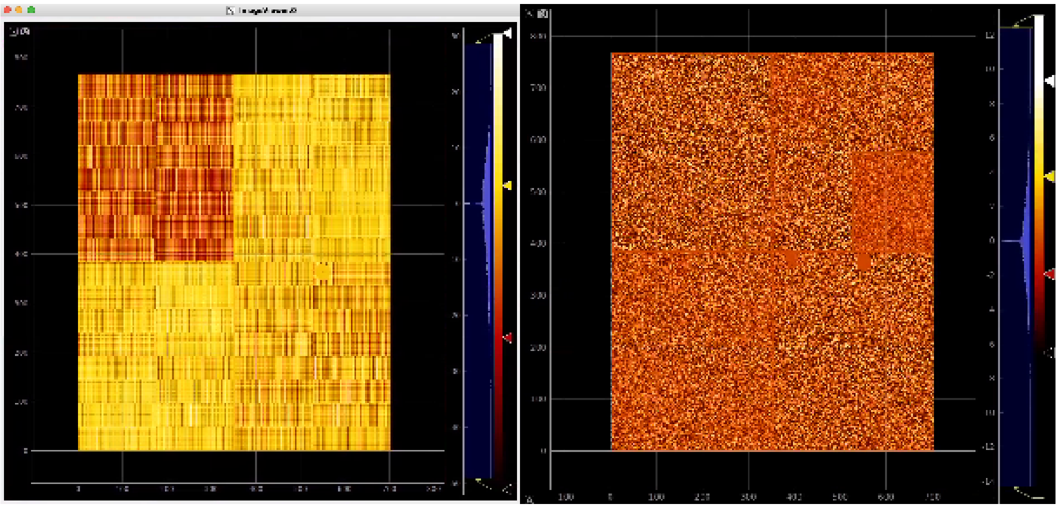


Image for common mode correction and calib1, respectively



## Implementation of the class calib_components_epix

- Module detector/utils_calib_components.py will be available in releases grater than ps-4.5.24.
- It is implemented for epix10ka and epixhr detector types.
- Code example is also available in the head of this module.

## Test example in psana

Example shows how to initialize and use the calib_components_epix object using parameters retrieved from the detector interface.

This script can be executed in the lcls2 psana environment by the command like

```
python ./lcls2/psana/psana/detector/test_issues_2023.py 2
```

# References

- ami
- Area Detector Interface
- Area detector mask examples