PACS Photometry
Data Processing and Calibration: processing up to Maps

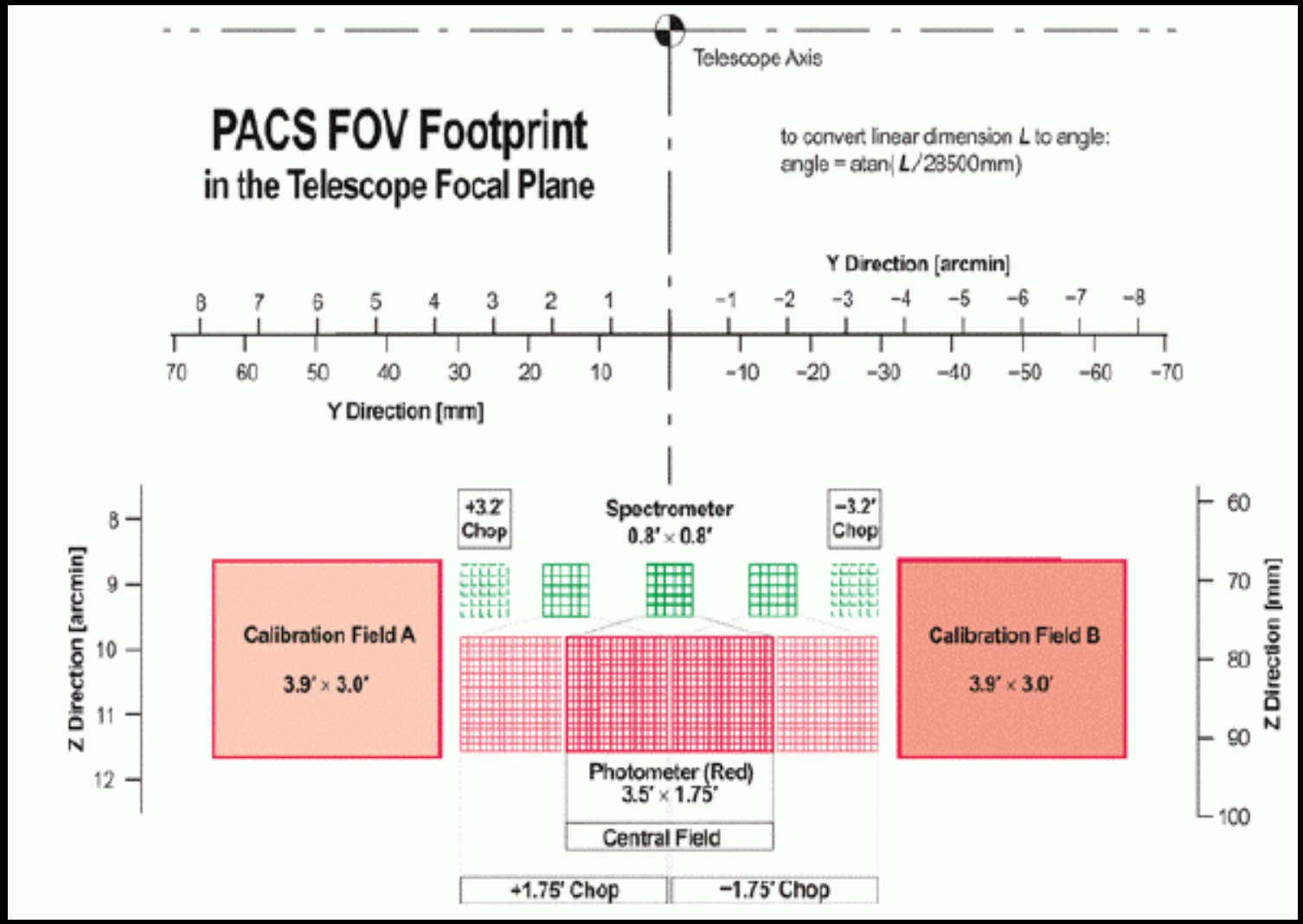
P.Popesso, D.Lutz, E.Wieprecht,
M.Wetzstein, S.Berta

4-5 December 2008,
Herschel pre-launch DP-Workshop

Outlines (2)

- PACS photometry AOTs
- PACS data: compression & decompression
- PACS photometry Product definition
- PACS pipeline & hipe: current status
- how to process and calibrate the data: the pipeline step by step
- open issues and future plans

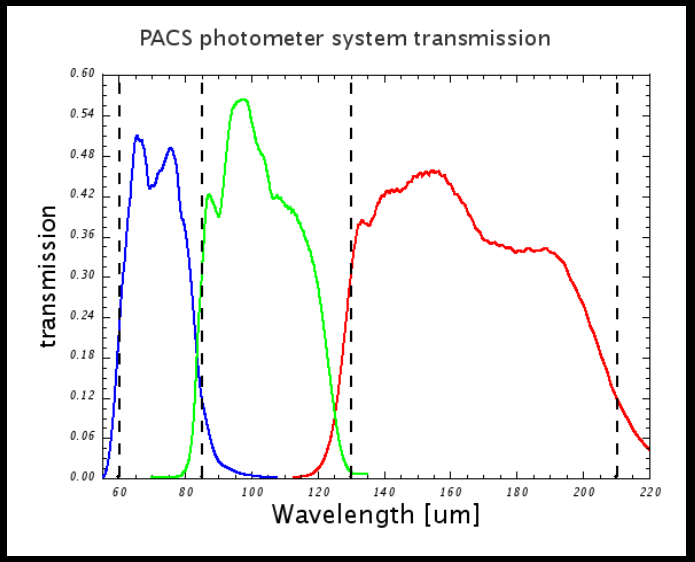
PACS FOV


Telescope Axis
PACS FOV Footprint
in the Telescope Focal Plane
to convert linear dimension L to angle:
angle = atan(L / 28500mm)
Y Direction [arcmin]
8 7 6 5 4 3 2 1 -1 -2 -3 -4 -5 -6 -7 -8
70 60 50 40 30 20 10 -10 -20 -30 -40 -50 -60 -70
Y Direction [mm]
Z Direction [arcmin]
8 9 10 11 12
+3.2' Chop
Spectrometer
0.8' × 0.8'
-3.2' Chop
Calibration Field A
3.9' × 3.0'
Calibration Field B
3.9' × 3.0'
Photometer (Red)
3.5' × 1.75'
Central Field
+1.75' Chop
-1.75' Chop
60 70 80 90 100
Z Direction [mm]

PACS Photometer

PACS photometer overall characteristics/performances

	70μm	100μm	160μm
wavelength range (μm)	60-85	85-130	130-210
Resolution	~3		~2
pixel size (arcsec)	3.2		6.4
FOV (arcmin)	3.5 x 1.75		
FWHM (arcsec)	5.2	7.7	12



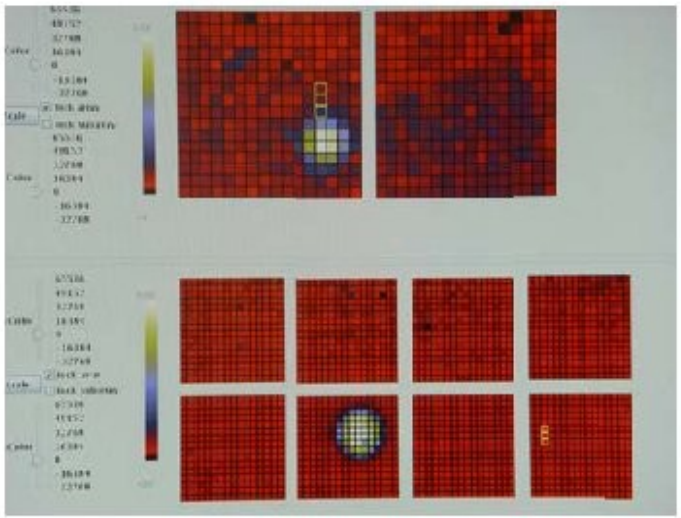


Figure 3.2. Snapshot a QLA screen during FM ILT testing where an external blackbody is seen through a 4mn aperture, simulating a source, much more extended than a point source. Top: red array, bottom: blue array.

PACS photometer predicted sensitivity 5σ-1 hour in mJy

central wavelength	70μm	100μm	160μm
off-array chopping	3.75	4.1	5.75
on-array chopping	2.7	2.8	4.1
scan mapping	2.25	2.4	3.4

PACS Photometry AOT

Four PACS photometry modes:

- point source photometry
- small source photometry
- raster map photometry
- scan map photometry

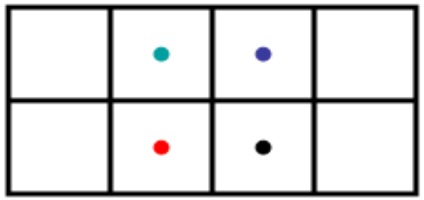
PACS Photometry AOT

Four PACS photometry modes:

- point source photometry
- small source photometry
- ~~raster map photometry~~
- scan map photometry

PACS Photometry AOT

Point Source



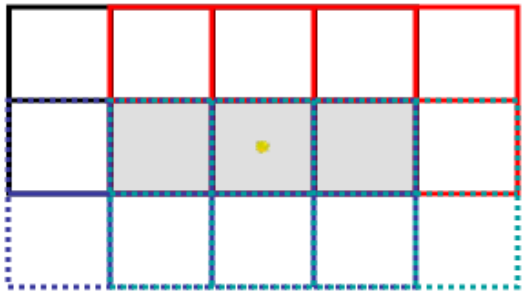
Nod 1 chop A

Nod 1 chop B

Nod 2 chop A

Nod 2 chop B

(nod1 chop A – nod1 chop B) – (nod2 chop A – nod2 chop B)



Point-source AOT footprint on the sky

Figure 4.1. Source positions in point-source photometry AOT. Sketch showing the source positions as a function of the nod and chopper positions. The Y-axis is to the left, the Z-axis to the top. Chop positions are defined by the internal chopper, while nod positions are defined by the satellite pointing. Dithering at each chopper position, performed with the internal chopper is not represented.

PACS Photometry AOT

Small Source

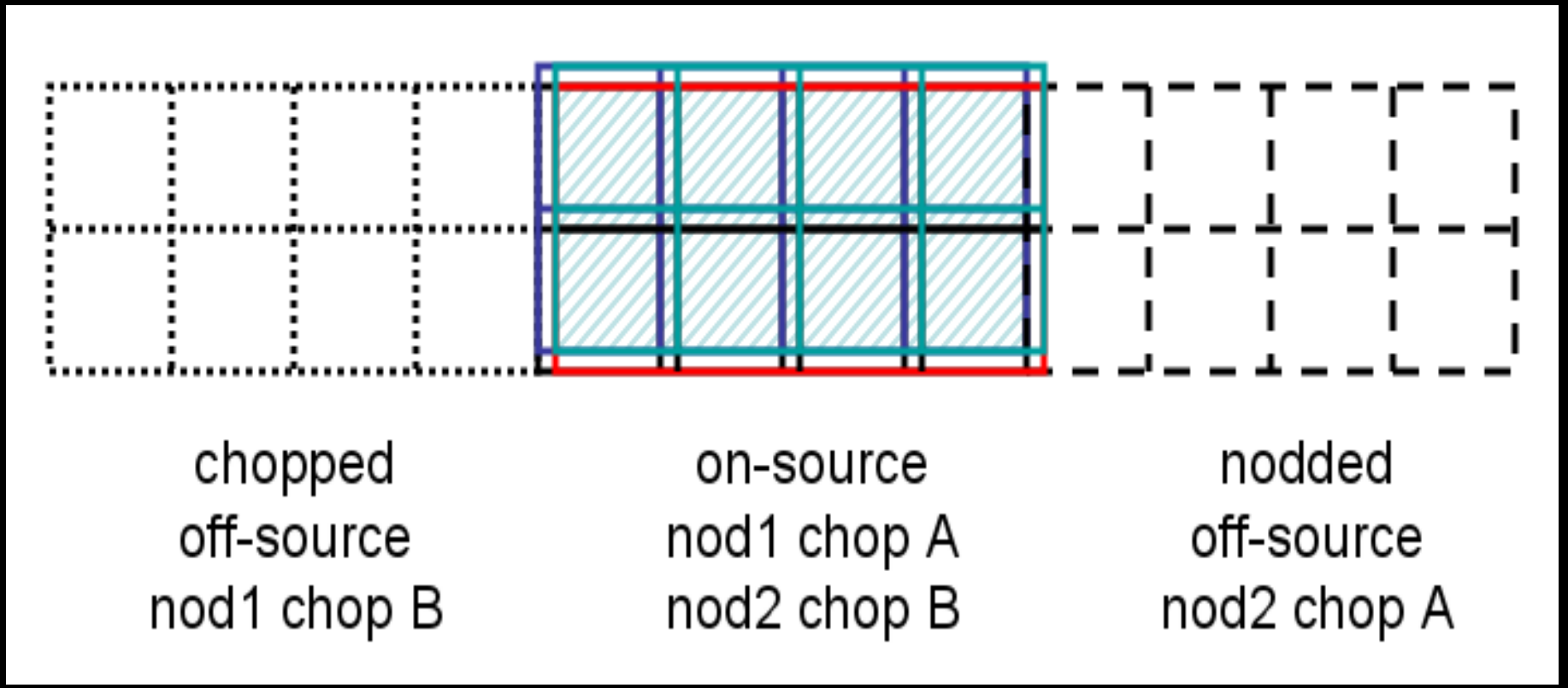
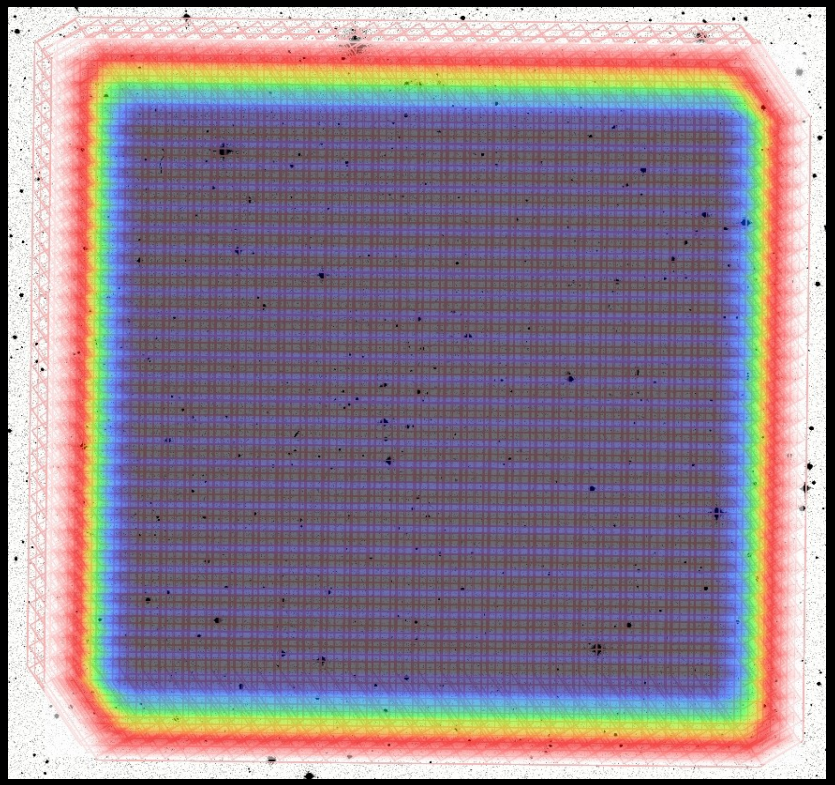


Figure 4.3. Footprint of detector on the sky in small-source photometry. The pointing sequence is colour coded and goes black, red, green, blue. By the Y-axis (long axis) motion alone, the horizontal gap between the 4 top and bottom matrices is still completely blind. The Z-axis motion allows to cover this area and leads to complete coverage. The completely covered area 3.2 by 1.5 arcmin at the end of the observation is indicated as a hatched zone.

PACS Photometry AOT

Scan Map Mode



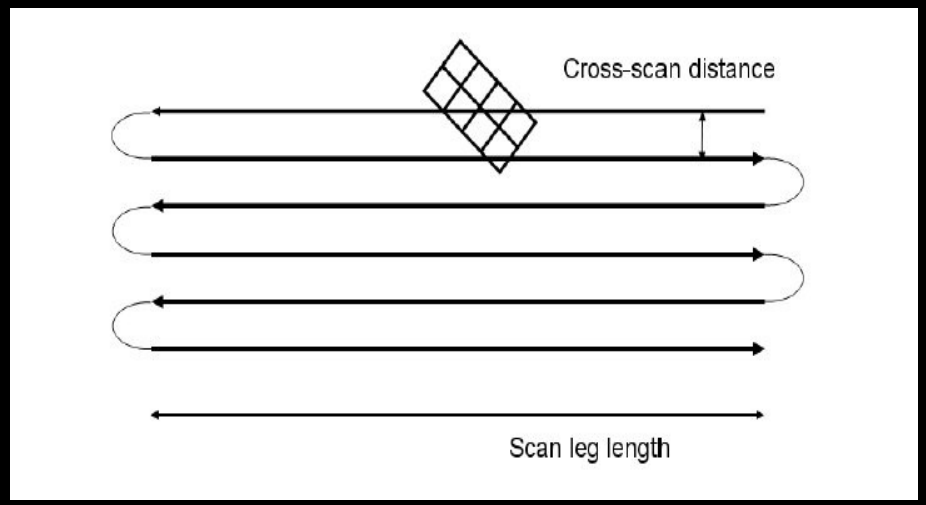


Figure 4.6. Example of PACS photometer scan map. Schematic of a scan map with 6 scan line legs. After the first line, the satellite turns left and continue with the next scan line in the opposite direction, just like in the raster map case. The reference scan direction is the direction of the first leg. Note that the turn around between line does take place as simplistically drawn in the figure.

The satellite slews continuously along parallel lines at a user-specified speed (10, 20 or 60 arcsec/s)

It is suggested to perform two scan maps of the same area with orthogonal coverage in order to remove more efficiently the stripping effects of the 1/f noise. For this purpose two AORs shall be concatenated in HSpot. In the second AOR the map orientation angle is then increased by 90 degrees to get an orthogonal coverage.

PACS scan maps can be performed either in the instrument reference frame or in sky coordinates.

The PACS Data

- Raw PACS data: compressed telemetry file

The PACS Data

- Raw PACS data: compressed telemetry file
- Simple processing (on-board averaging of 4 frames, final 10 Hz sampling) and compression of the data on-board. This enable us to downlink more science data than it would be possible with the current down-link telemetry rate

The PACS Data

- Raw PACS data: compressed telemetry file
- Simple processing (on-board averaging of 4 frames, final 10 Hz sampling) and compression of the data on-board. This enable us to downlink more science data than it would be possible with the current down-link telemetry rate
- Decompressed PACS Frames. This dataset contains for the different detectors in one camera (photoconductor array or bolometer camera) all signals for subsequent integration intervals.

Product Definition

There is a Herschel-wide convention on processing levels of the different instruments.

- *Raw Telemetry* : All telemetry packets produced by the instrument in the course of the observation. In PACS IA, we store/manipulate this level as a *PacketSequence* .

Product Definition

There is a Herschel-wide convention on processing levels of the different instruments.

- *Raw Telemetry* : All telemetry packets produced by the instrument in the course of the observation. In PACS IA, we store/manipulate this level as a *PacketSequence* .

- *Level 0 data:*

  Level 0 data is a complete set of data requested to do the scientific data reduction. It is saved in a Level 0 Data Pool in form of Fits files. After Level 0 data generation no connection to the Database is possible any more. Therefore Level 0 data need to contain all information needed from the Database (e.g. uplink information).

  - Science data

    Science data are organized in user friendly classes. The `Frames` class for reduced data and the `PhotRaw` class for additional raw channel data will be the basic data products for this processing steps.

  - Axillary data

    Auxiliary data for the time span covered by the Level 0 data, such as the spacecraft pointing (attitude history), the time correllation, selected spacecraft housekeeping, etc

    The information is partly merged as status entries into the basic science classes Frames and PhotRaw or available as Products (Pointing)

  - Decoded HK Data

    HK data Tables with converted and raw HK values.

+ Calibration files

Product Definition

There is a Herschel-wide convention on processing levels of the different instruments.

- *Raw Telemetry* : All telemetry packets produced by the instrument in the course of the observation. In PACS IA, we store/manipulate this level as a *PacketSequence* .

- *Level 0 data*:

- *Level 0.5 data* :

Processing until Level 0.5 is AOT independent These data are saved in the Product Pool.

On this Level additional information is added to the Frames class (Flags for Saturation, Flags Bad Pixel, BlockTable,...) and basic unit conversion are done (digital values to Volts, chopper angle).

Product Definition

There is a Herschel-wide convention on processing levels of the different instruments.

- *Raw Telemetry* : All telemetry packets produced by the instrument in the course of the observation. In PACS IA, we store/manipulate this level as a *PacketSequence* .

- *Level 0 data:*

- *Level 0.5 data :*

- *Level 1 data:*

Level 1 data generation is AOT dependent. Level 1 data are saved in the Product Pool.

Detector readouts calibrated, converted to physical units and grouped into blocks. For PACS photometry this is a data cube with flux densities with associated sky coordinates. Mostly every step before actual Image construction is done.

The `Frames or FramesStack` class will be the basic Level 1 product of photometer data

Possibly the Level 1 data generation can be done automatically to a large extend after the instrument has been calibrated.

Product Definition

There is a Herschel-wide convention on processing levels of the different instruments.

- *Raw Telemetry* : All telemetry packets produced by the instrument in the course of the observation. In PACS IA, we store/manipulate this level as a *PacketSequence* .

- *Level 0 data:*

- *Level 0.5 data :*

- *Level 1 data:*

- *Level 2 data:*

  Further processed level-1 data to such a level that scientific analysis can be performed.

  The actual Image construction is done in this step. It is highly AOT dependent and may involve software like MadMap.

  For optimal results many of the processing steps involved to generate level-2 data may require human interaction, based both on instrument understanding as well as understanding of the scientific aims of the observation.

  The result is an Image Product.

Product Definition

There is a Herschel-wide convention on processing levels of the different instruments.

- *Raw Telemetry* : All telemetry packets produced by the instrument in the course of the observation. In PACS IA, we store/manipulate this level as a *PacketSequence* .

- *Level 0 data:*

- *Level 0.5 data :*

- *Level 1 data:*

- *Level 2 data:*

- *Level 3 data:*

  These are the publishable science products where level-2 data products are used as input. These products are not only from the specific instrument, but are usually combined with theoretical models, other observations, laboratory data, catalogues, etc. Their formats should be VO compatible and these data products should be suitable for VO access.

The hipe java-environment

- *Brand new java environment*
  - new plot
  - new display
  - new libraries
- *advantages & disadvantages:*
  - customized for non-standard data reduction
  - so far tested only by Herschel calibration-astronomers and not by community
  - to go beyond pipeline scripts, astronomers have to get used to Object-oriented coding, Pools, Classes, Products, ArrayDatasets, TableDatasets, etc

PACS pipeline & hipe

The PACS Photometer Pipeline is composed of Tasks:

- most of the tasks are written in Java
- several are still Jython prototypes

PACS pipeline & hipe

The P… of
Tasks
- mos…
- sev…

```
lstore = LocalStoreFactory.getStore("sovt1")
store  = ProductStorage()
store.register(lstore)
result = browseProduct(store)
obs=result[0].product
frames=obs.level0.refs["HPPAVGB"].product.refs[0].product
pp=obs.auxiliary.pointing
calTree = getCalTree("FM", "BASE")
frames = findBlocks(frames)
frames=photAddInstantPointing(frames,pp)
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec=PlotXY(ra,dec,titleText="Position (of Herschel? optical axis?) on sky",xtitle="RA [deg]",ytitle="DEC [deg]")
frames = frames.select( frames.status["ScanLineNumber"].data < 8)
frames = frames.select( frames.status["CALSOURCE"].data == 0)
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec.addLayer(LayerXY(ra,dec))
frames = photFlagBadPixels(frames)
frames= photFlagSaturation(frames)
frames = photConvDigit2Volts(frames)
frames = photCorrectCrosstalk(frames)
frames = convertChopper2Angle(frames)
frames = photAssignRaDec(frames)

#################################################################################################
############### Add few sources to the real data ################################################
print "Now starting to add artificial sources"
rs=158.9
ds=-16.48
sigs=0.001
ps=1.e-4
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.95
ds=-16.54
sigs=0.001
ps=3.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.85
ds=-16.50
sigs=0.001
ps=1.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
#################################################################################################



frames = filterSlew(frames)
frames = photHighpassFilter(frames,200)
map = photProject(frames)
```

How a pipeline script looks like

PACS pipeline & hipe

The PACS Photometer Pipeline is composed of Tasks:

- most of the tasks are written in Java
- several are still Jython prototypes

Pipeline starting point

- Tar ball with all chosen Pipeline Products (Level 0, 1, 2) and data associated to observation (Cal files, Pointing products, etc)
- Data organized in the ***Observation Context***
- Local pool (storage) on the disk

The Observation Context

```
lstore = LocalStoreFactory.getStore("sovt1")
store  = ProductStorage()
store.register(lstore)
result = browseProduct(store)
obs=result[0].product
frames=obs.level0.refs["HPPAVGB"].product.refs[0].product
pp=obs.auxiliary.pointing
calTree = getCalTree("FM", "BASE")
frames = findBlocks(frames)
frames=photAddInstantPointing(frames,pp)
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec=PlotXY(ra,dec,titleText="Position (of Herschel? optical axis?) on sky",xtitle="RA [deg]",ytitle="DEC [deg]")
frames = frames.select( frames.status["ScanLineNumber"].data < 8)
frames = frames.select( frames.status["CALSOURCE"].data == 0)
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec.addLayer(LayerXY(ra,dec))
frames = photFlagBadPixels(frames)
frames= photFlagSaturation(frames)
frames = photConvDigit2Volts(frames)
frames = photCorrectCrosstalk(frames)
frames = convertChopper2Angle(frames)
frames = photAssignRaDec(frames)

##################################################################################################
############### Add few sources to the real data ###########################################
print "Now starting to add artificial sources"
rs=158.9
ds=-16.48
sigs=0.001
ps=1.e-4
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.95
ds=-16.54
sigs=0.001
ps=3.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.85
ds=-16.50
sigs=0.001
ps=1.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
##################################################################################################



frames = filterSlew(frames)
frames = photHighpassFilter(frames,200)
map = photProject(frames)
```

The Observation Context

```
lstore = LocalStoreFactory.getStore("sovt1")
store  = ProductStorage()
store.register(lstore)
result = browseProduct(store)
obs=result[0].product
frames=obs.level0.refs["HPPAVGB"].product.refs[0].product
pp=obs.auxiliary.pointing
```

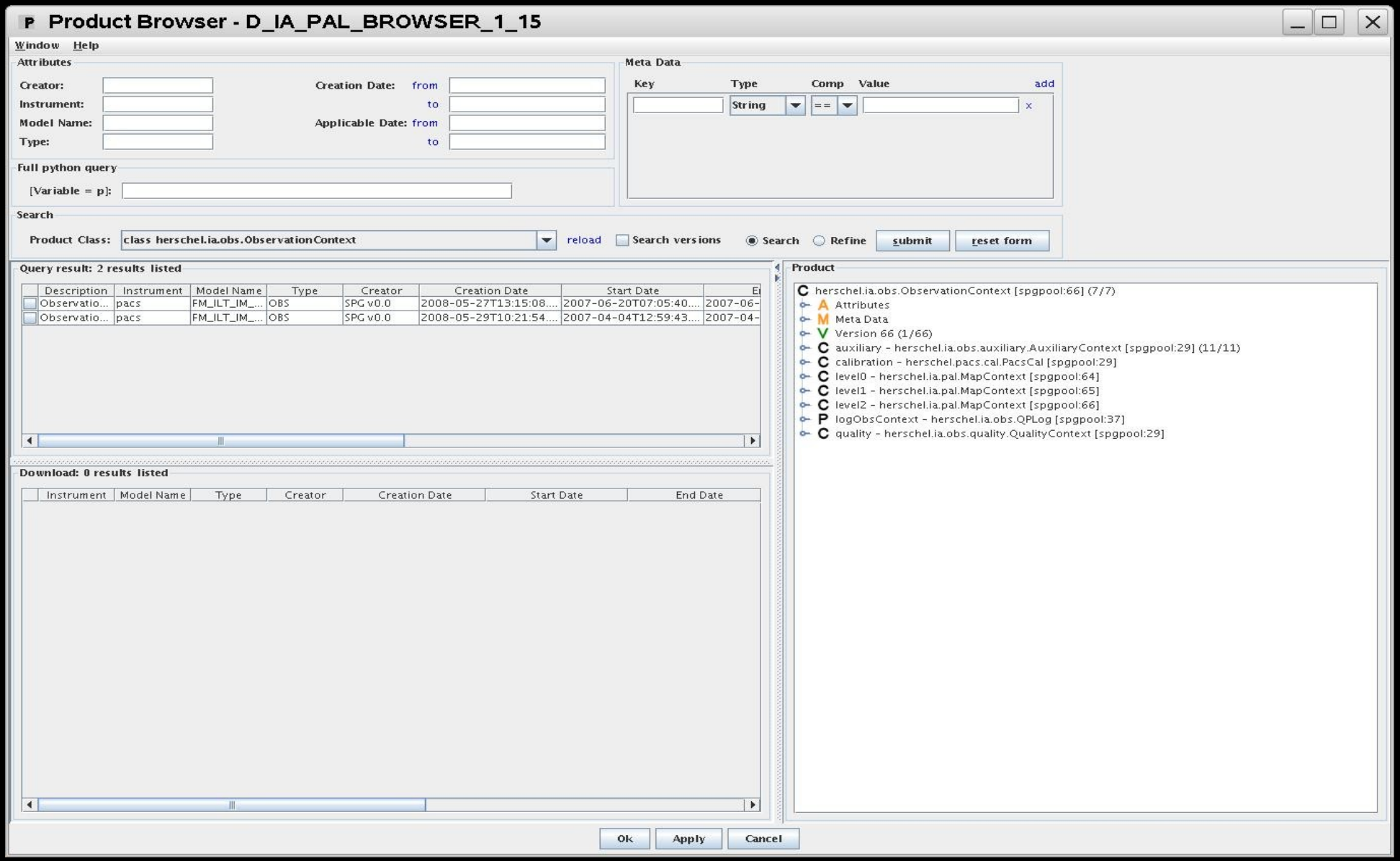
```
frames = findBlocks(frames)
frames=photAddInstantPointing(fr
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec=PlotXY(ra,dec,titleTex
frames = frames.select( frames.s
frames = frames.select( frames.s
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec.addLayer(LayerXY(ra,de
frames = photFlagBadPixels(frame
frames= photFlagSaturation(frame
frames = photConvDigitsToVolts(fra
frames = photCorrectCrosstalk(fr
frames = convertChopper2Angle(fr
frames = photAssignRaDec(frames)

################################################################################
############### Add few sources to the real data ###############################
print "Now starting to add artificial sources"
rs=158.9
ds=-16.48
sigs=0.001
ps=1.e-4
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.95
ds=-16.54
sigs=0.001
ps=3.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.85
ds=-16.50
sigs=0.001
ps=1.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
################################################################################################



frames = filterSlew(frames)
frames = photHighpassFilter(frames,200)
map = photProject(frames)
```

The Observation Context

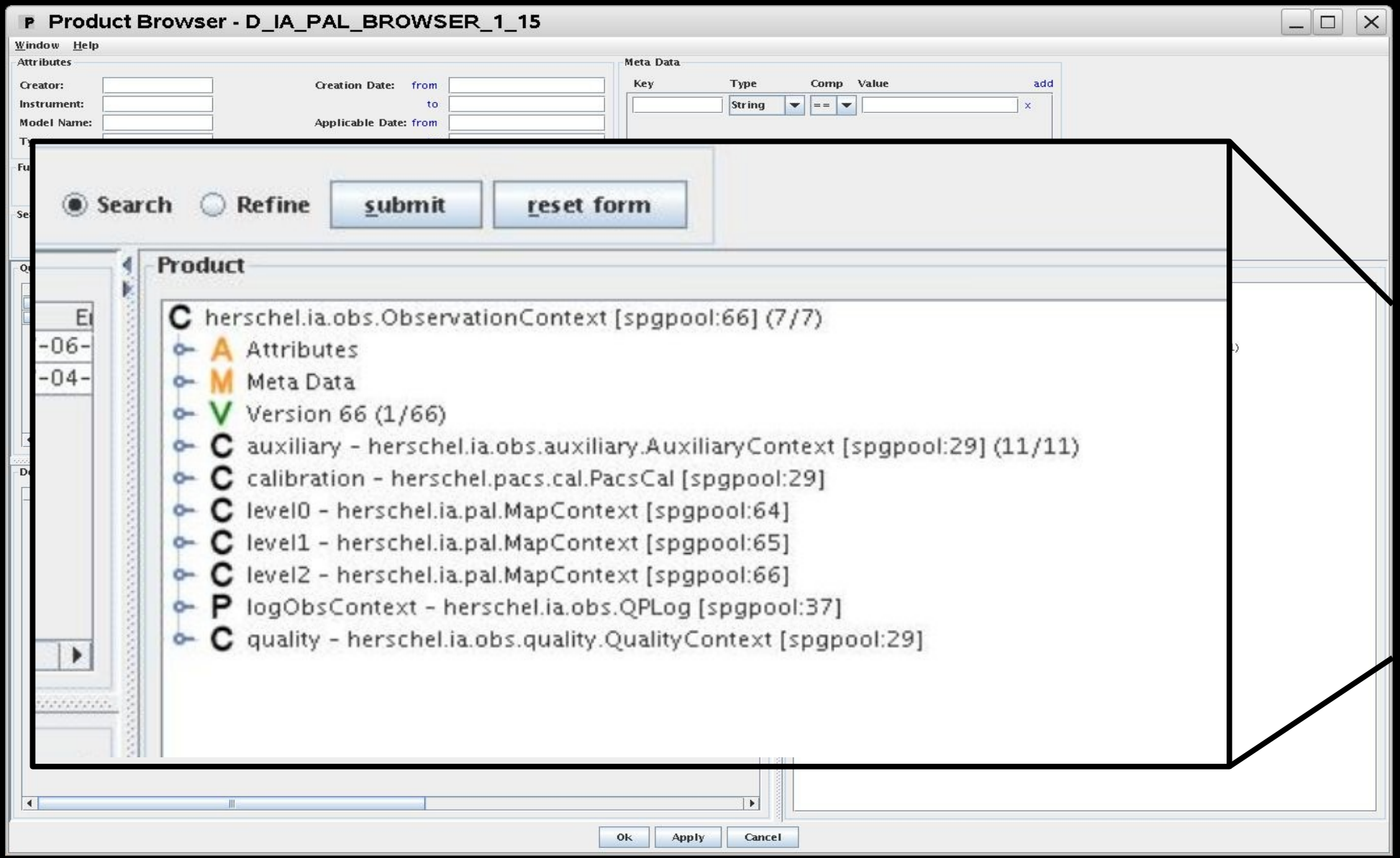
```
lstore = LocalStoreFactory.getStore("sovt1")
store  = ProductStorage()
store.register(lstore)
result = browseProduct(store)
obs=result[0].product
frames=obs.level0.refs["HPPAVGB"].product.refs[0].product
pp=obs.auxiliary.pointing
```

```
frames = findBlocks(frames)
frames=photAddInstantPointing(fr
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec=PlotXY(ra,dec,titleTex
frames = frames.select( frames.s
frames = frames.select( frames.s
ra=frames.getStatus("RaArray")
dec=frames.getStatus("DecArray")
plotradec.addLayer(LayerXY(ra,de
frames = photFlagBadPixels(frame
frames= photFlagSaturation(frame
frames = photConvDigitsToVolts(fra
frames = photCorrectCrosstalk(fr
frames = convertChopper2Angle(fr
frames = photAssignRaDec(frames)

################################################################################
############### Add few sources to the real data ###############################
print "Now starting to add artificial sources"
rs=158.9
ds=-16.48
sigs=0.001
ps=1.e-4
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.95
ds=-16.54
sigs=0.001
ps=3.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
rs=158.85
ds=-16.50
sigs=0.001
ps=1.e-5
frames.signal=frames.signal+ps*\
 EXP(-(((frames.ra-rs)*COS(3.14159*ds/180.)/sigs)**2+((frames.dec-ds)/sigs)**2)/2.)
################################################################################################



frames = filterSlew(frames)
frames = photHighpassFilter(frames,200)
map = photProject(frames)
```

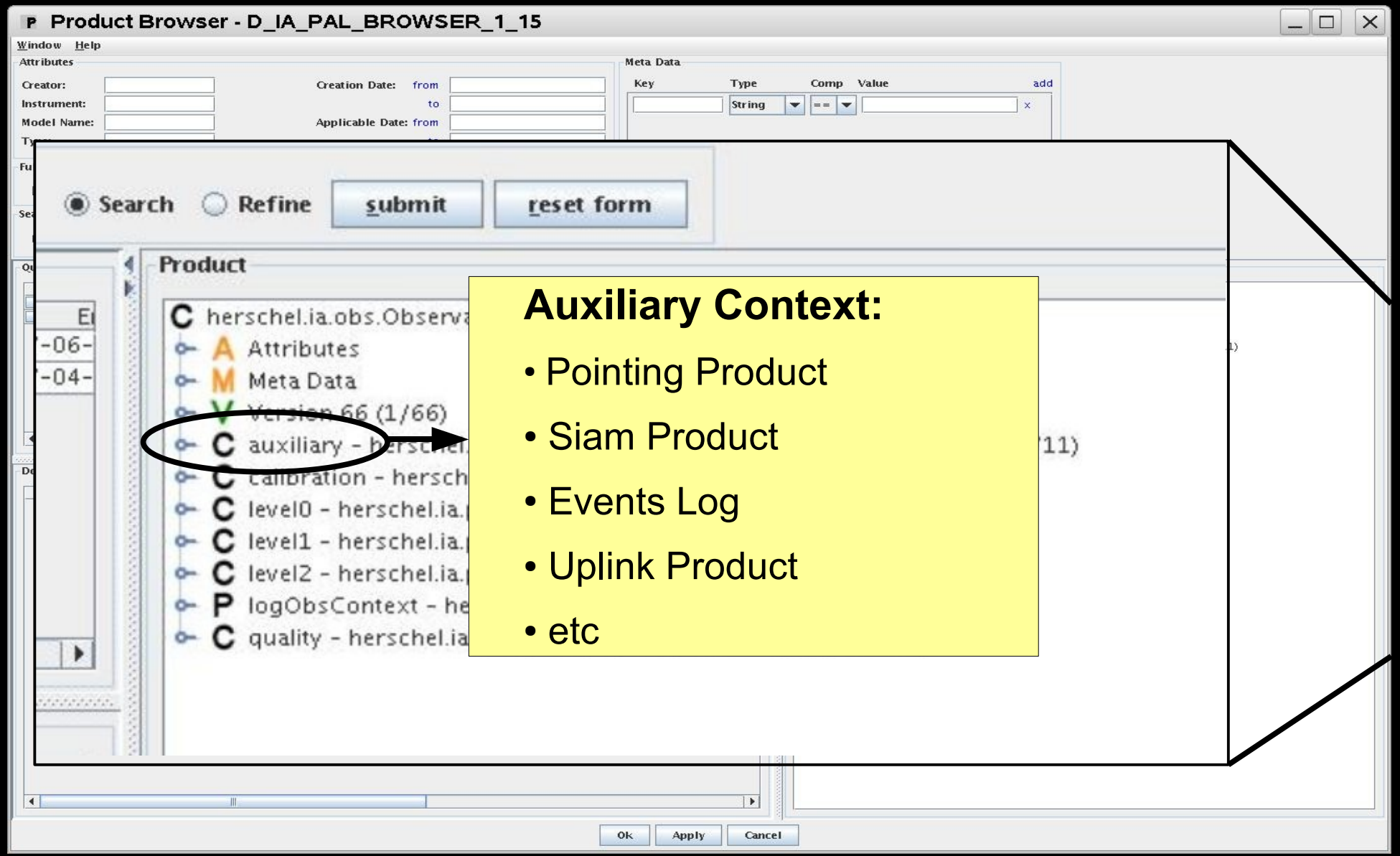
The Observation Context



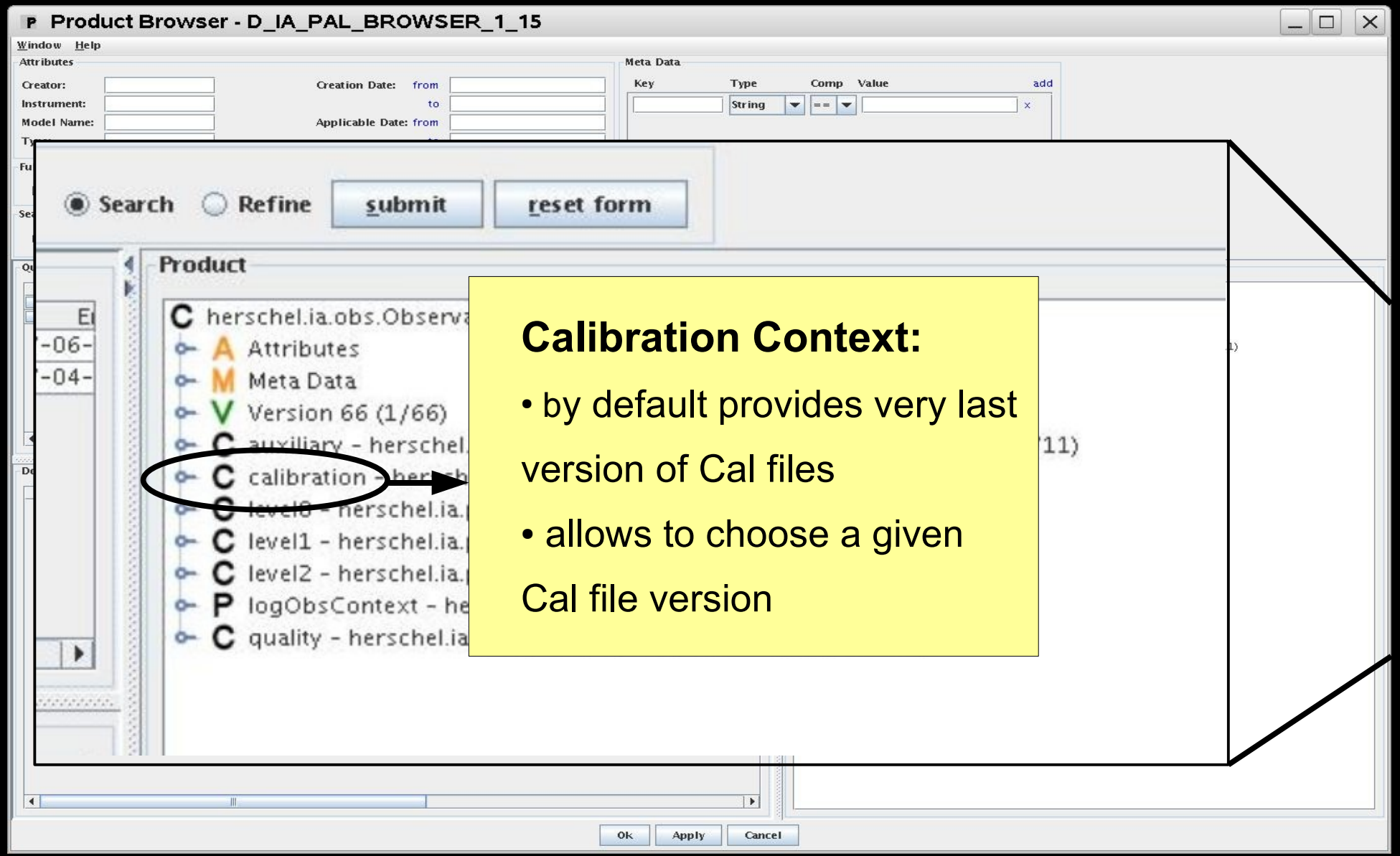
The Observation Context



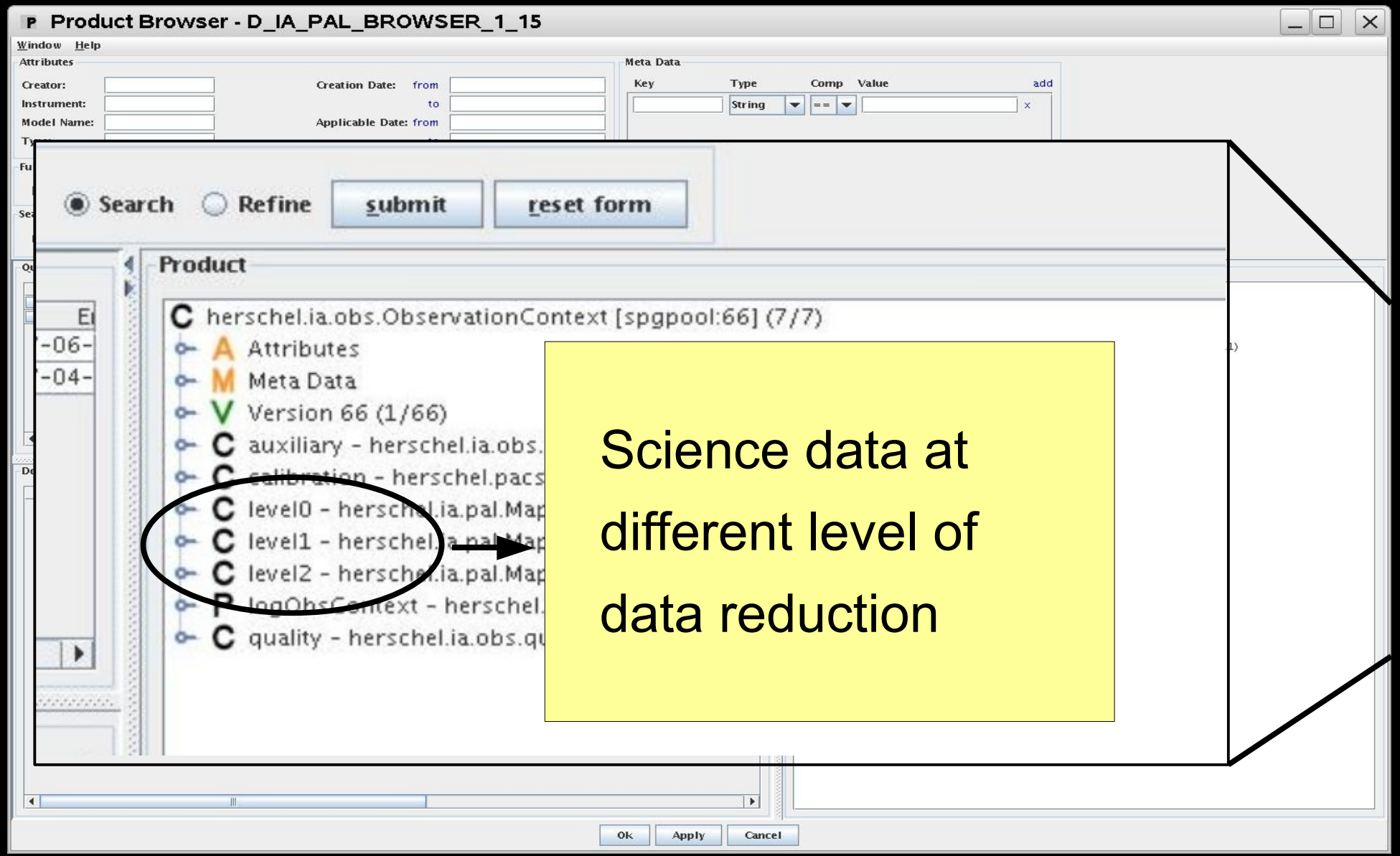
The Observation Context



The Observation Context



The Observation Context



Pacs Photometry Data Volume

Blue Bolometer: 32 x 64 = 2048 channels
Red Bolometer: 16 x 32 = 512 channels
Readout frequency: 40Hz averaged on board to 10Hz
All data stored as doubles: 8 bytes

data volume of Level 0 data:

- Blue bolometer: Signal → 2048 x 10 x 8 ~ 164 Kb/s
  Noise → 2048 x 10 x 8 ~ 164 Kb/s
  Ra → 2048 x 10 x 8 ~ 164 Kb/s
  Dec → 2048 x 10 x 8 ~ 164 Kb/s

Pacs Photometry Data Volume

Blue Bolometer: 32 x 64 = 2048 channels
Red Bolometer: 16 x 32 = 512 channels
Readout frequency: 40Hz averaged on board to 10Hz
All data stored as doubles: 8 bytes

data volume of Level 0 (Level 1) data:

- Blue bolometer: Signal → 2048 x 10 x 8 ~ 164 Kb/s
  Noise → 2048 x 10 x 8 ~ 164 Kb/s
  Ra → 2048 x 10 x 8 ~ 164 Kb/s
  Dec → 2048 x 10 x 8 ~ 164 Kb/s

→ **~ 2.6 Gb/hour**

- Red Bolometer: → **~ 0.6 Gb/hour**

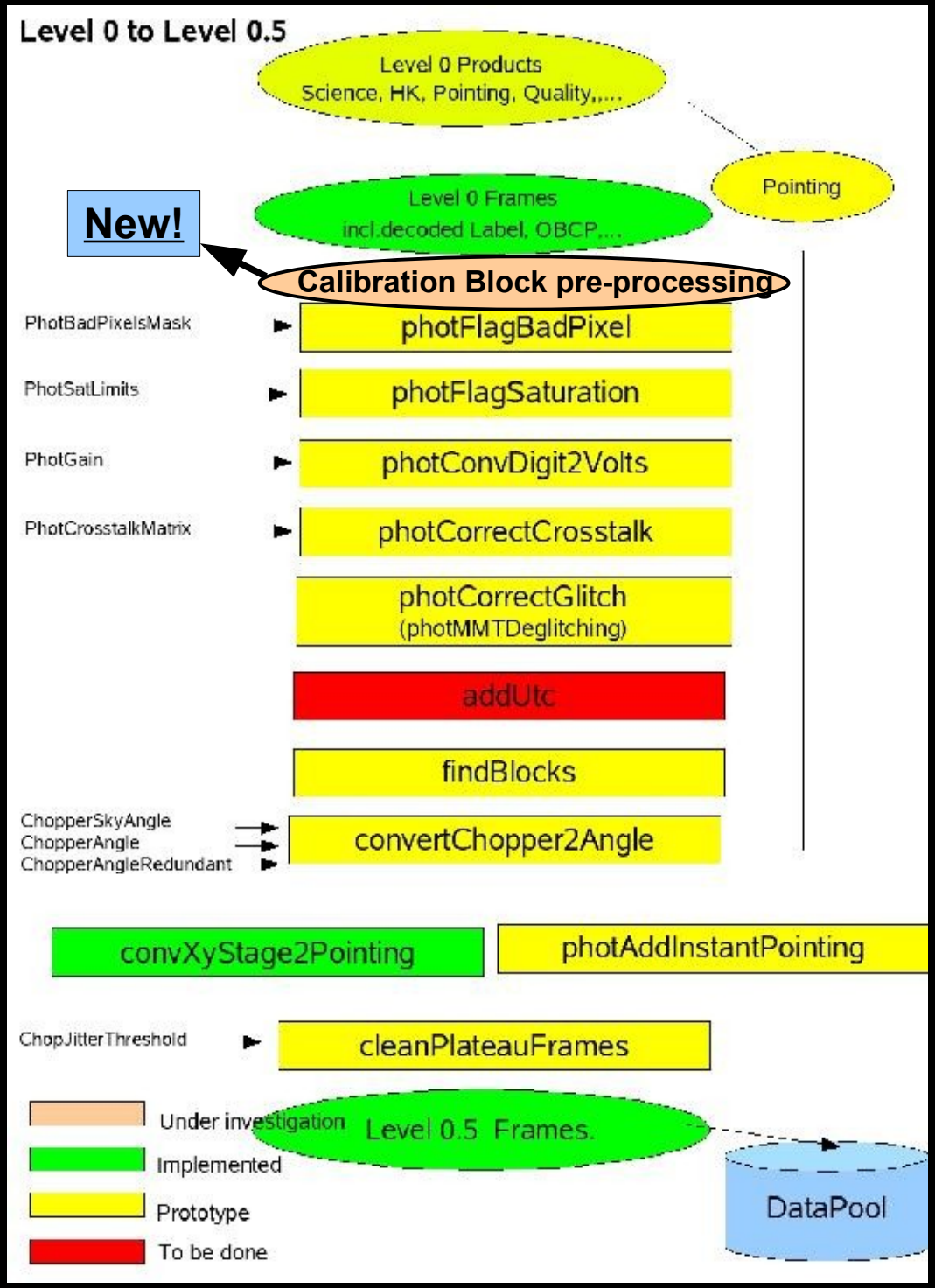
Most of AOT are much longer than 1h:
memory issue in the data processing!!!

Pacs Photometry Data Volume

Memory issue

Possible solutions:

- PACS photometry data processing feasible only on very powerful machines: at least 8G of ram
- Store Signal, Noise, Ra, Dec as float and not double
- Chunking of PACS photometry data (on the basis of repetition factor for point/small source aot, scan legs for scan maps aot)
- Clever memory interaction

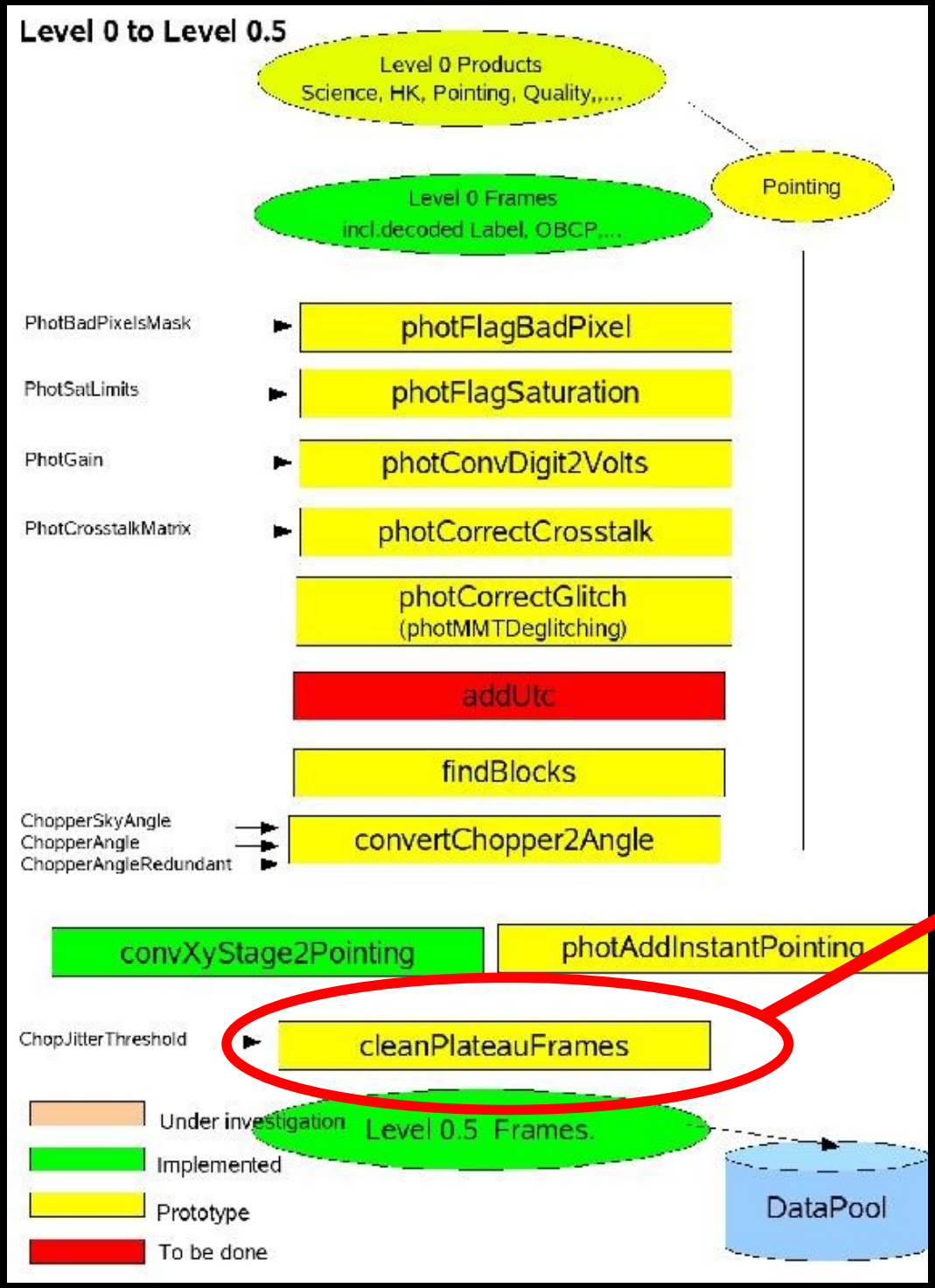


The Pipeline charts: from Level 0 to 0.5

- From Level 0 up tp 0.5 the pipeline is AOT independent

The Pipeline charts: from Level 0 to 0.5

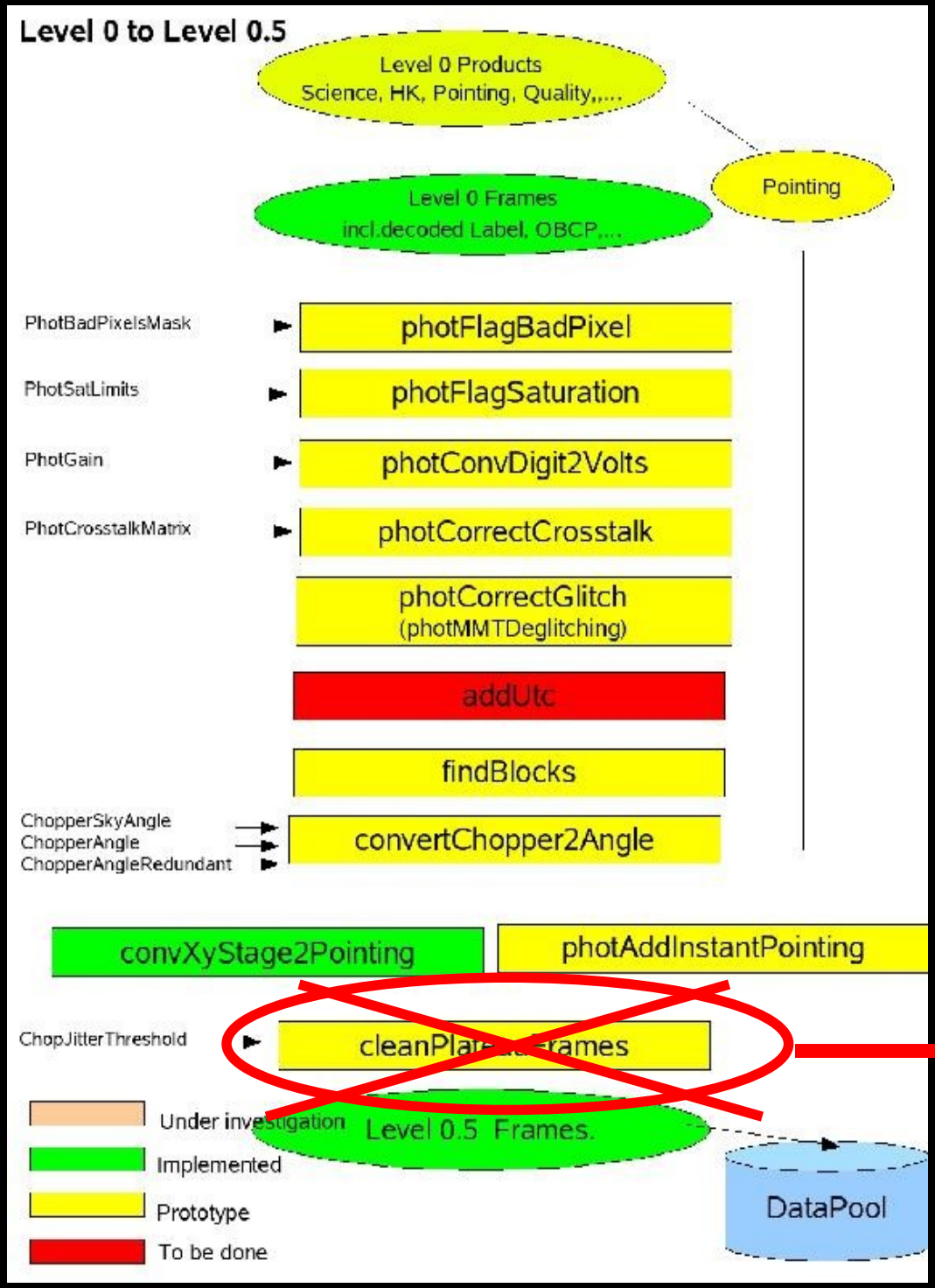
- From Level 0 up tp 0.5 the pipeline is AOT independent



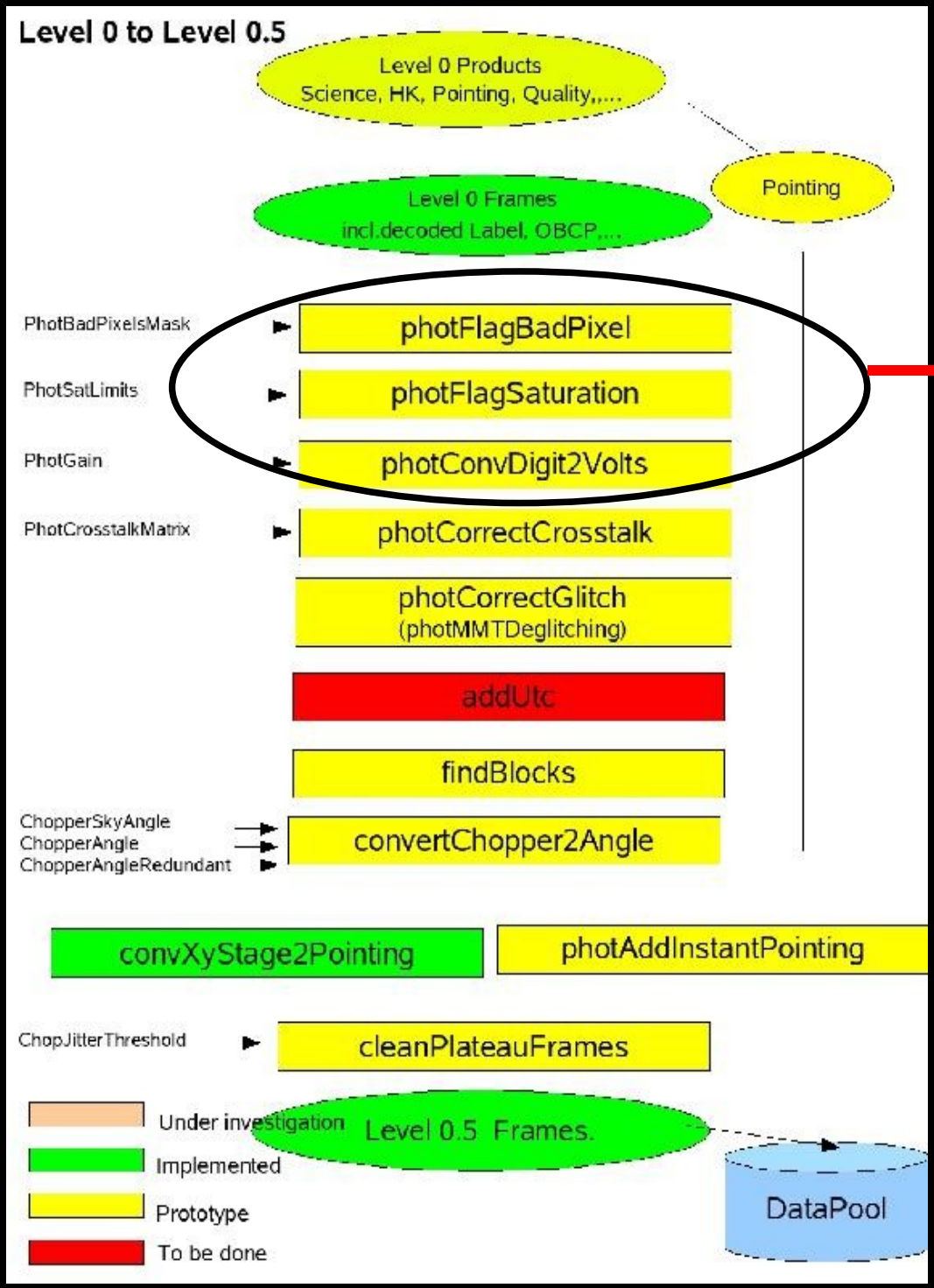
Only for Point Source, Small Source and Chopped Raster AOT

The Pipeline charts: from Level 0 to 0.5

- From Level 0 up tp 0.5 the pipeline is AOT independent



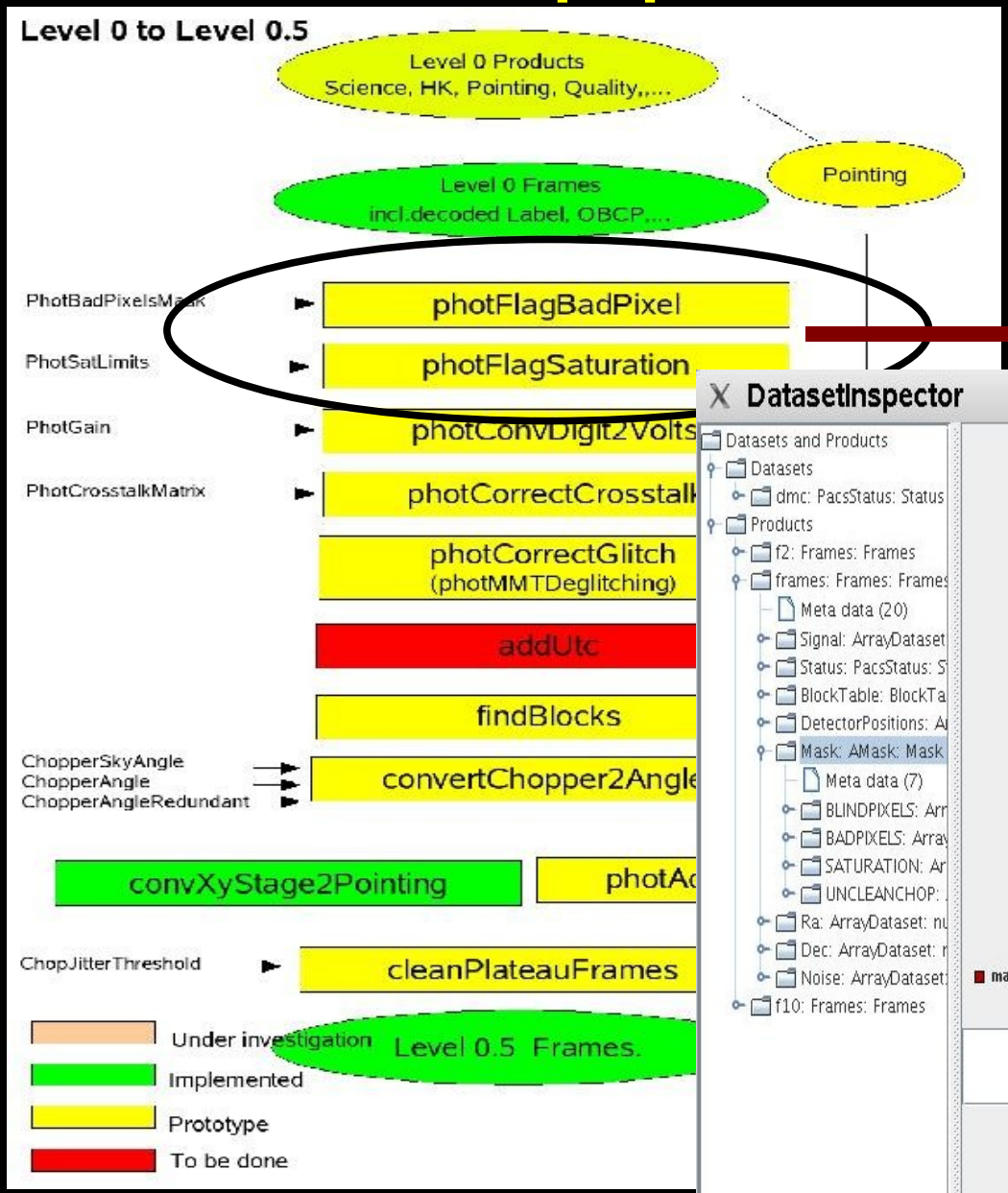
Not executed in Scan Map AOT



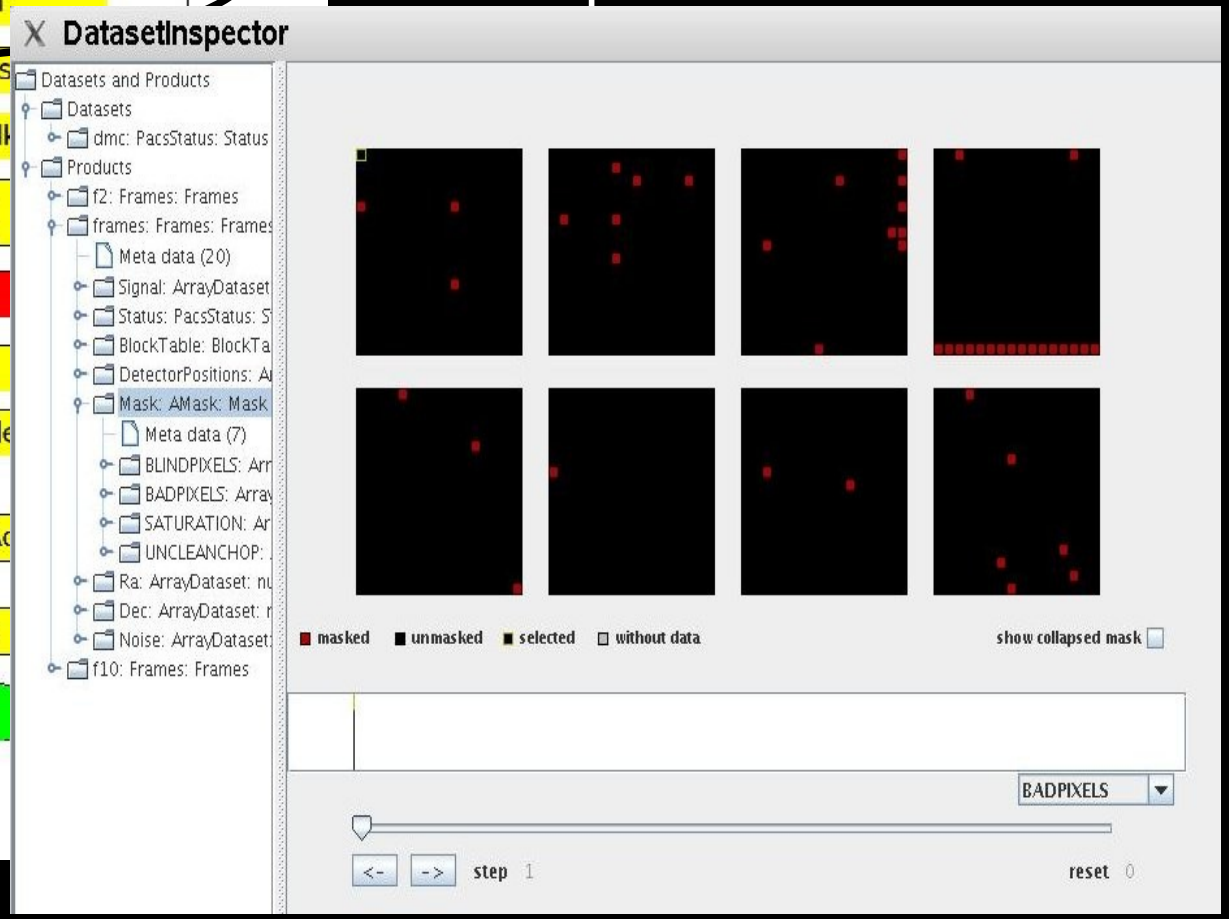
The pipeline step by step

- Mask creation and manipulation
- Conversion of digits in Volts

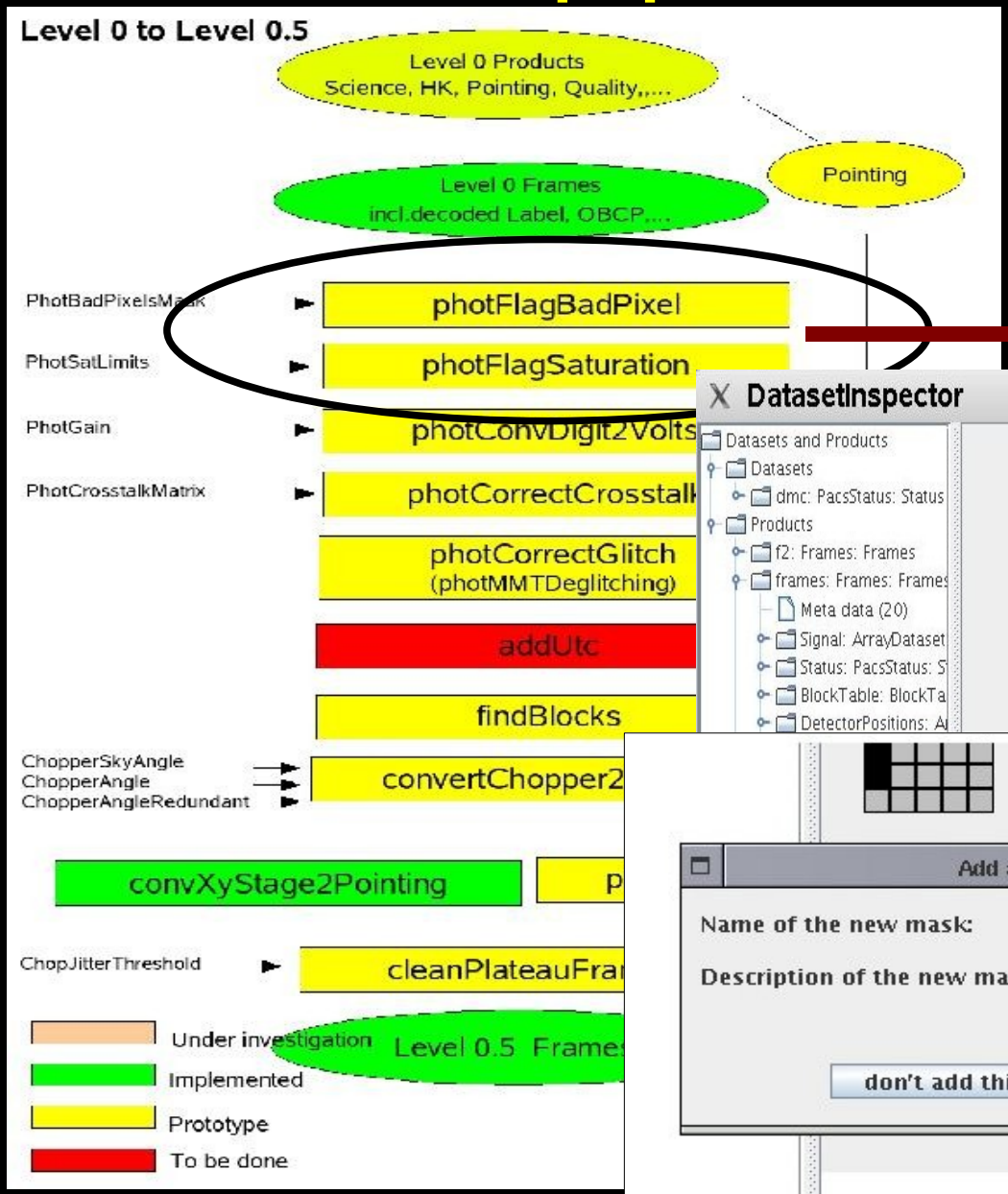
The pipeline step by step



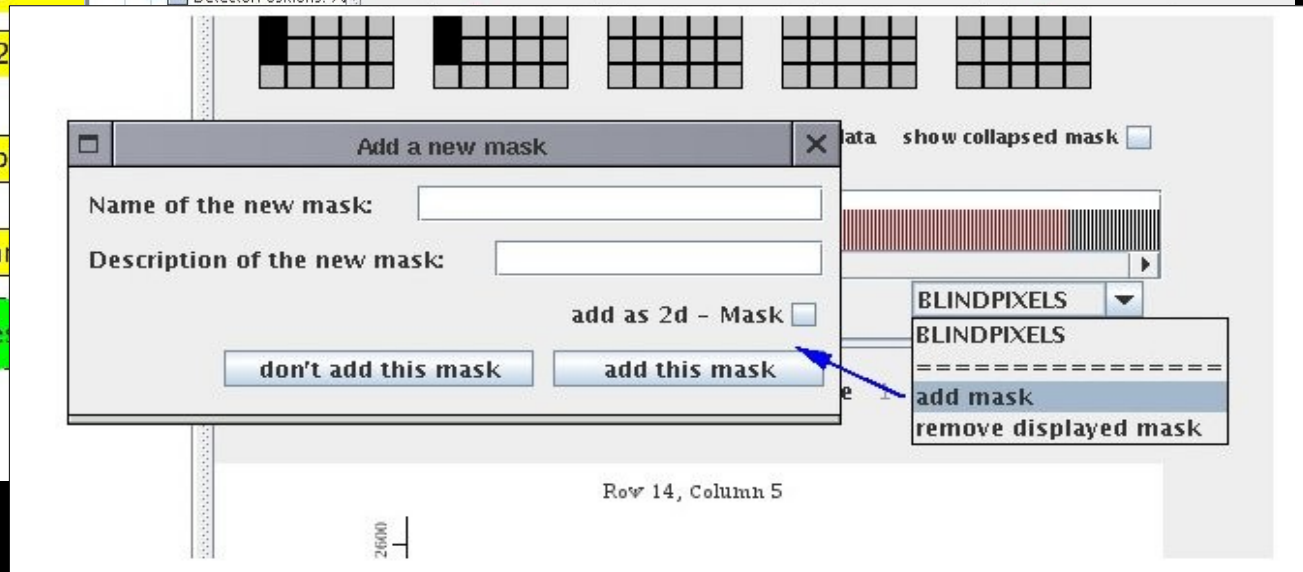
Mask creation and manipulation



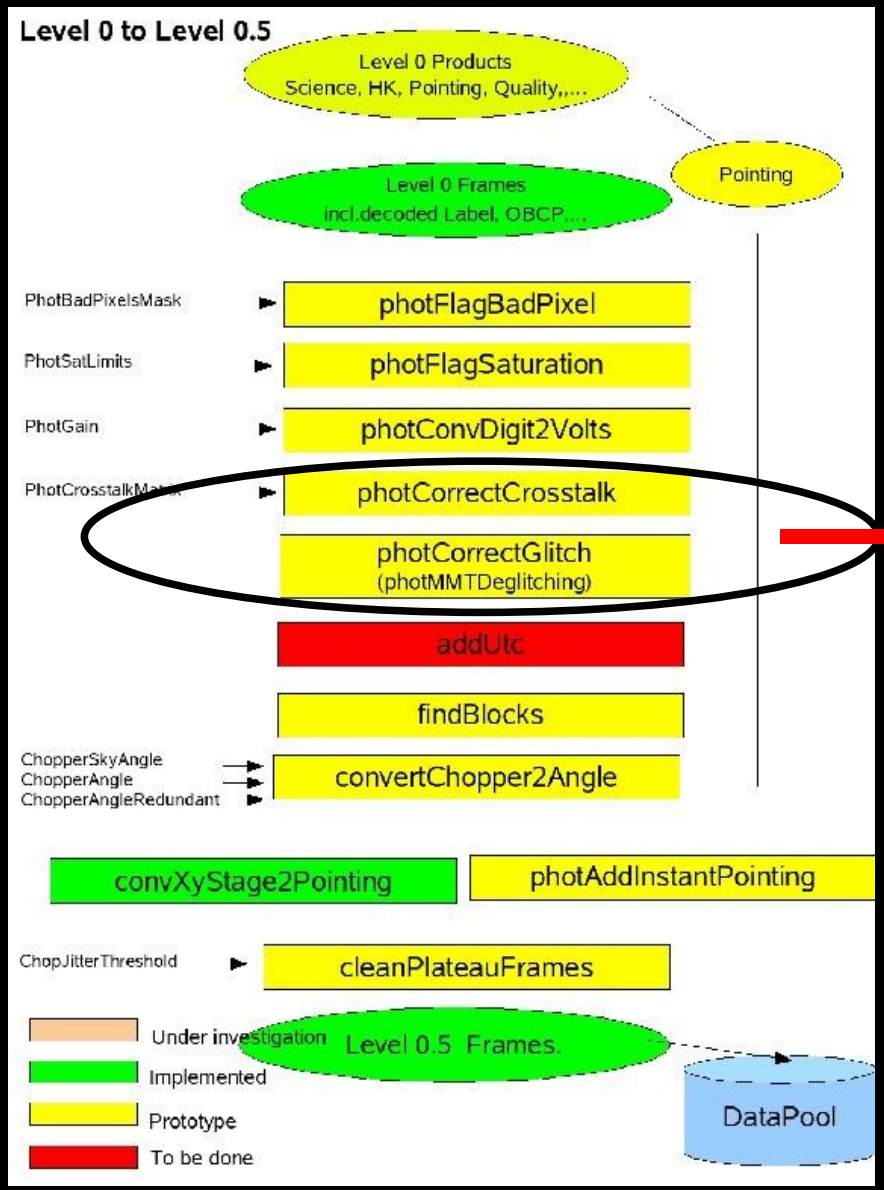
The pipeline step by step



Mask creation and manipulation

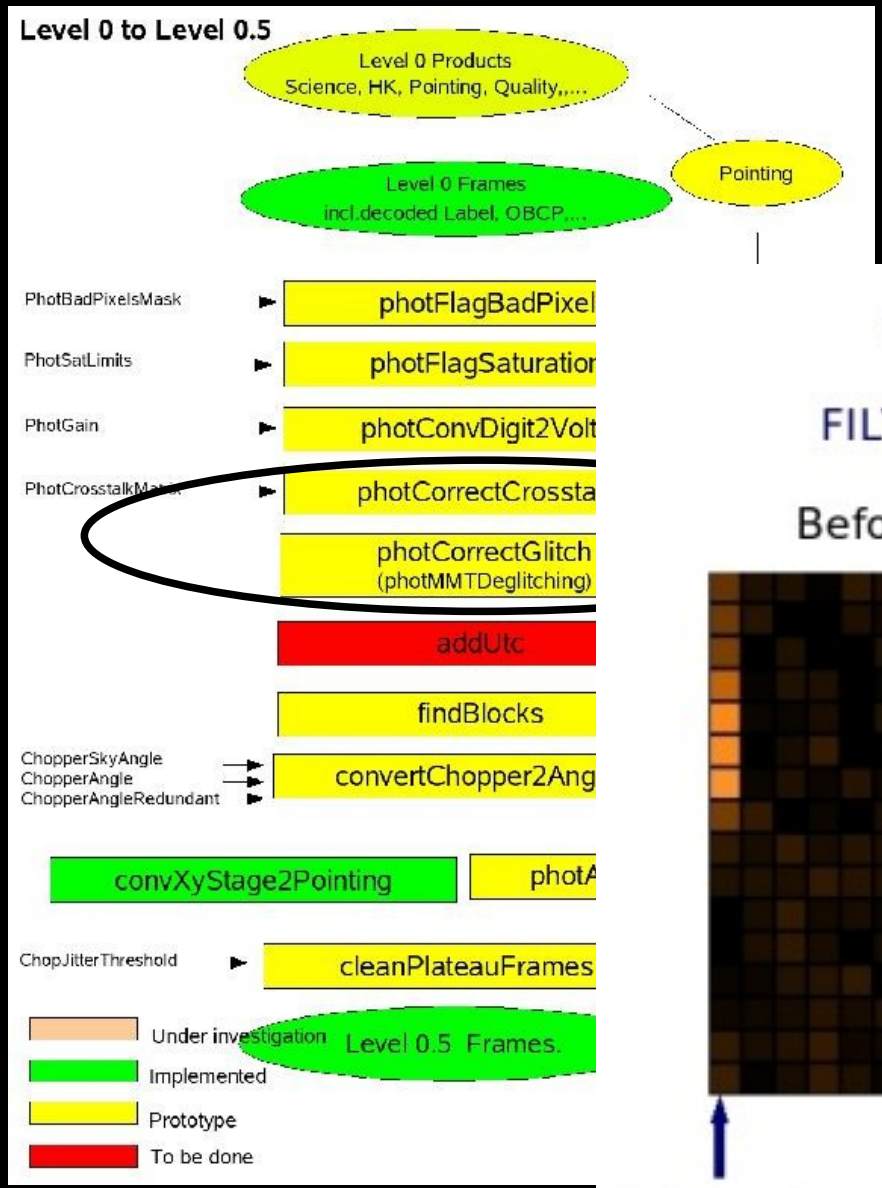


The pipeline Cosmetics

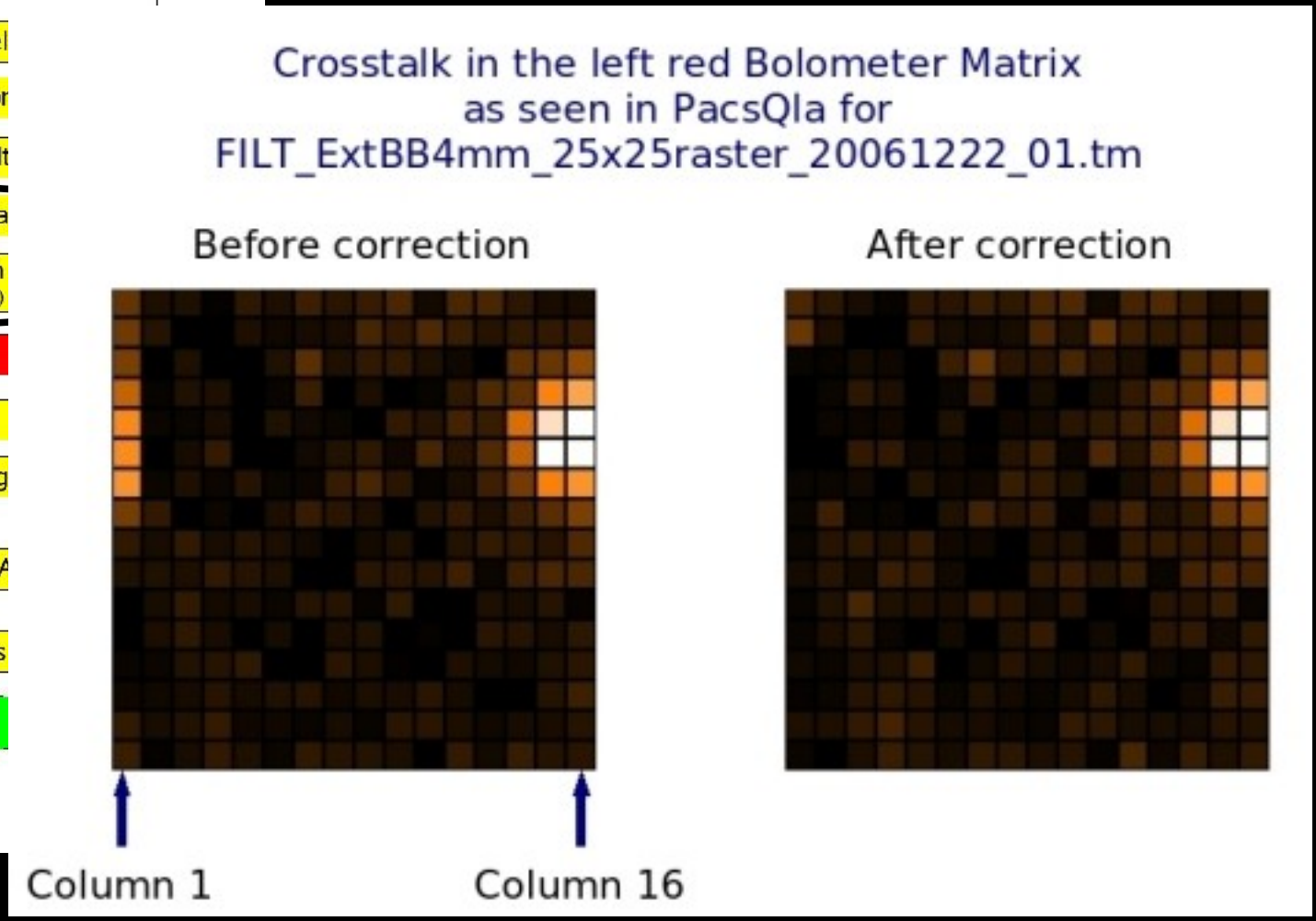


Manipulation of the signal: correction of cross-talk and glitches removal

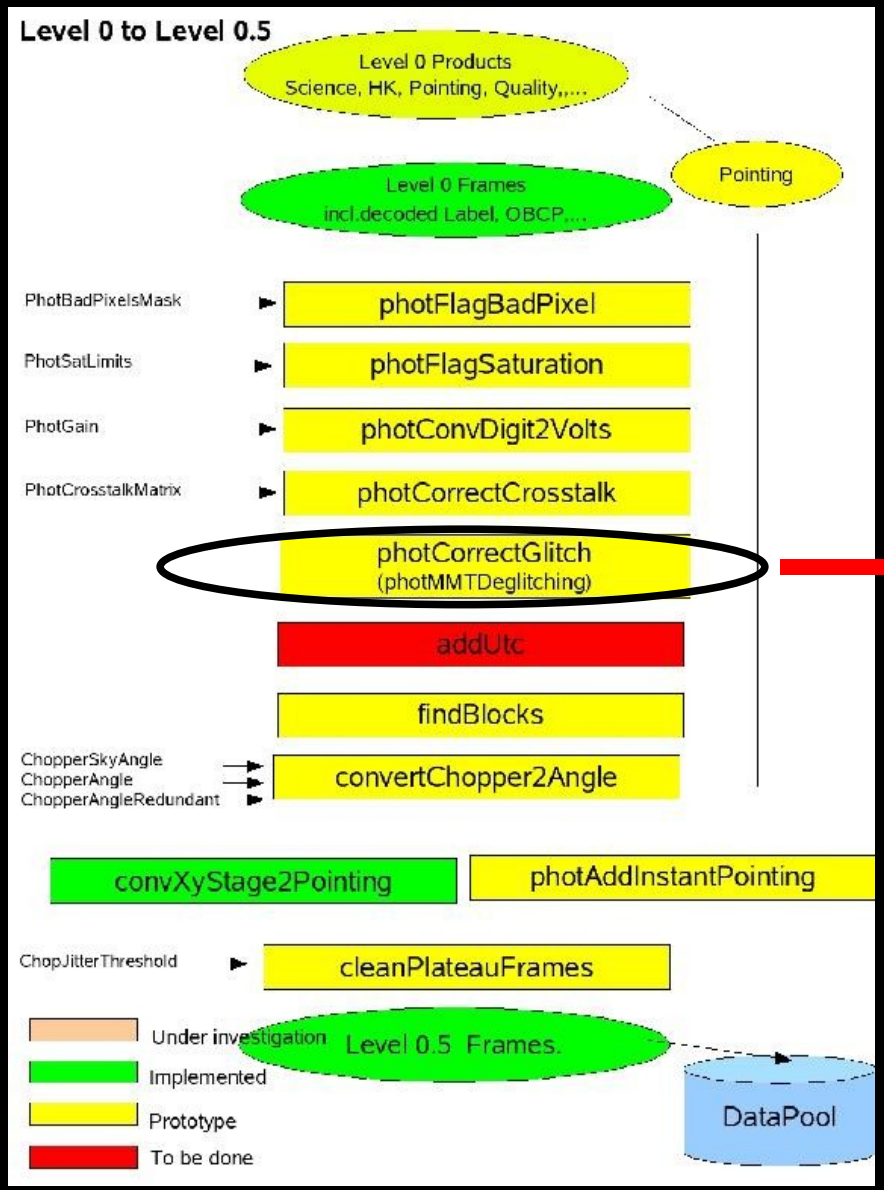
The pipeline Cross-talk correction



Cross-talk correction Based on calibration file (based on PACS test data)



The pipeline Deglitching



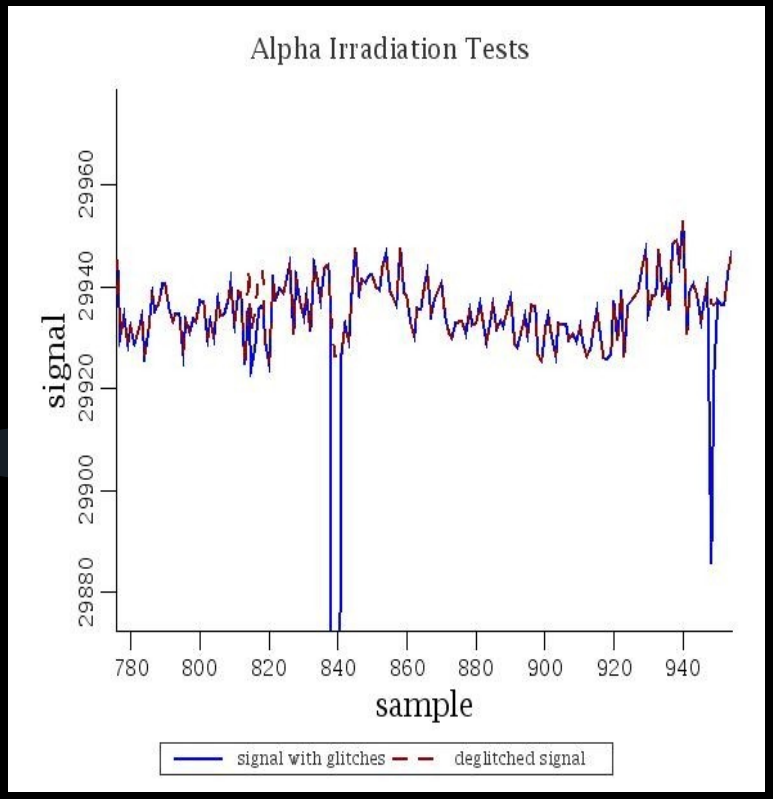
Deglitching based on the Multiresolution Median Transform

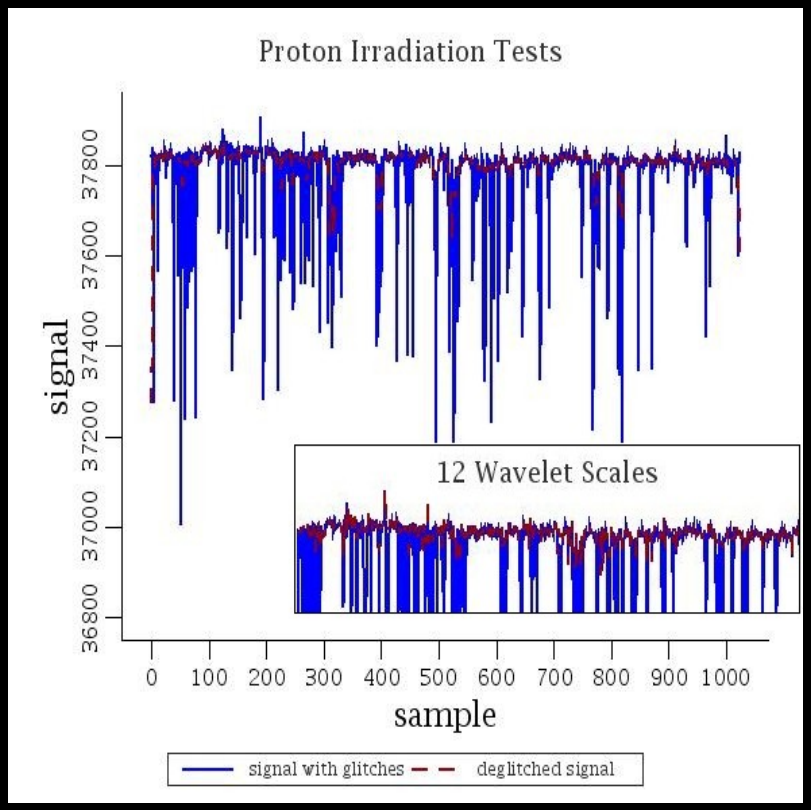
(Isocam Data Processing, Starck et al. 1999)

Method tested on alpha and proton irradiation

The pipeline Deglitching

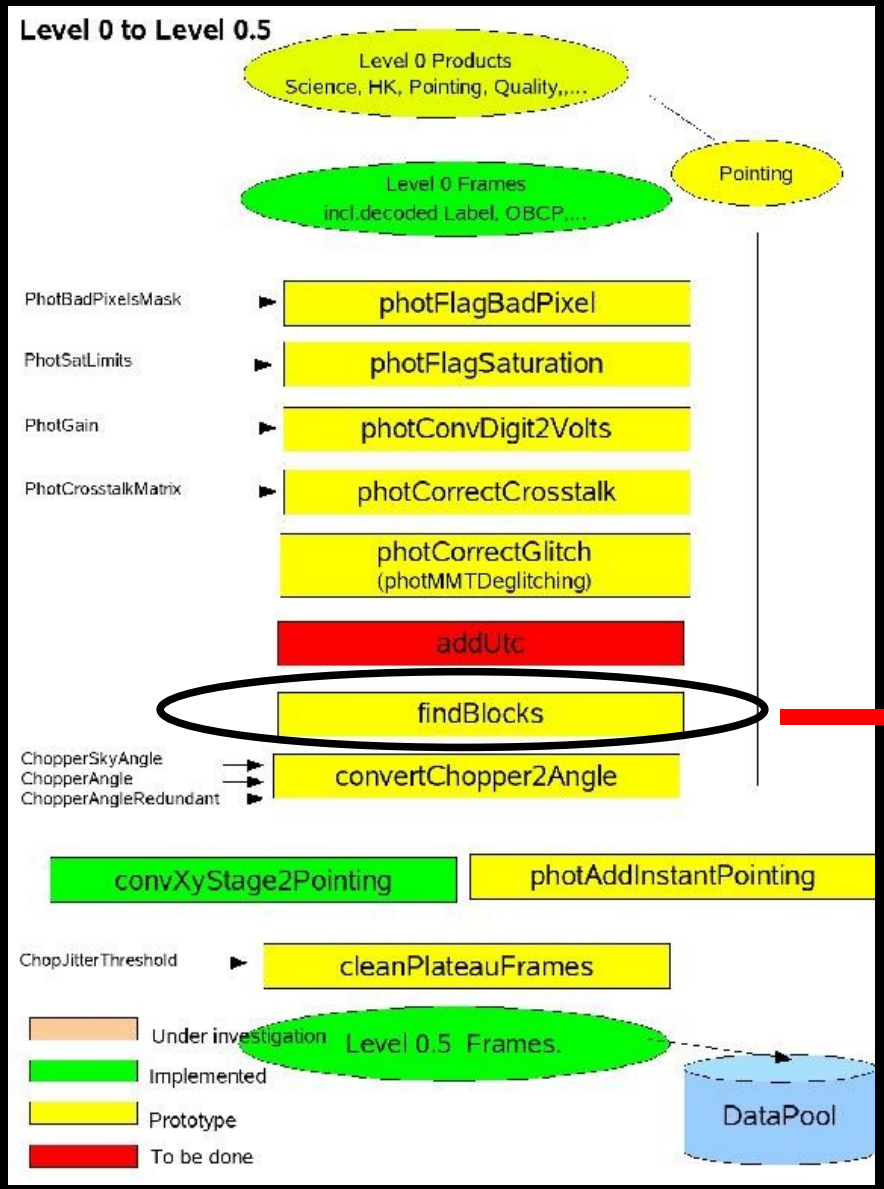
Multiresolution Median Transform





- The pixel signal can be corrected or masked out (up to user)
- Still to be done: error estimate of the deglitching correction
- Alternative method: *Wavelet Transform Modulus Maxima Lines Analysis* (status: implemented but not tested)

The data structure

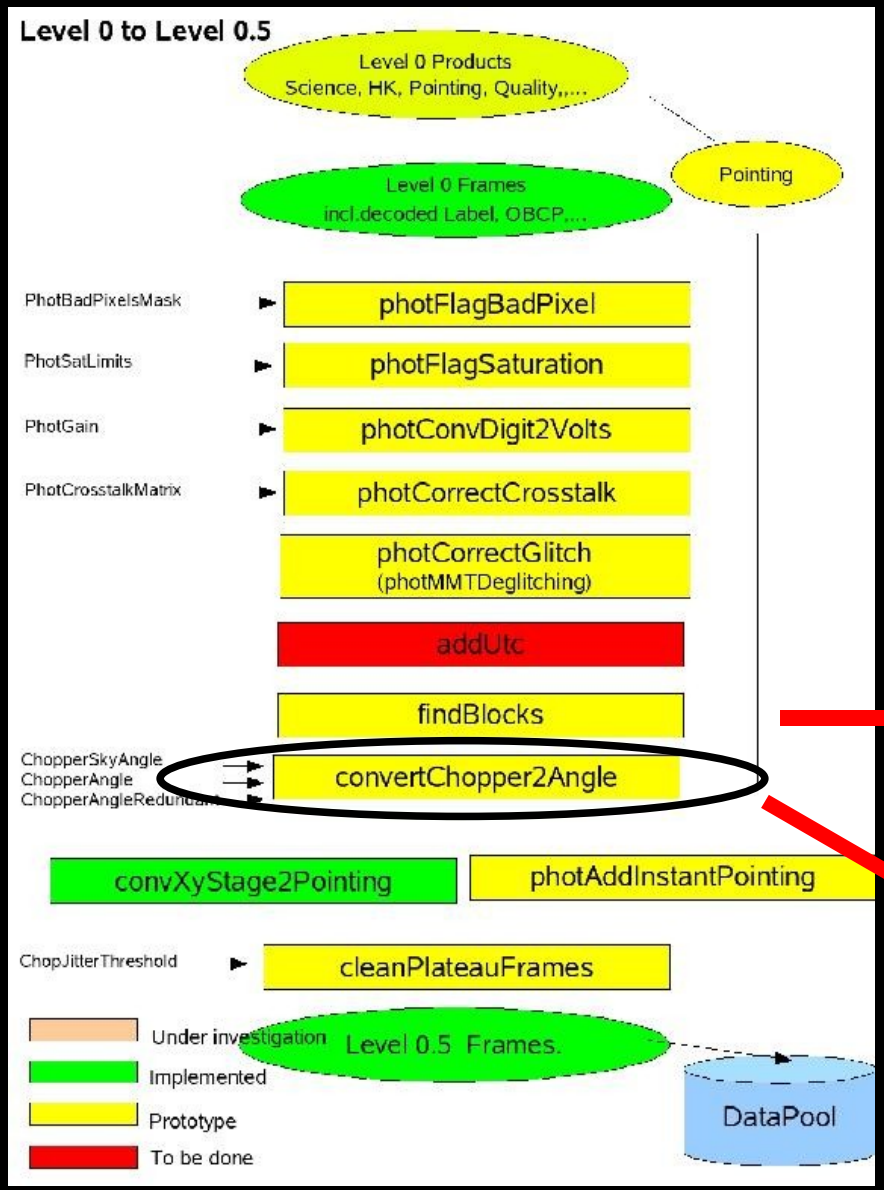


Block Table shows structure of observations through OBCP block.

OBCP block number identifies:

- Calibration block
- Science block
- Science data type: AOT mode

simple conversion...



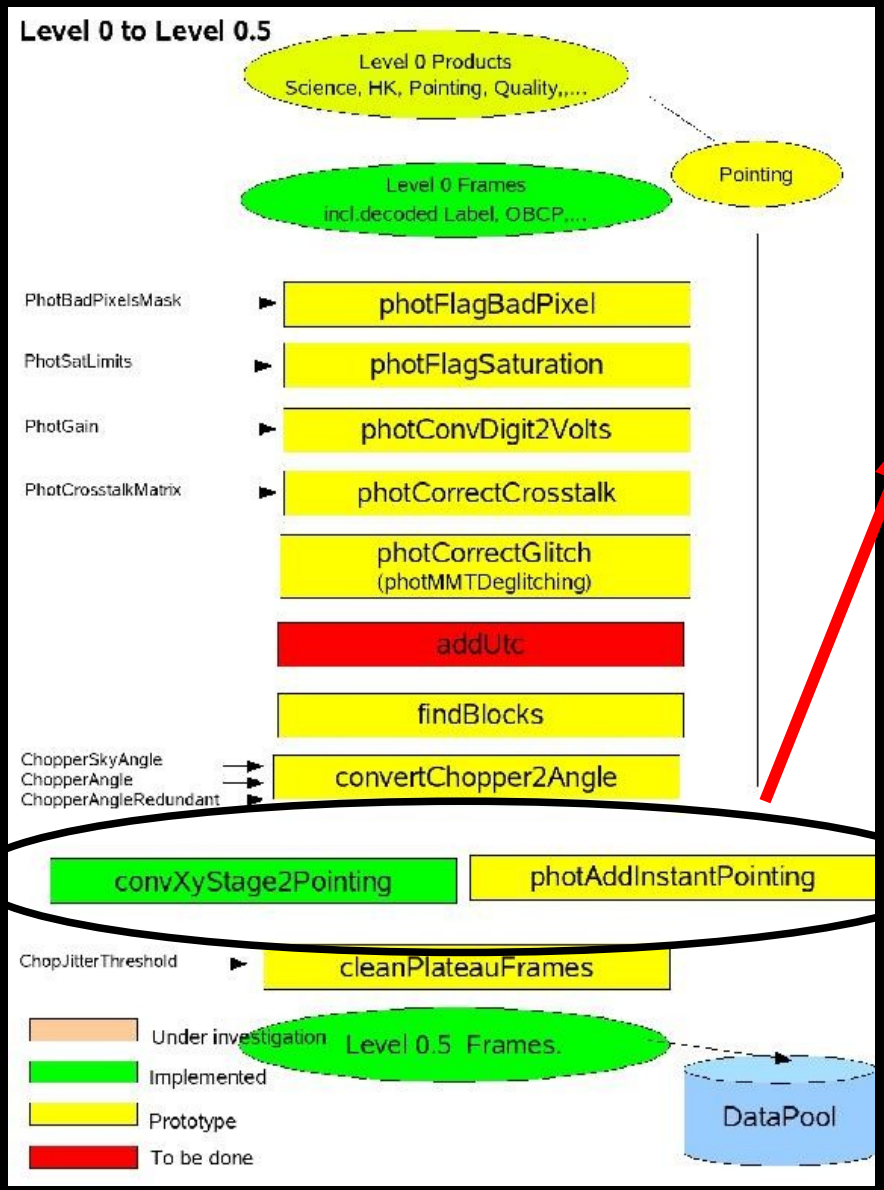
Block Table shows structure of observations through OBCP block.

OBCP block number identifies:

- Calibration block
- Science block
- Science data type: AOT mode

Conversion of chopper position technical unit in physical unit

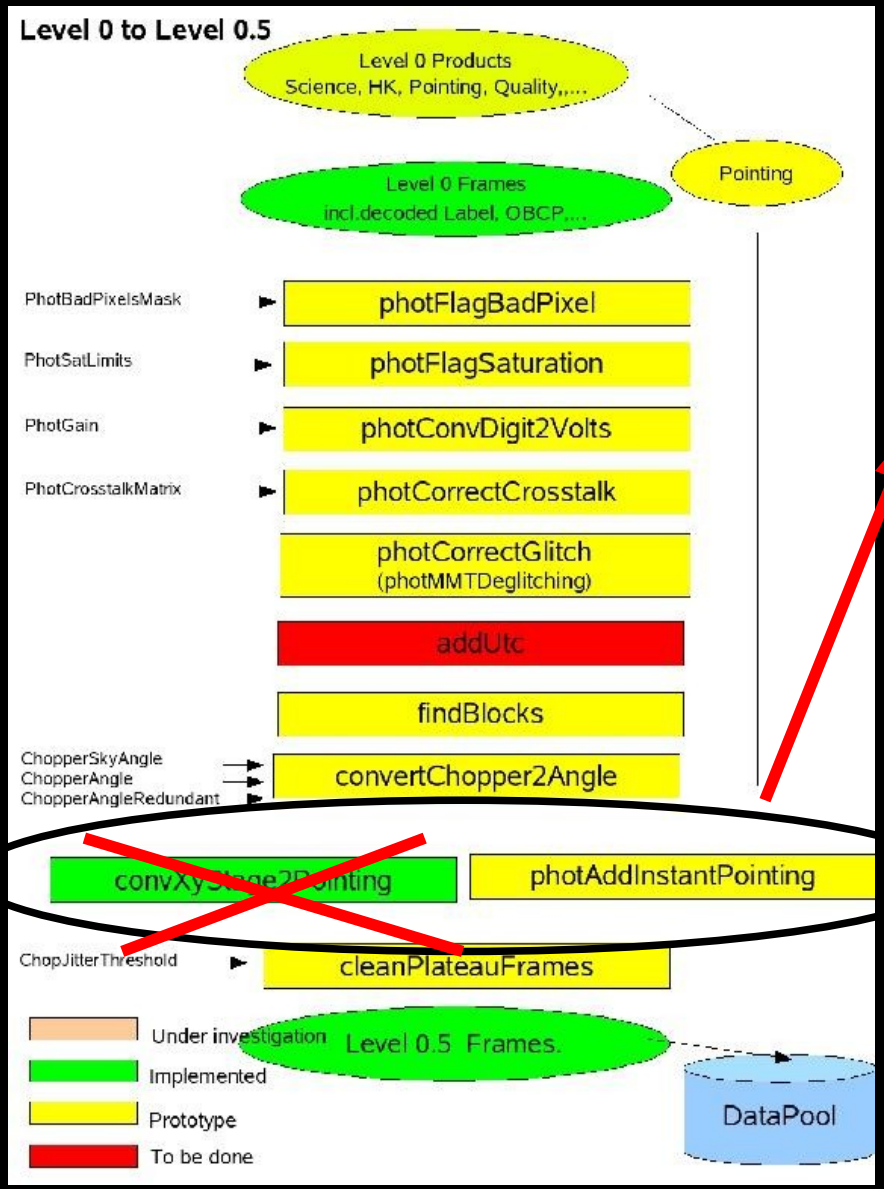
The pipeline Pointing information



PhotSddInstantPointing is accessing:

- The Herschel pointing Product
- The SIAM product which provides position of PACS virtual aperture (center of the bolometer) respect to the spacecraft pointing
- Ra and Dec of virtual aperture (plus error) stored for each frame in the Status table

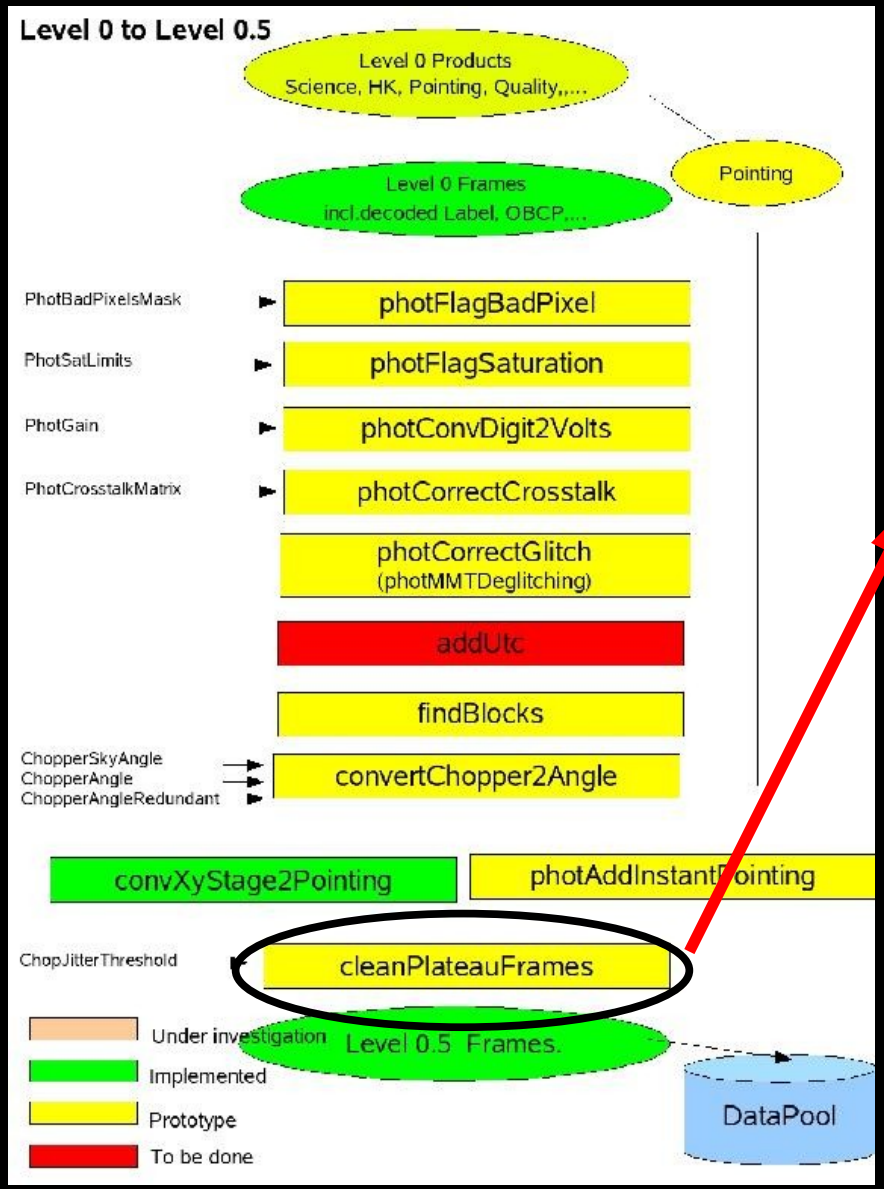
The pipeline Pointing information



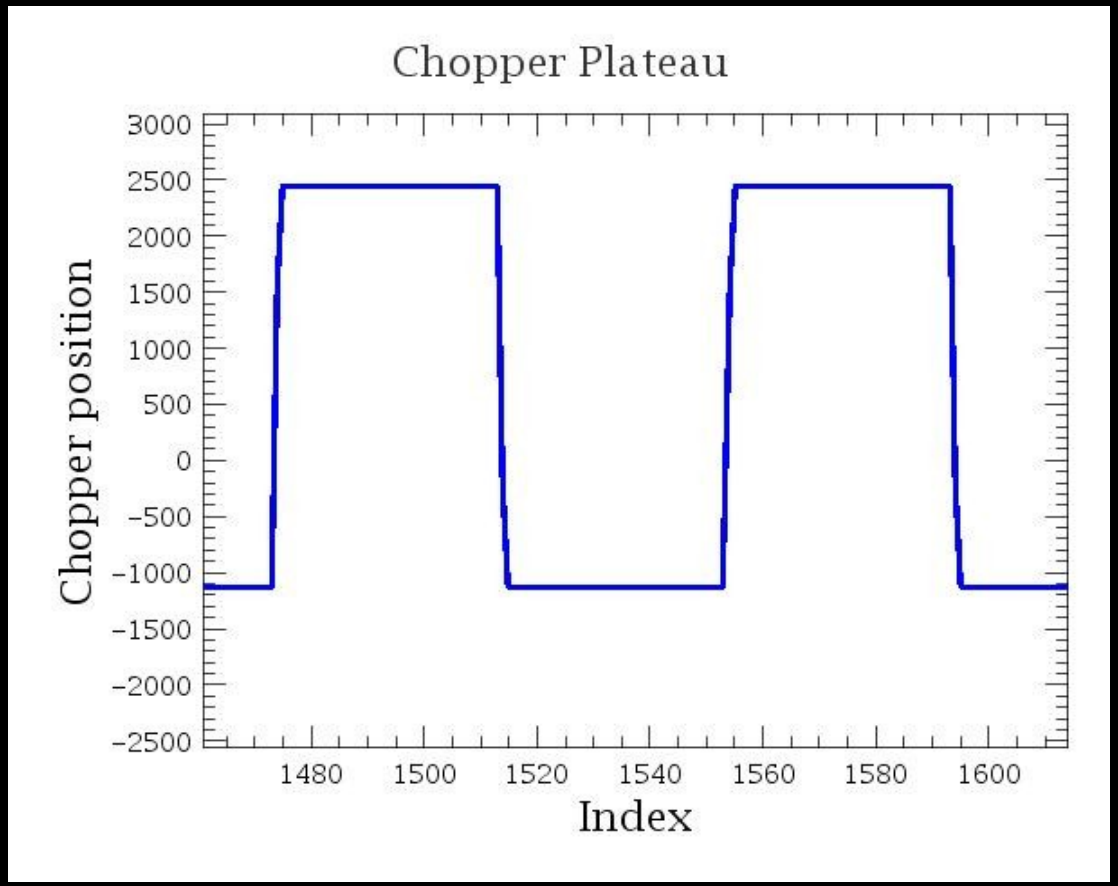
PhotSddInstantPointing is accessing:

- The Herschel pointing Product
- The SIAM product which provides position of PACS virtual aperture (center of the bolometer) respect to the spacecraft pointing
- Ra and Dec of virtual aperture (plus error) stored for each frame in the Status table

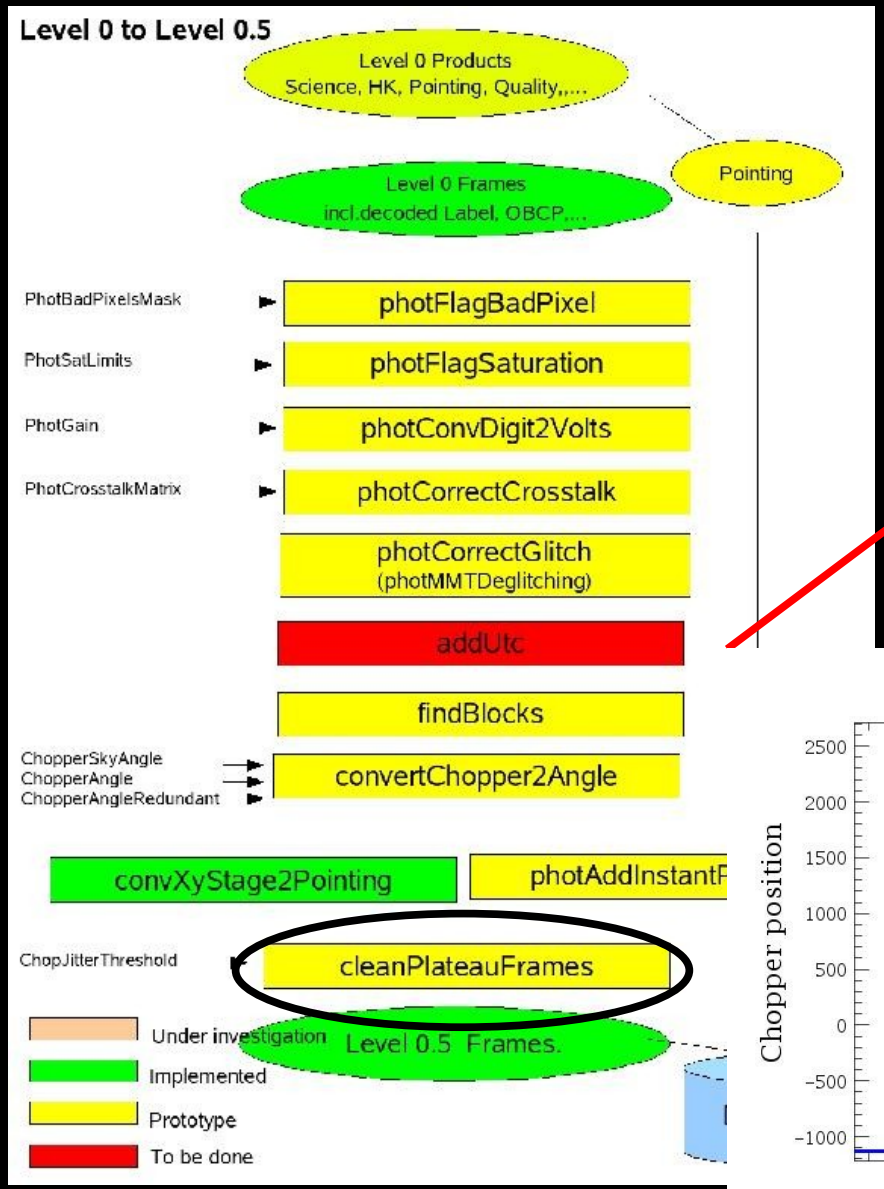
The chopper plateau



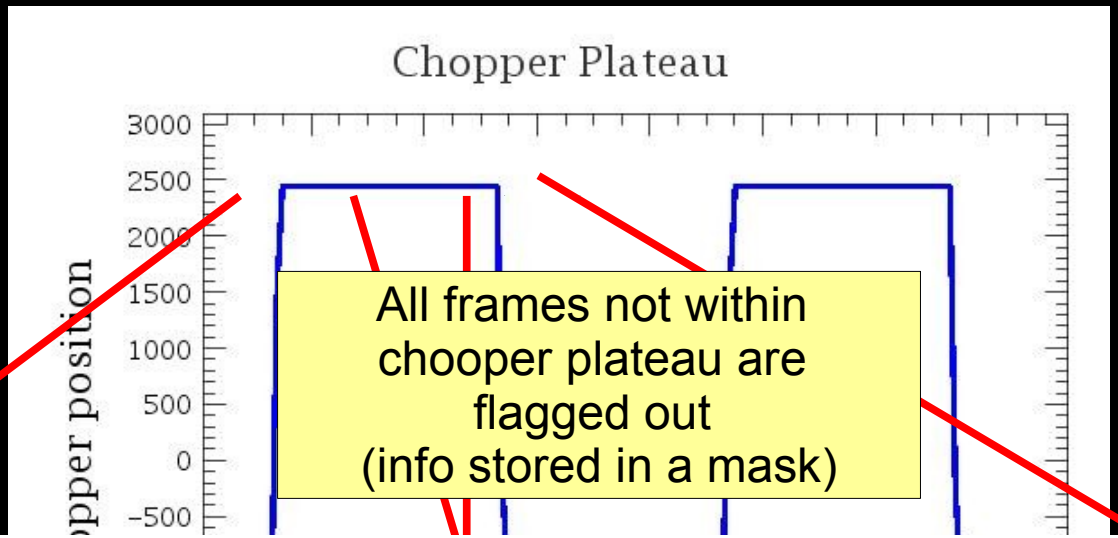
For chopped observation only

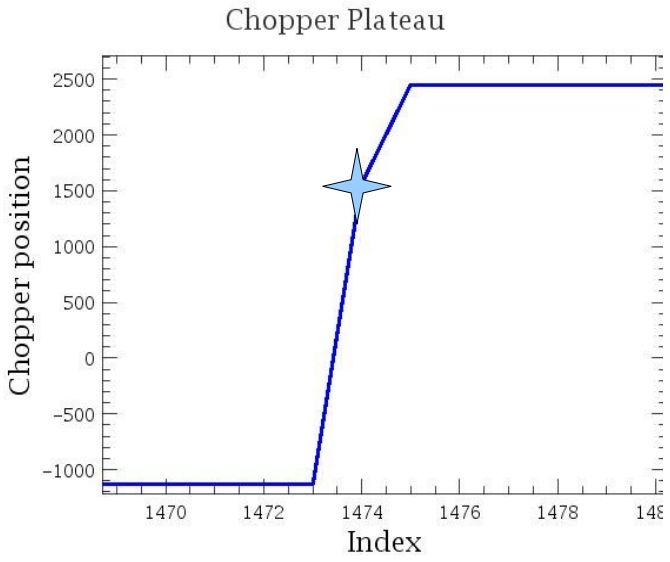


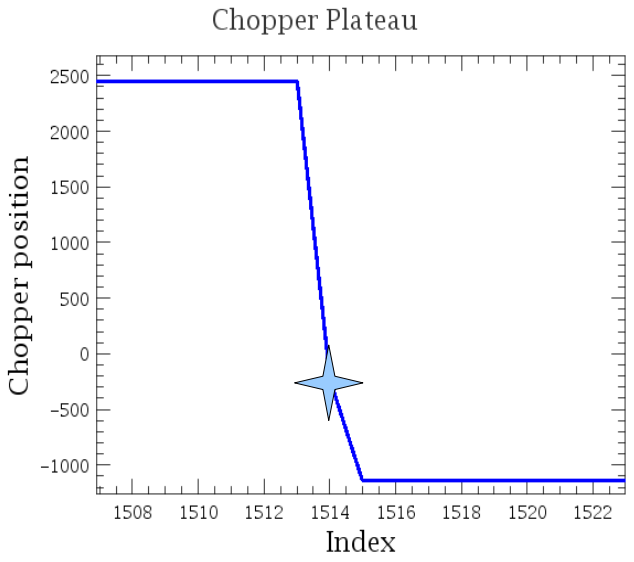
The chopper plateau



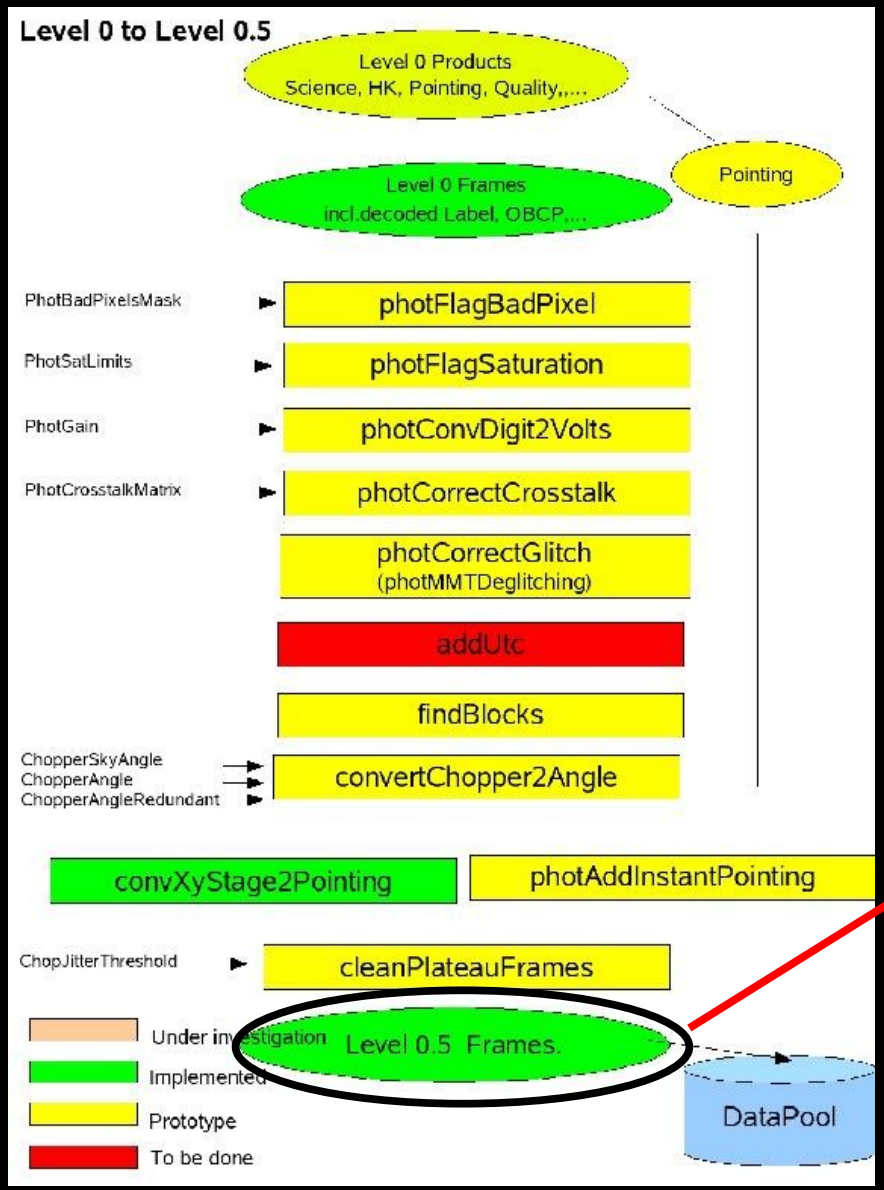
For chopped observation only





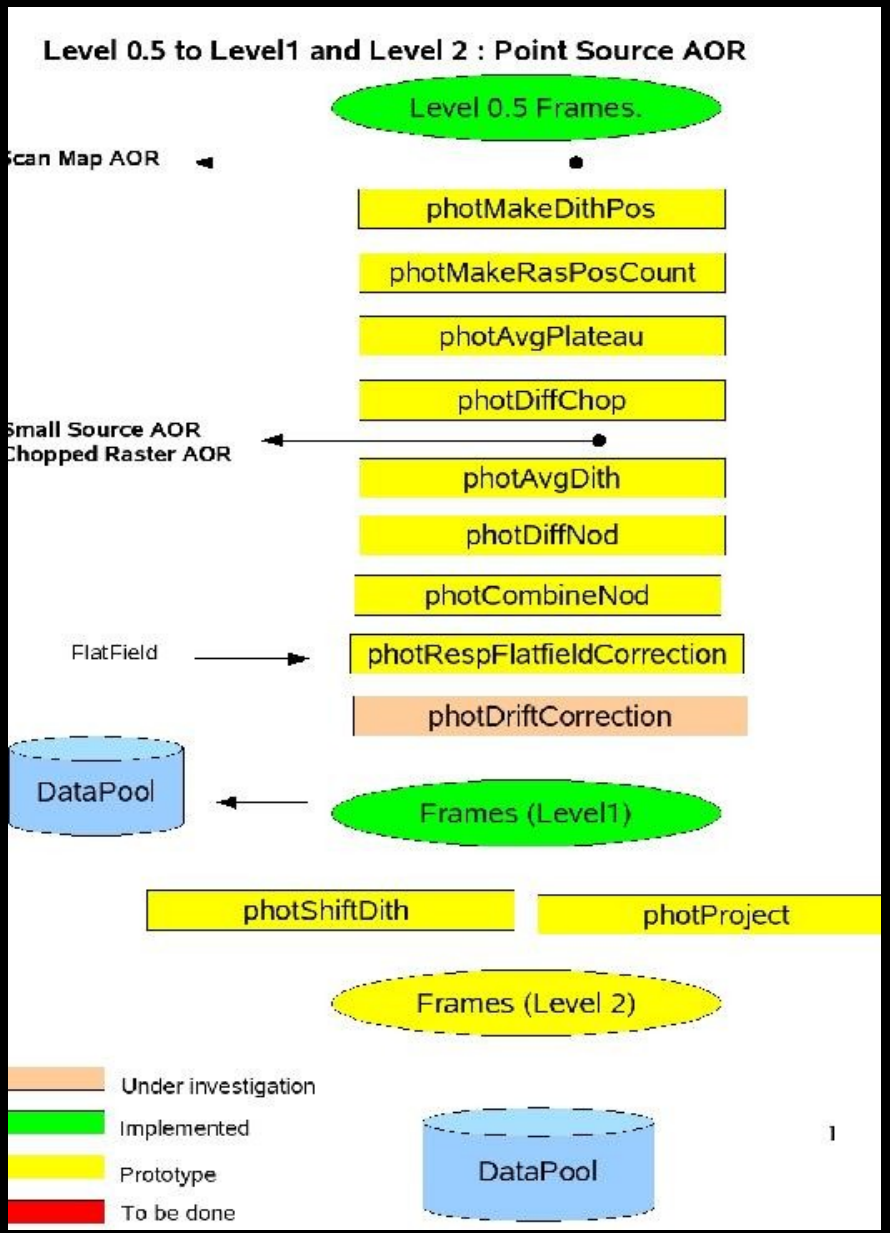


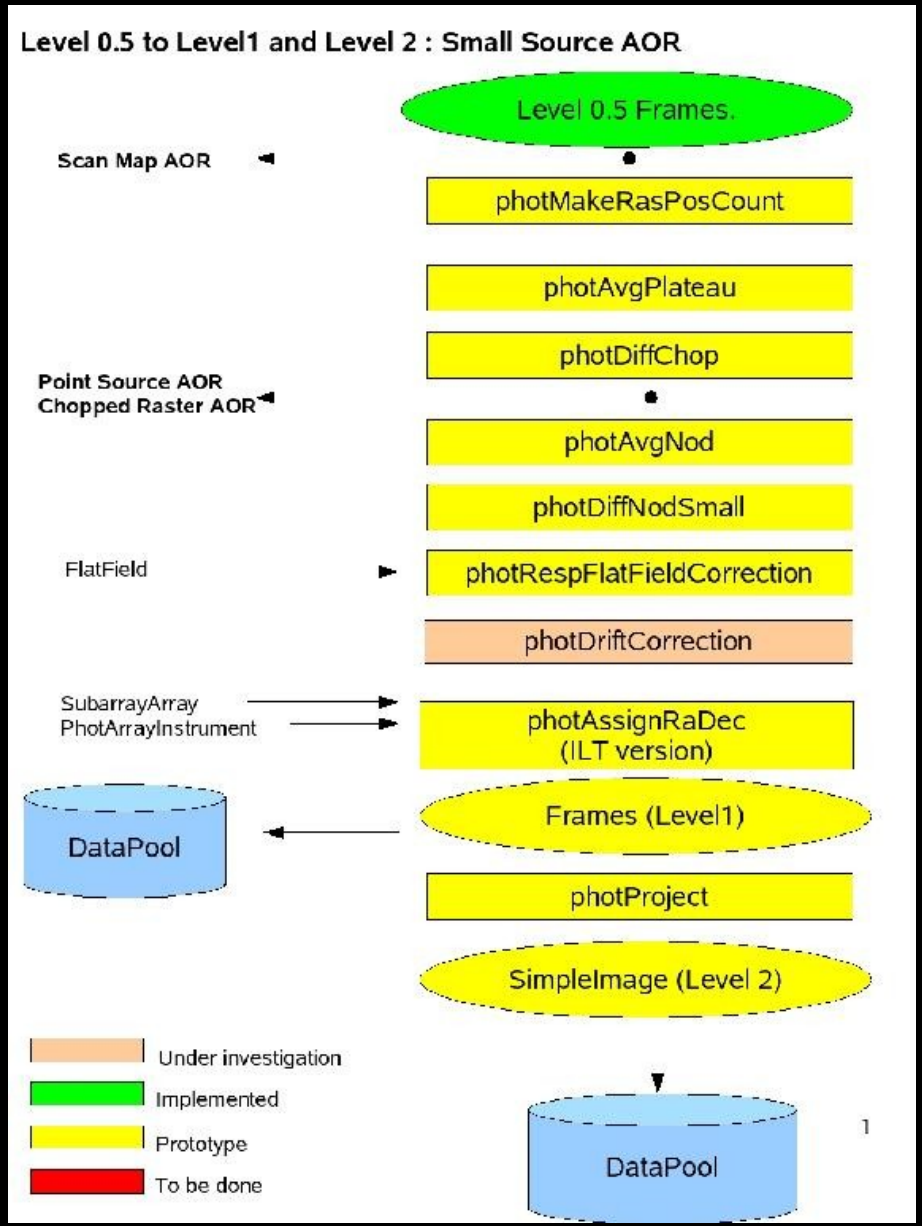
Level 0.5



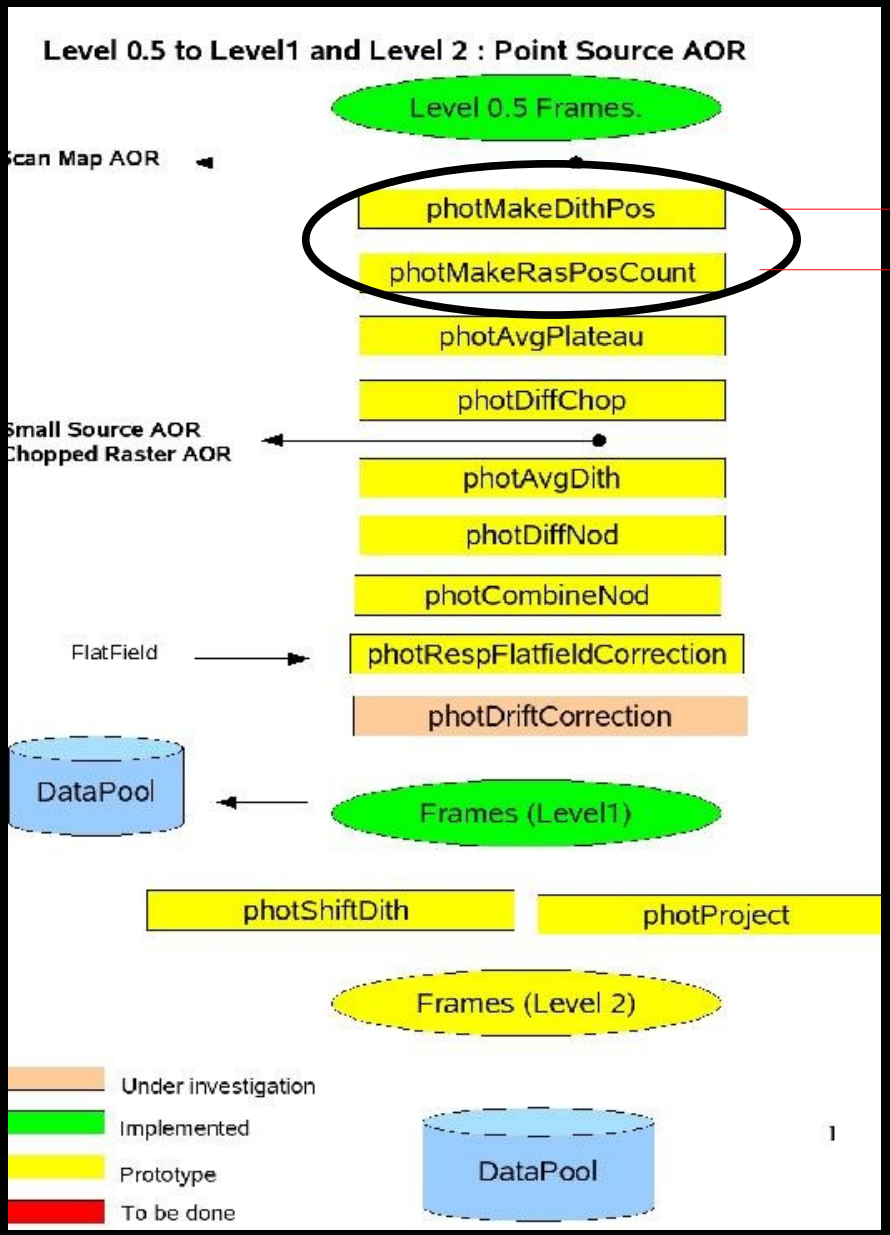
From Level 0.5 on the pipeline is AOT dependent

Point-source and Small Source from level 0.5 up to level 2

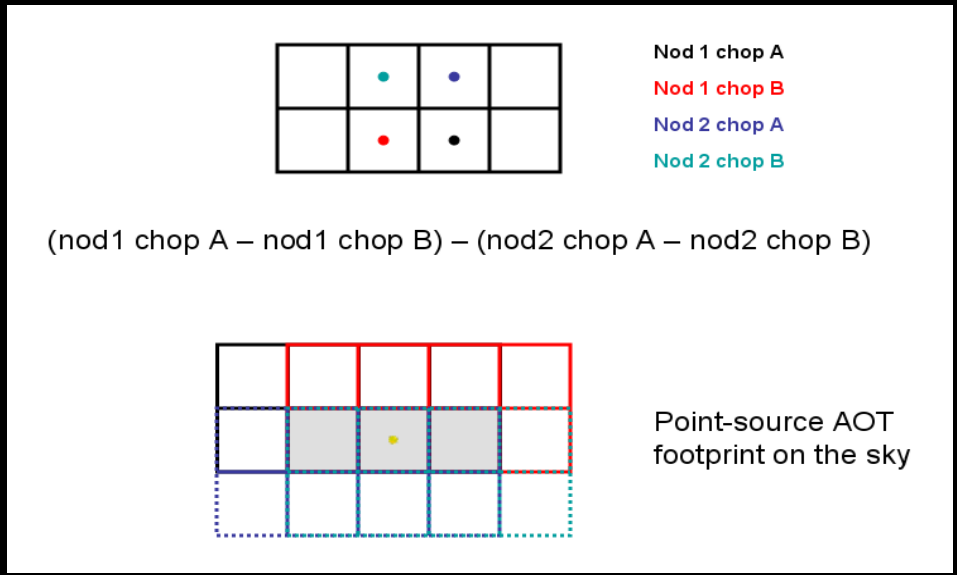




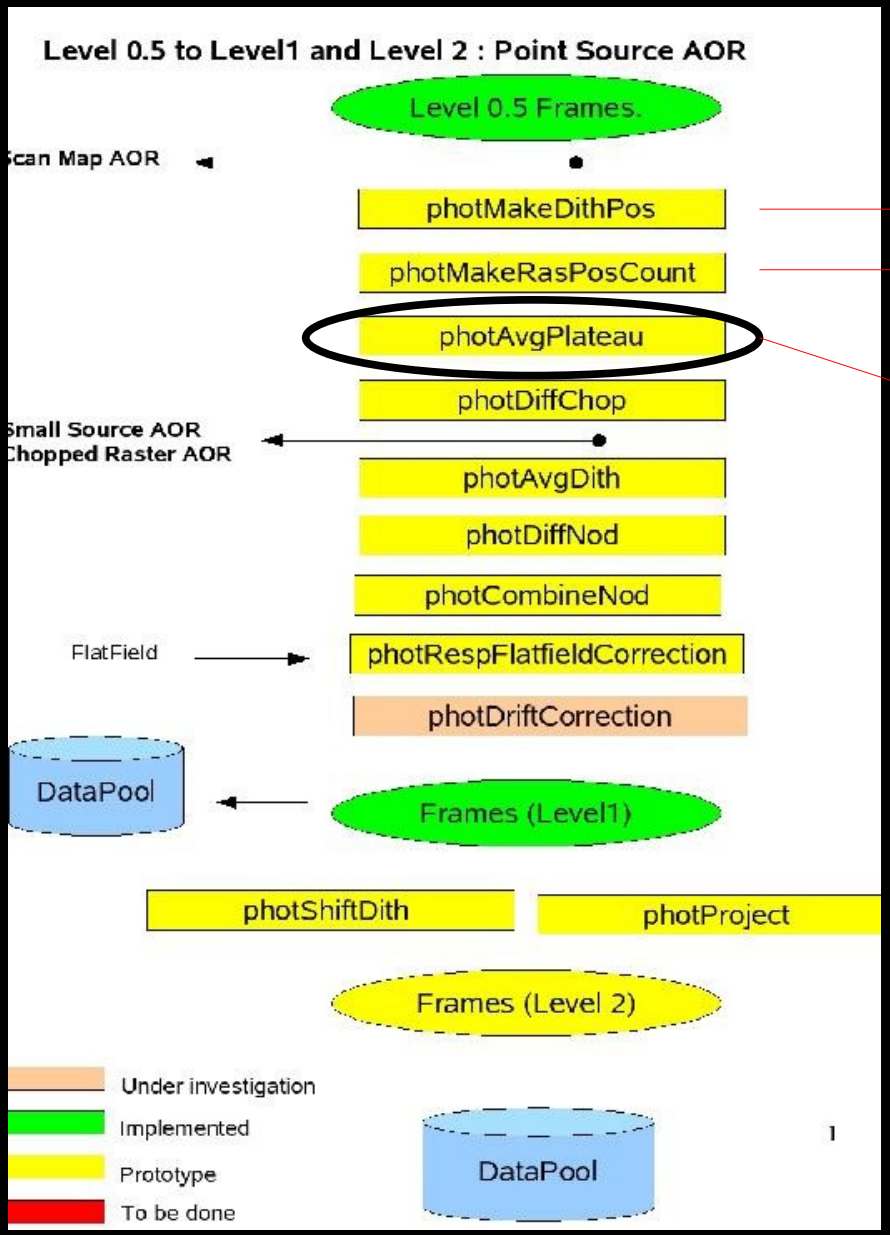
Point-source and Small Source from level 0.5 up to level 2



- Identifies dithering pattern
- Identifies on-off source and nodA-nodB position on the basis of info in the Status table

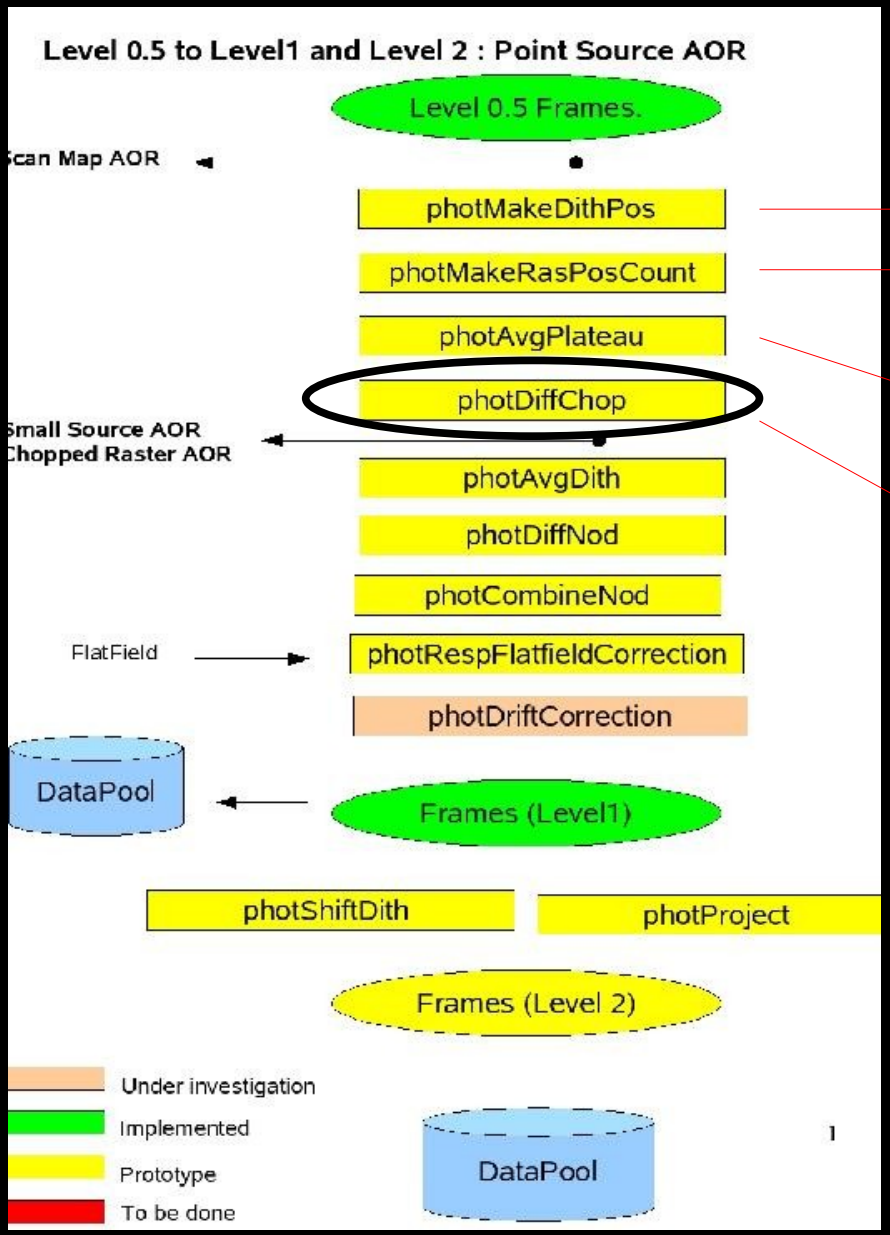


Point-source and Small Source from level 0.5 up to level 2



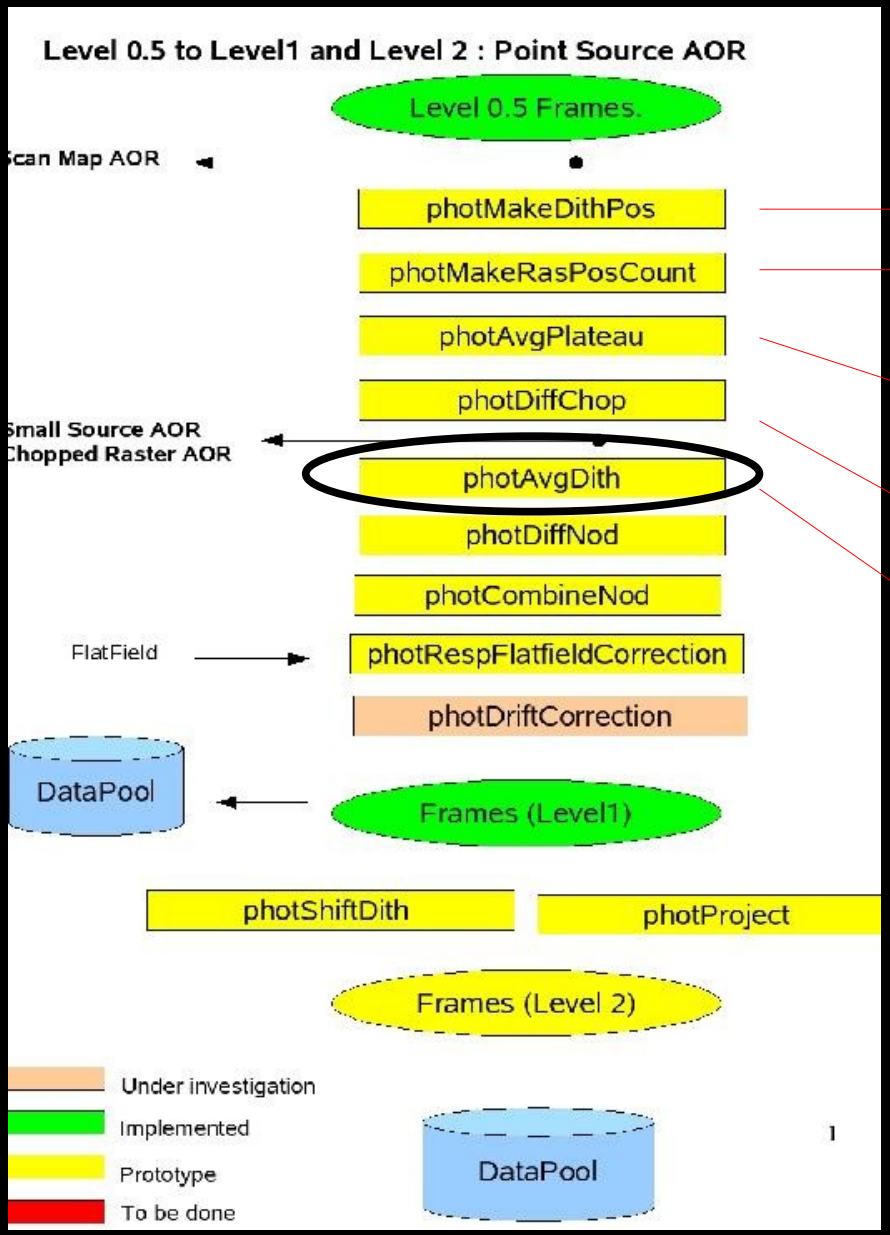
- identifies dithering pattern
- identifies on-off source positions and nodA-nodB positions on the basis of info stored in the Status table
- averages the signal over a chopper plateau (with the exclusion of frames flagged by CleanPlateau task)

Point-source and Small Source from level 0.5 up to level 2



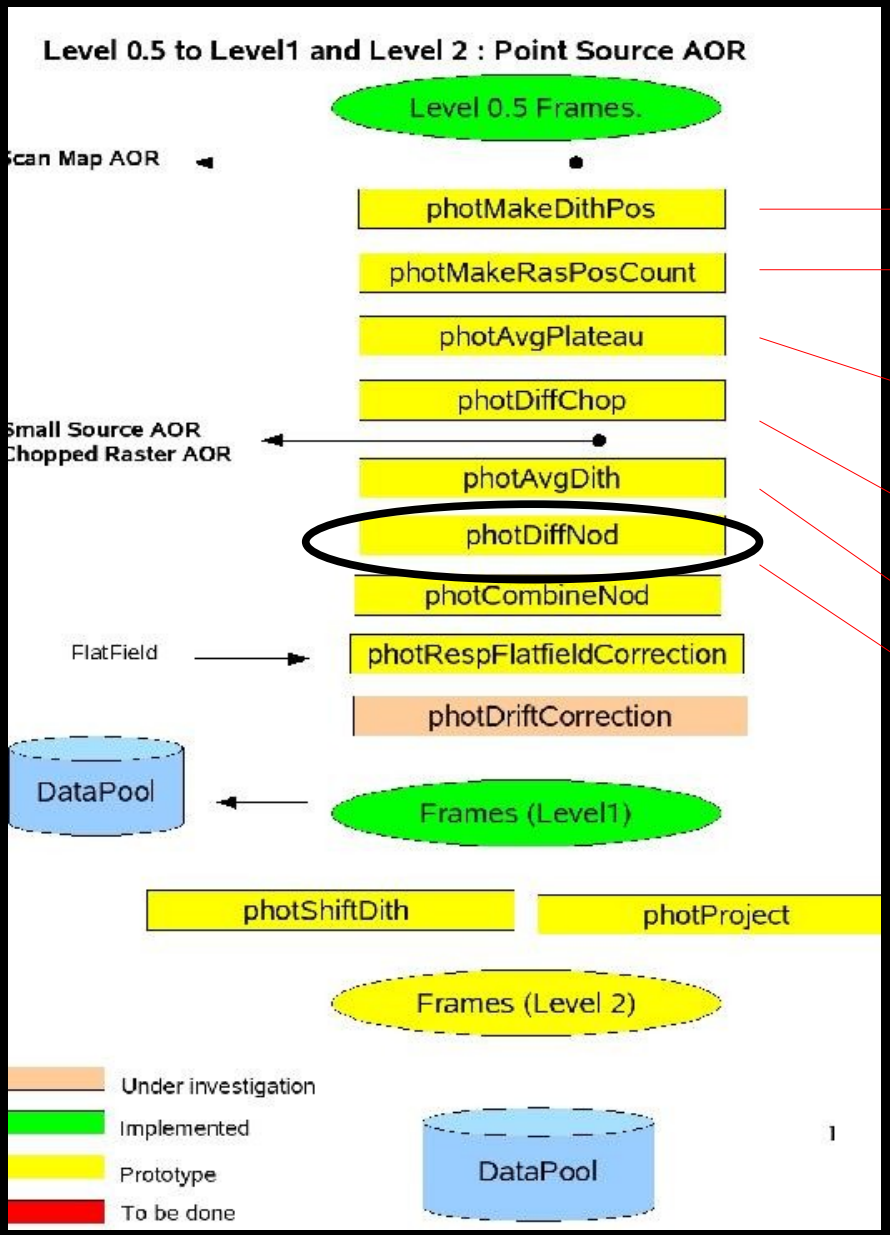
- identifies dithering pattern
- identifies on-off source positions and nodA-nodB positions on the basis of info stored in the Status table
- averages the signal over a chopper plateau (with the exclusion of frames flagged by CleanPlateau task)
- takes the difference of averaged Chopper plateaus (on-off)

Point-source and Small Source from level 0.5 up to level 2



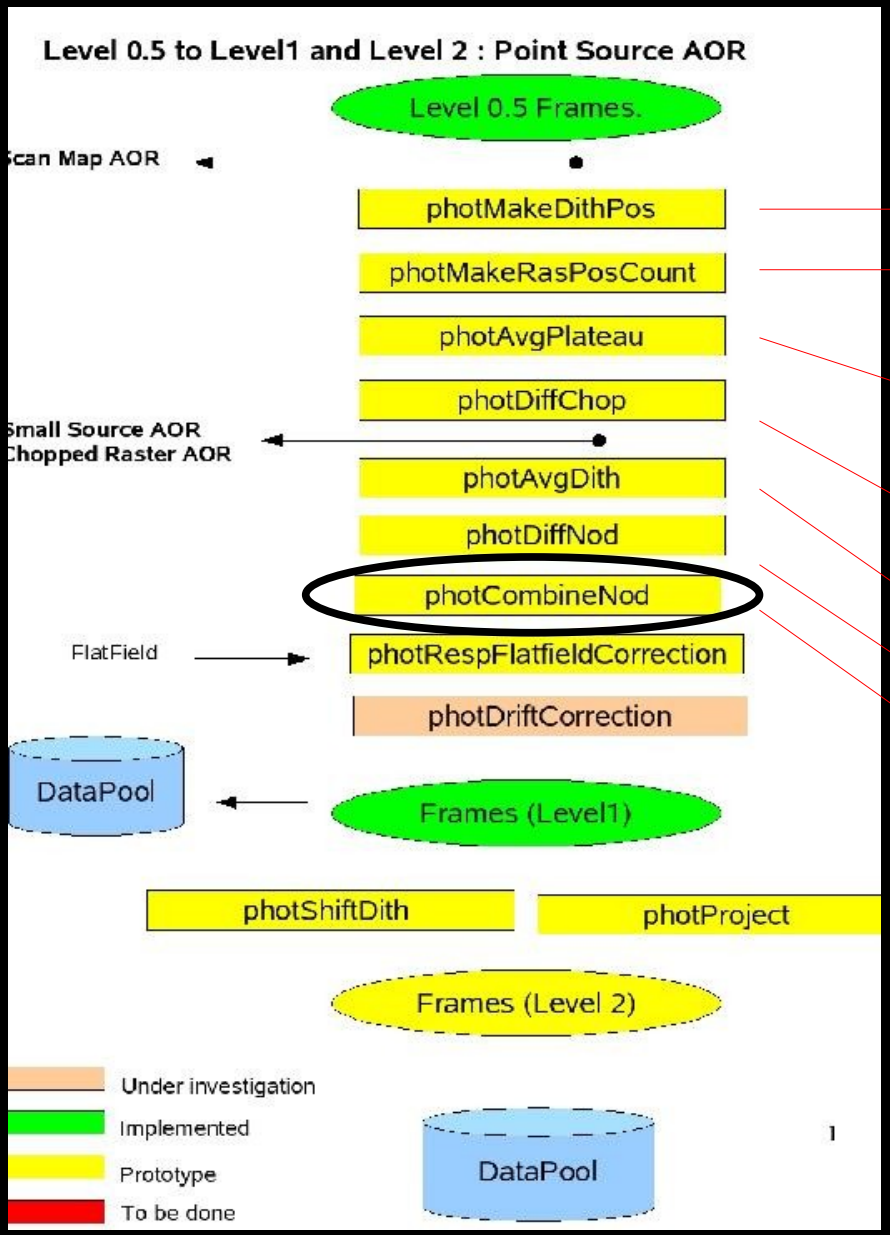
- identifies dithering pattern
- identifies on-off source positions and nodA-nodB positions on the basis of info stored in the Status table
- averages the signal over a chopper plateau (with the exclusion of frames flagged by CleanPlateau task)
- takes the difference of averaged Chopper plateaus (on-off)
- Averages chopped differential images per nod and dither position

Point-source and Small Source from level 0.5 up to level 2



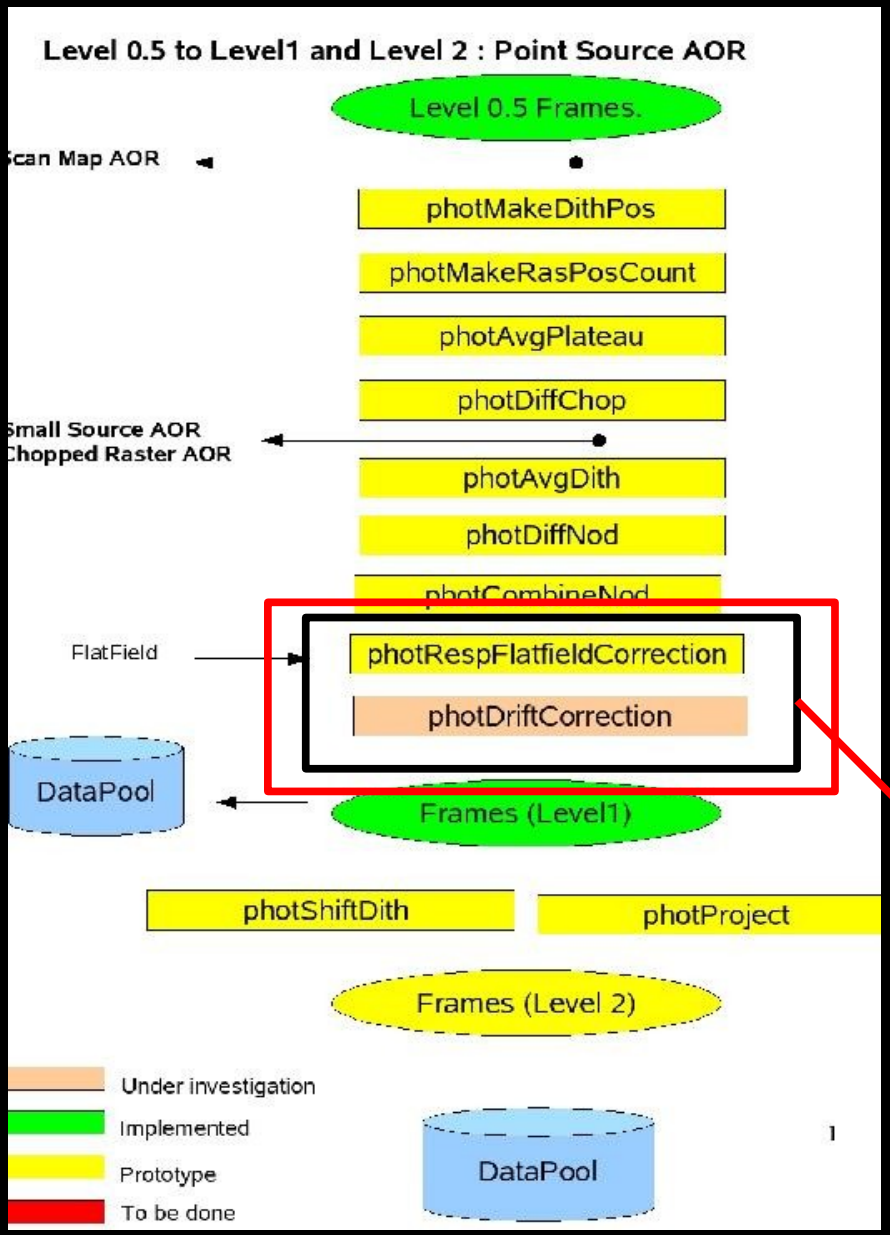
- identifies dithering pattern
- identifies on-off source positions and nodA-nodB positions on the basis of info stored in the Status table
- averages the signal over a chopper plateau (with the exclusion of frames flagged by CleanPlateau task)
- takes the difference of averaged Chopper plateaus (on-off)
- averages chopped differential images per nod and dither position
- takes the difference nodA(on-off)-nodB(on-off)

Point-source and Small Source from level 0.5 up to level 2



- identifies dithering pattern
- identifies on-off source positions and nodA-nodB positions on the basis of info stored in the Status table
- averages the signal over a chopper plateau (with the exclusion of frames flagged by CleanPlateau task)
- takes the difference of averaged Chopper plateaus (on-off)
- averages chopped differential images per nod and dither position
- takes the difference nodA(on-off)-nodB(on-off)
- averages nodA-nodB per dither position

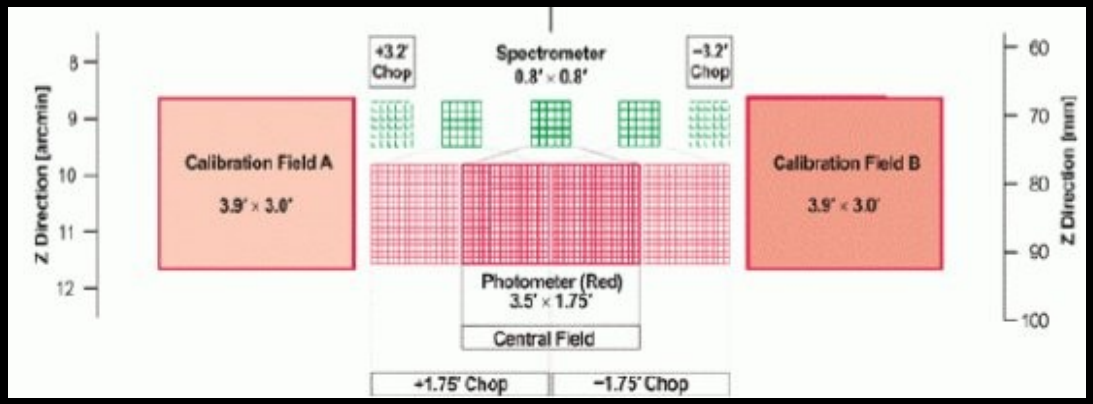
Point-source and Small Source from level 0.5 up to level 2

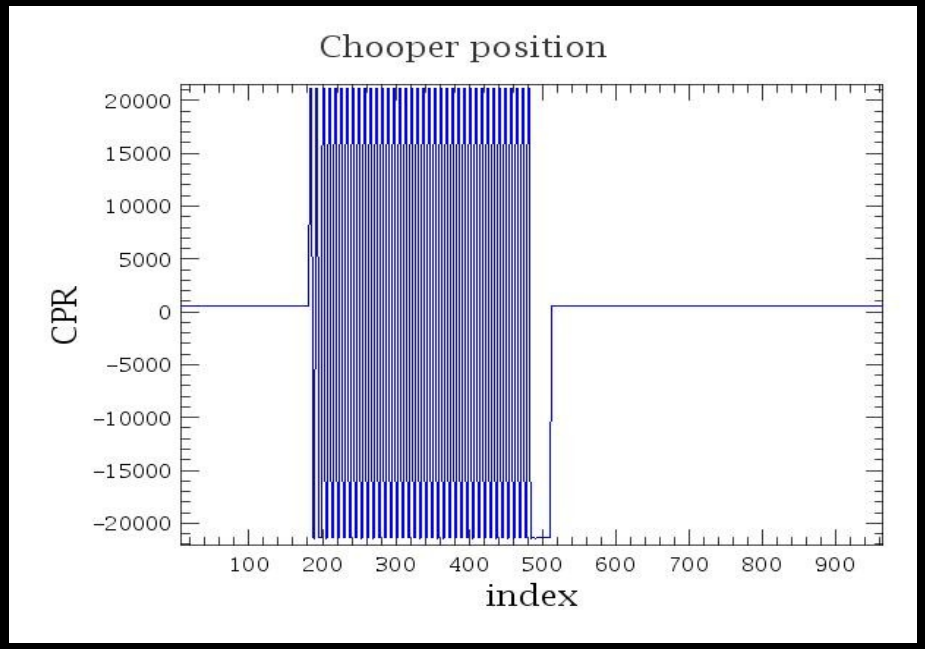


- No `flat' astronomical calibrators
- Internal calibrators are stable but not flat
- Blackbodies used for tests in Lab can be assumed flat, but some differential distortion to in-orbit case
- start with ground testing based flat-field with given instrument settings (bias, gain, etc)
- verify large scale trends from point sources scan maps/rasters
- Correct drift with internal Calibration sources

+ Preprocessing of the Calibration Block

Flat-Field and Flux calibration





Cal Block pre-processing

- extracts cal block from observation
- reduces Cal Block as point-source AOT up to PhotDiffChop
- provides differential image of calibration sources (CS1-CS2 image)
- + noise + HK data (e.g. Gain, bias settings) to correct flat-field and flux calibration

Flat-Field and Flux calibration

- $\delta S(t)$ = background subtracted detector science signal of a given pixel in volts.
- $\delta C(\mathrm{pix})$ = signal difference (in V) between the two CSs for a given pixel, measured during the AOR calblock (this could be an interpolarion between defferent calblocks, if needed, i.e. due to some smooth time dependency).
- J = fixed scalar calfile value Jy/V for a given array and filter combination. This includes the conversion through the instrument and the telescope, such as the filter transmission and also the inverse of the responsivity.
- $\phi(\mathrm{pix})$ = flat field for the pixel in the normalized 2D flat field. This has a value around the unity.
- $\delta C_0(\mathrm{pix})$ = signal difference (in V) between the two CSs measured especially for the calibration independently from the science data, i.e. consistent with the data from which the flat field ϕ and the responsivity R were derived.
- $\delta f(t)$ = back ground subtracted calibrated flux seen by a given pixel in Jy/pixel.

- The flux calibration is done by the following:

$$\delta f(t) = \delta S(t)\,\frac{\delta C_0(\mathrm{pix})}{\delta C(\mathrm{pix})}\,\frac{J}{\phi(\mathrm{pix})}$$

Monitoring with a calibration block consists monitoring the ratio ρ of the seconf term:

$$\rho(\mathrm{pix}) = \frac{\delta C_0(\mathrm{pix})}{\delta C(\mathrm{pix})}$$

Drift correction

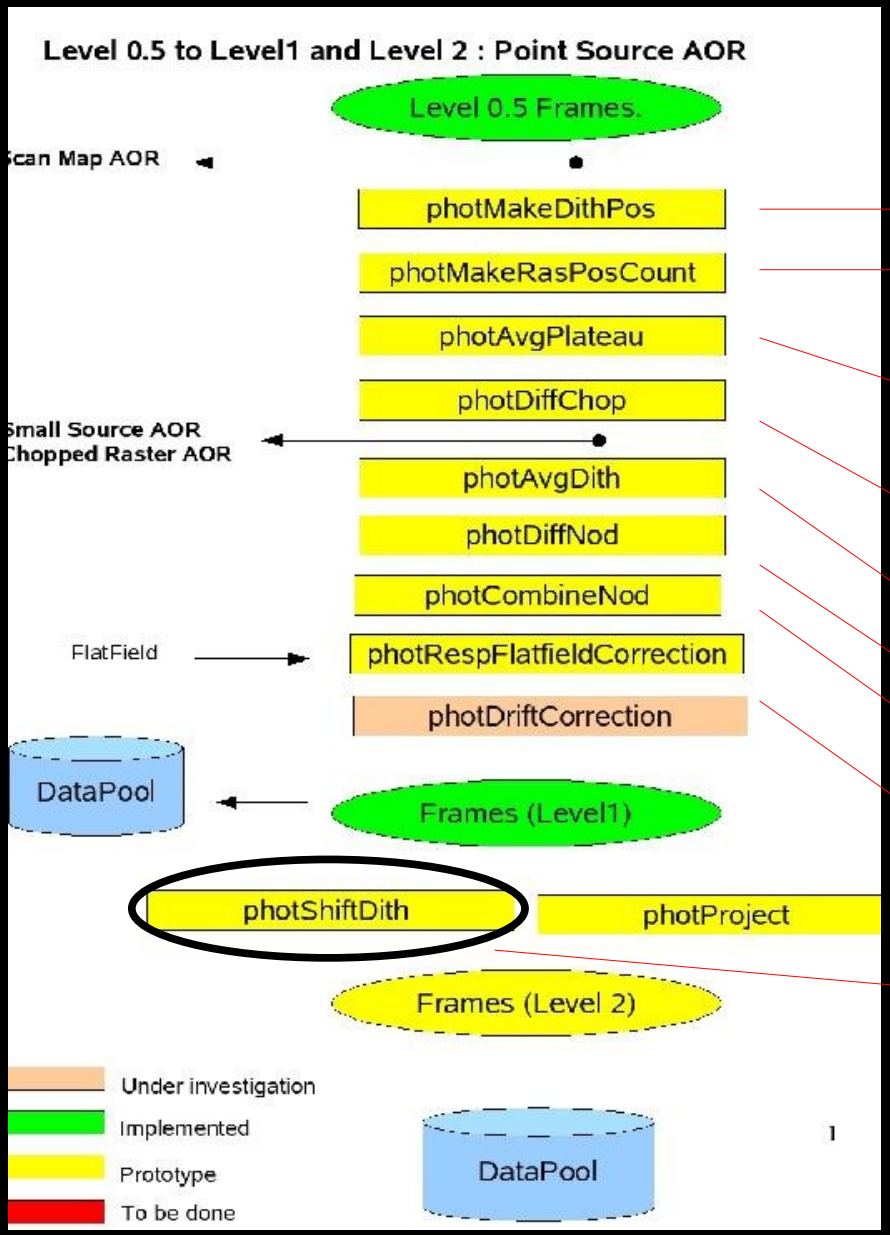
CalBlock pre-processing:
δC=CS1-CS2
δC noise map
HK info

From Cal files:
J, ϕ(pix), δC_0 derived from PACS test data with same instrument settings

PhotRespFlatFieldCorrection:
$\delta f(t)=\delta S(t) \times J/\phi(\mathrm{pix})$

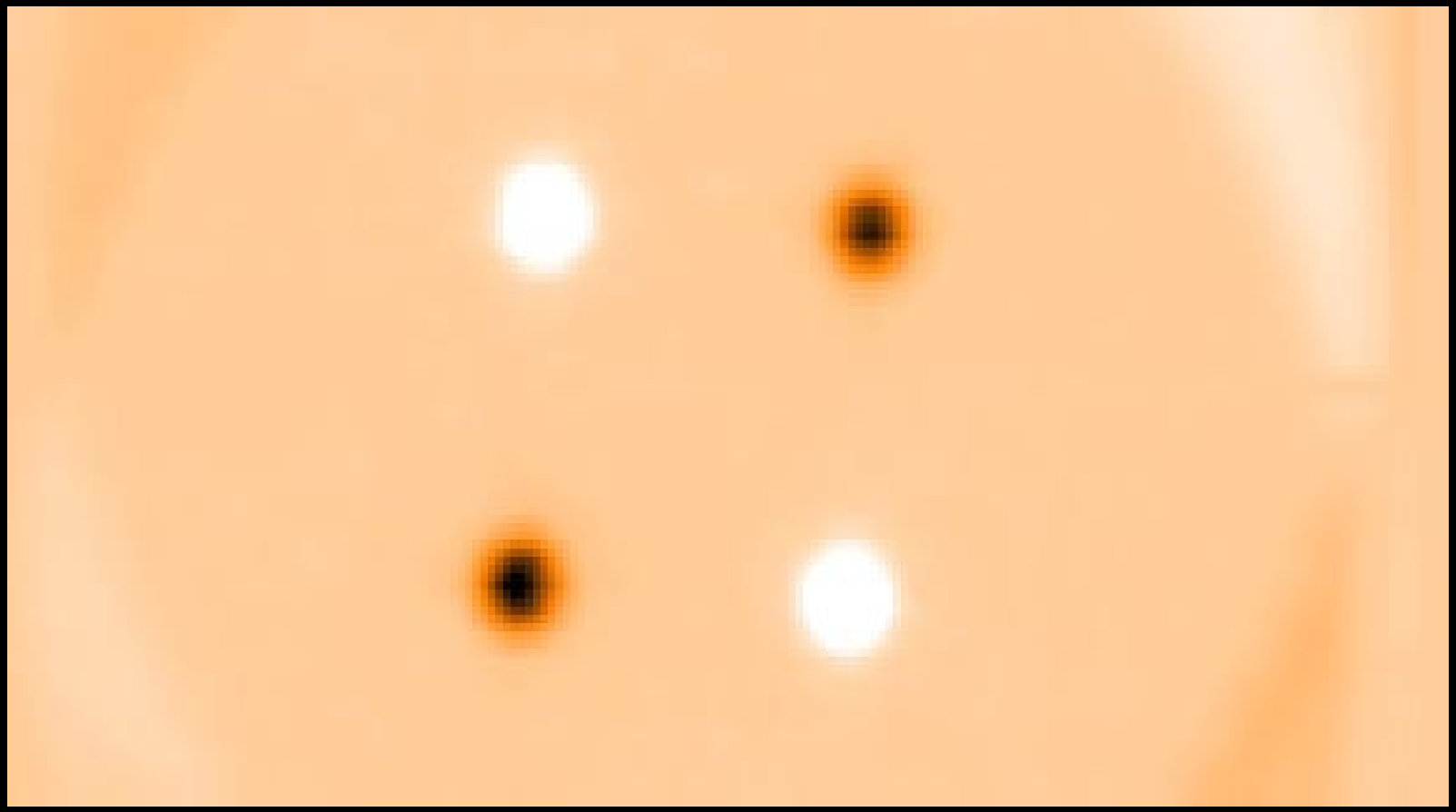
PhotDriftCorrection:
$\delta f_{\mathrm{flux\_cal}}(t)=\delta f(t) \times \rho(\mathrm{pix})$

Point-source and Small Source from level 0.5 up to level 2



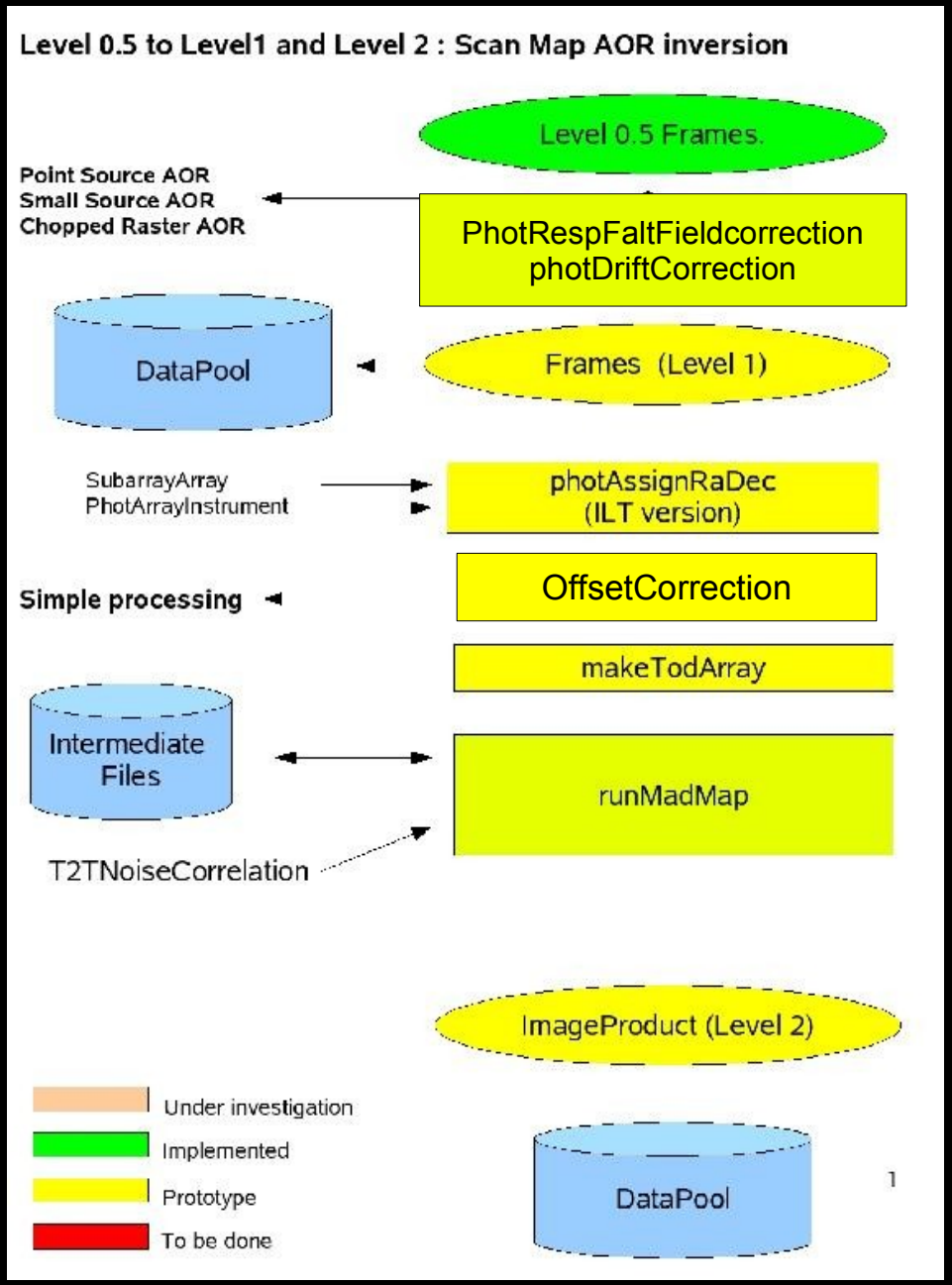
- identifies dithering pattern
- identifies on-off source positions and nodA-nodB positions on the basis of info stored in the Status table
- averages the signal over a chopper plateau (with the exclusion of frames flagged by CleanPlateau task)
- takes the difference of averaged Chopper plateaus (on-off)
- averages chopped differential images per nod and dither position
- takes the difference nodA(on-off)-nodB(on-off)
- averages nodA-nodB per dither position
- Flat fielding & flux calibration
- Combining dithered images (1/3 of pixels)

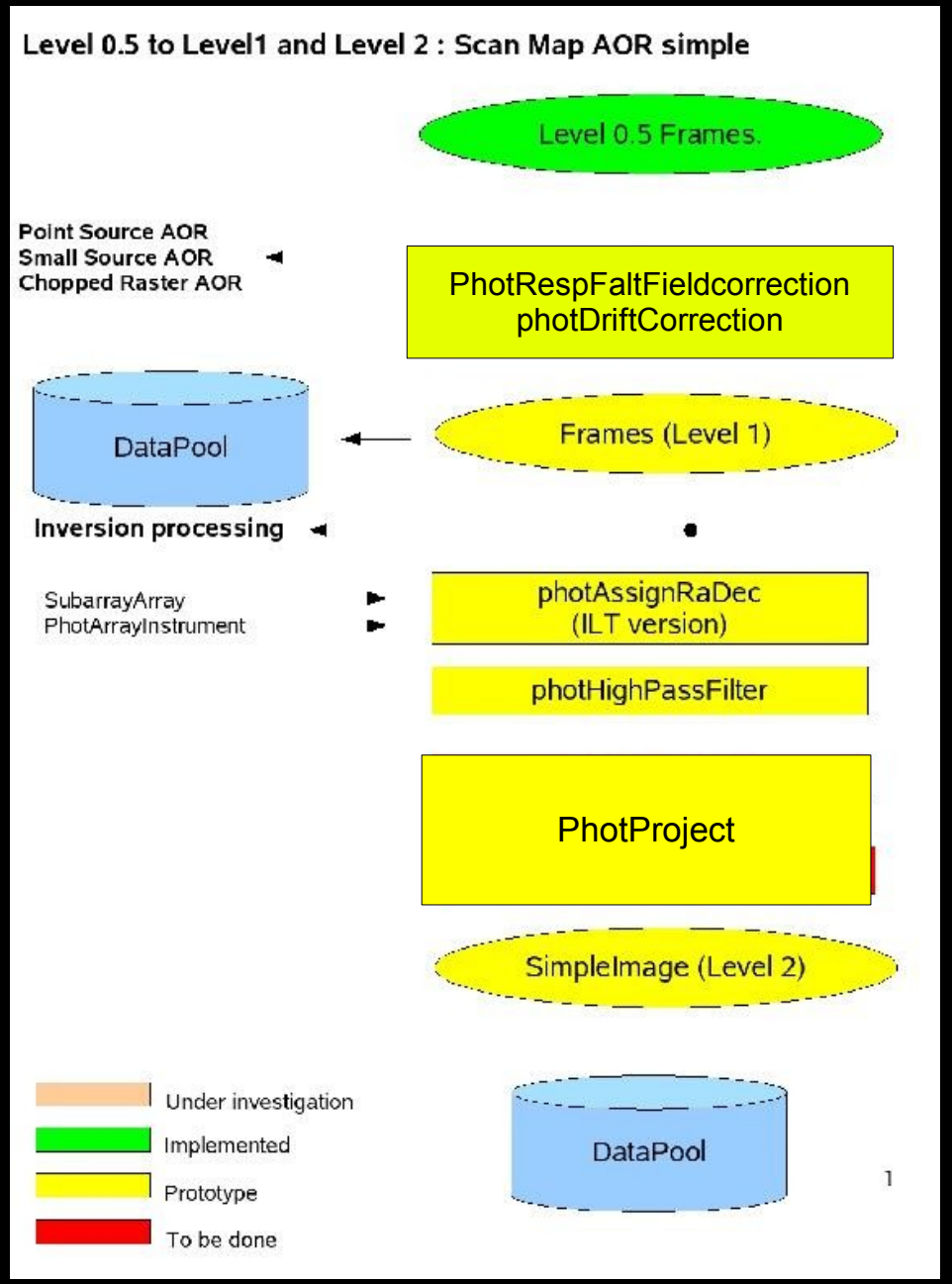
Point Source Level 2 Product



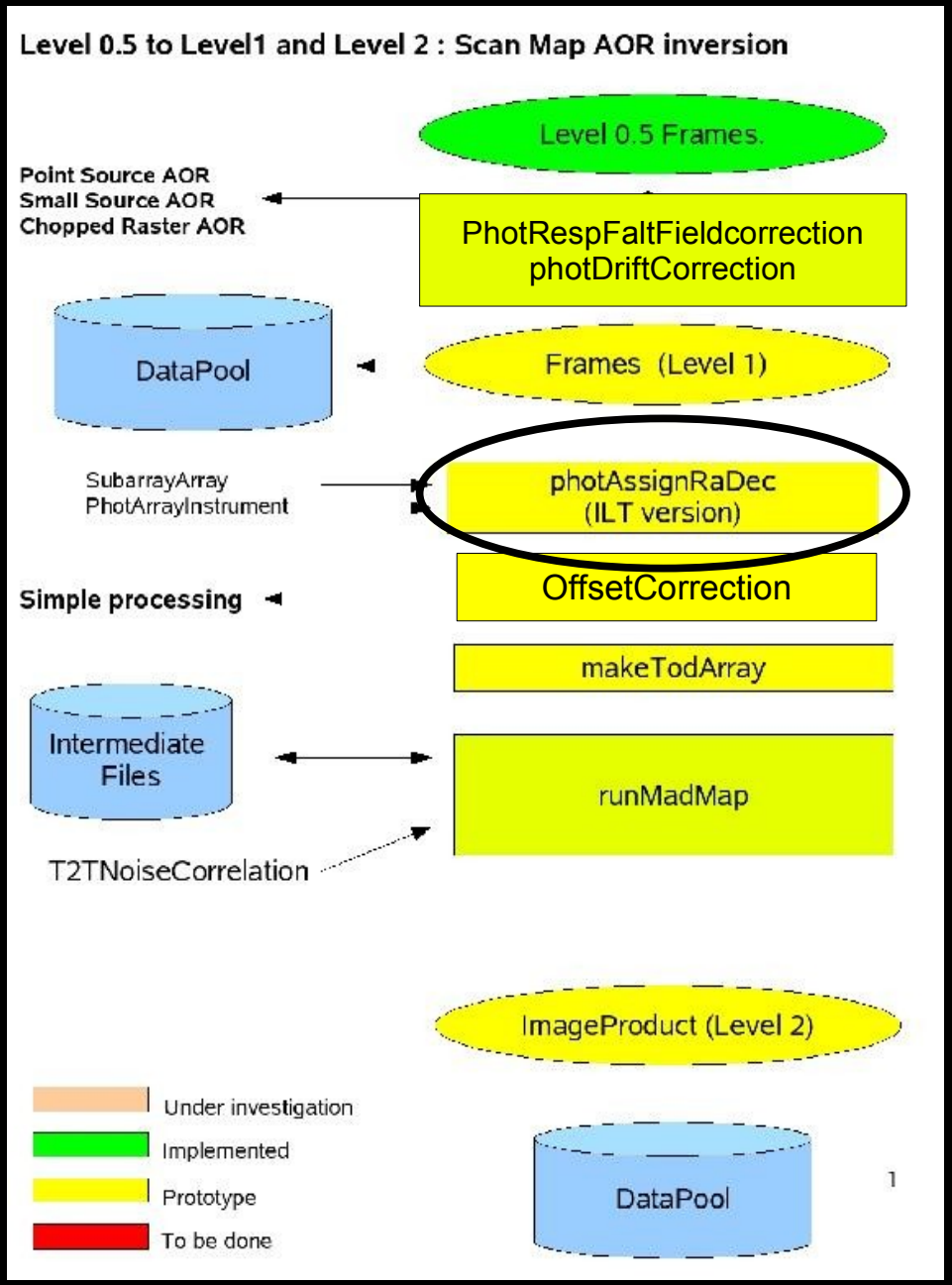
Double differential image
based on PACS test data

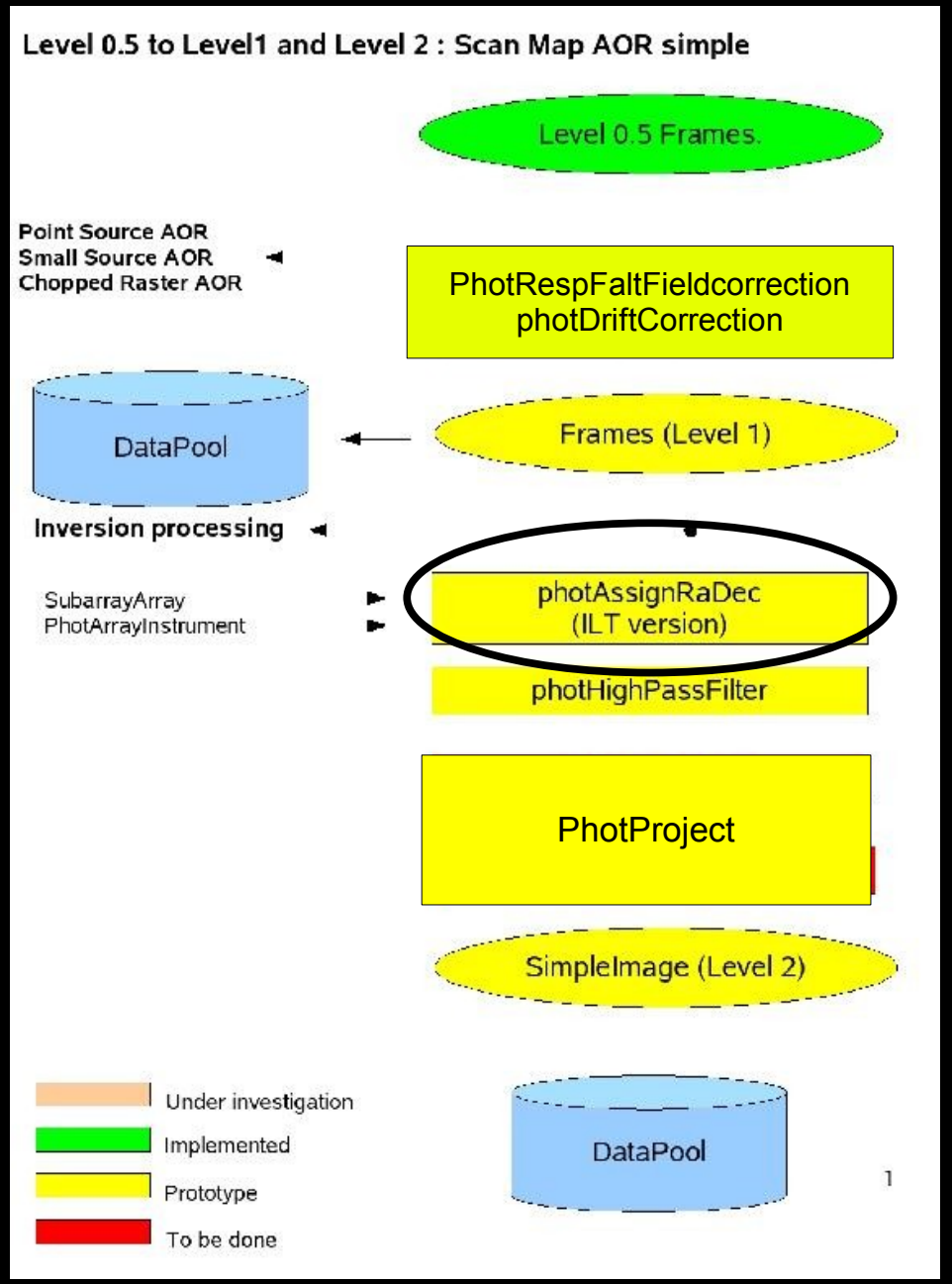
Scan map from level 0.5 to 2



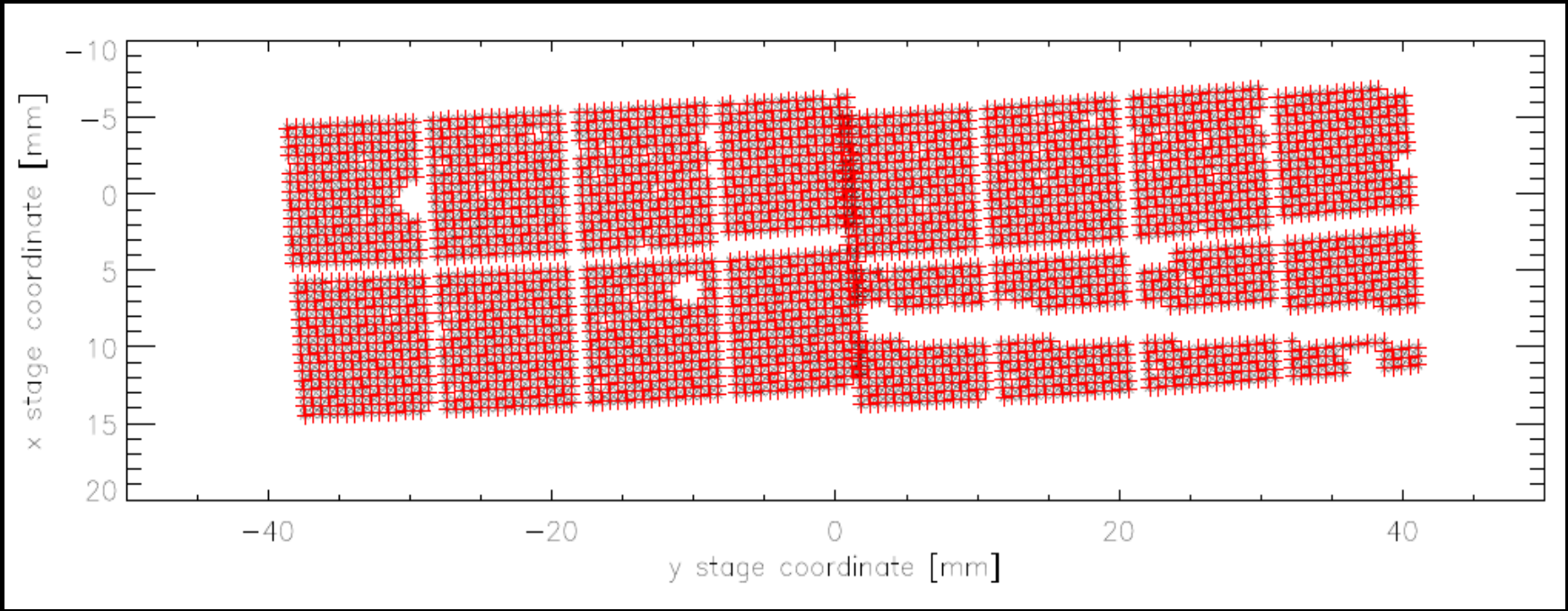


Scan map from level 0.5 to 2



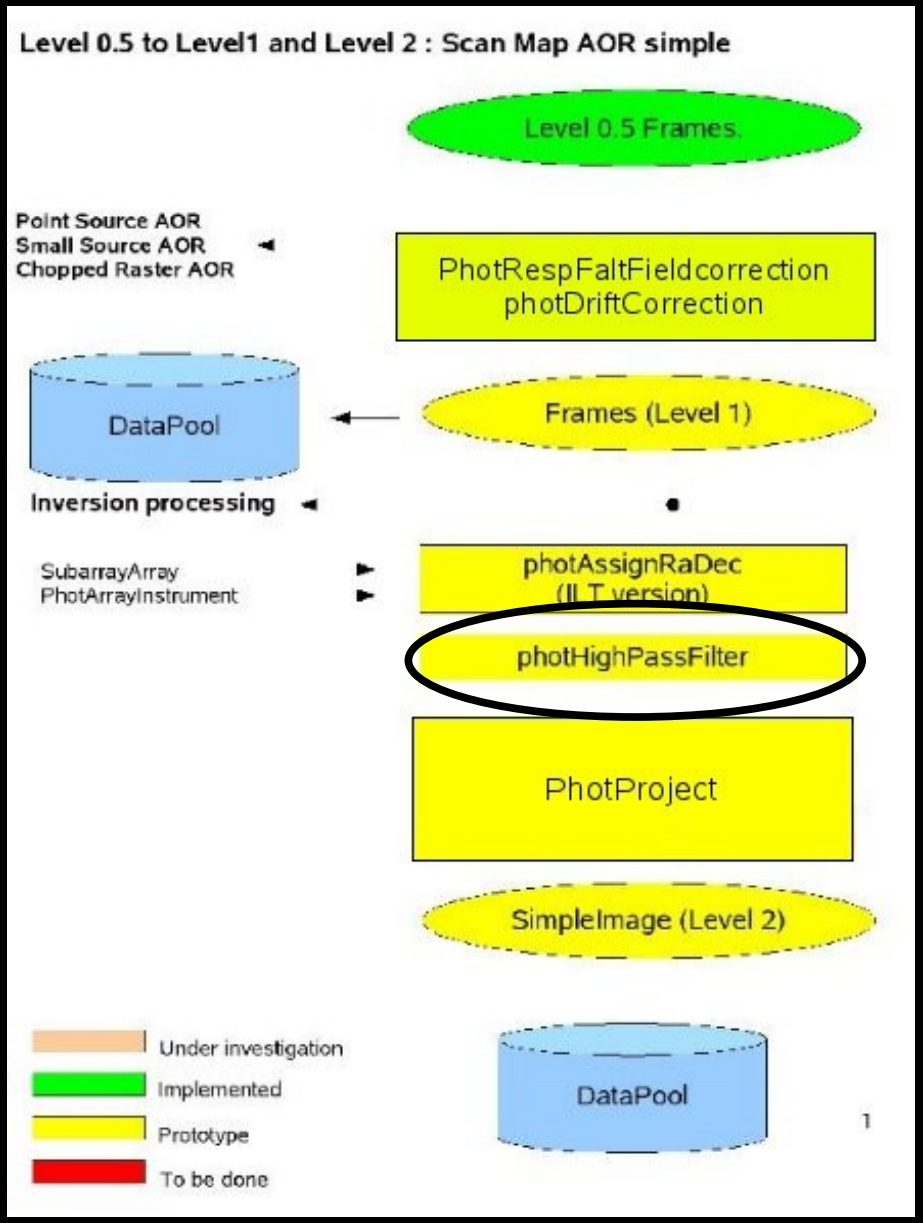


Pipeline Spatial Calibration



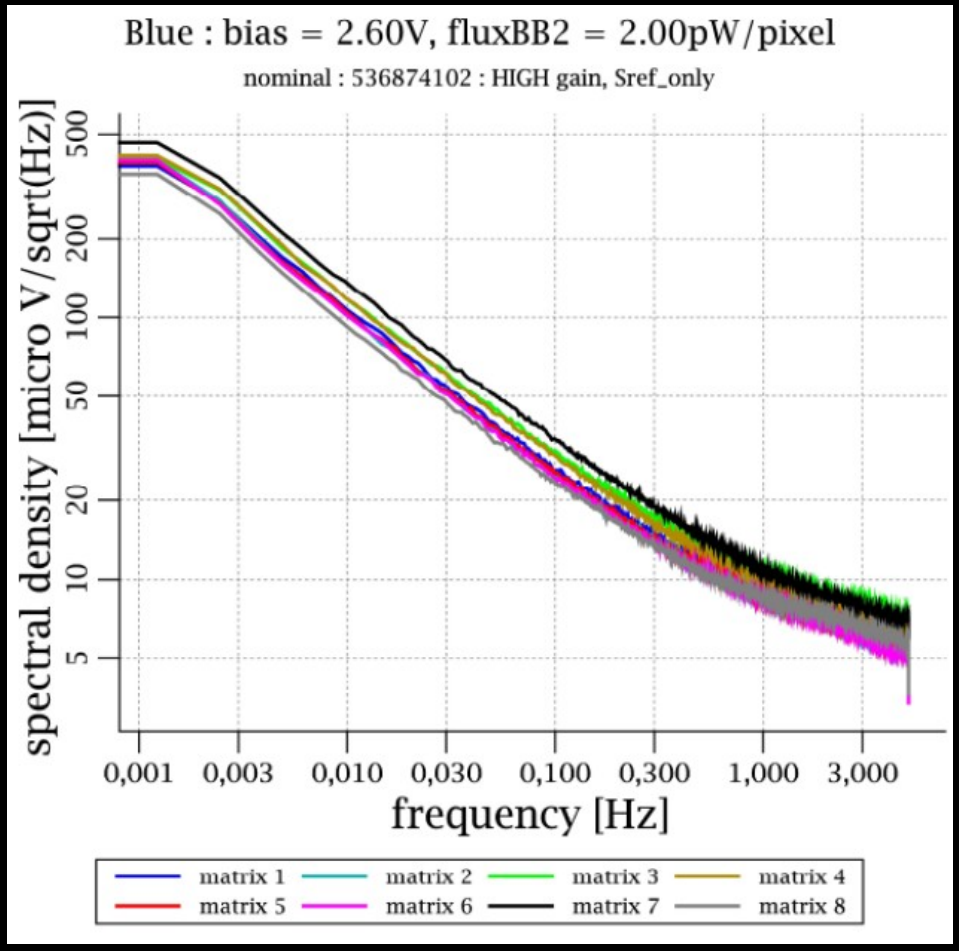
- 2nd order polynomial fit of distortions as a function of (x,y,α)
- In the current CalFile version : pixel coordinates transformed into XY stage coordinates
- Accurately characterized by large point source raster maps during PACS (Instrument level) tests in lab (for “external-optical-setup”+PACS, not Herschel+PACS)
- Method cannot be simply repeated in orbit: pointing accuracy!
- Differential optical modeling ongoing
- Verification from scans across point (or double) source

Pipeline Naive Projection



First Problem: 'Image sharpening'

Pipeline Naive Projection

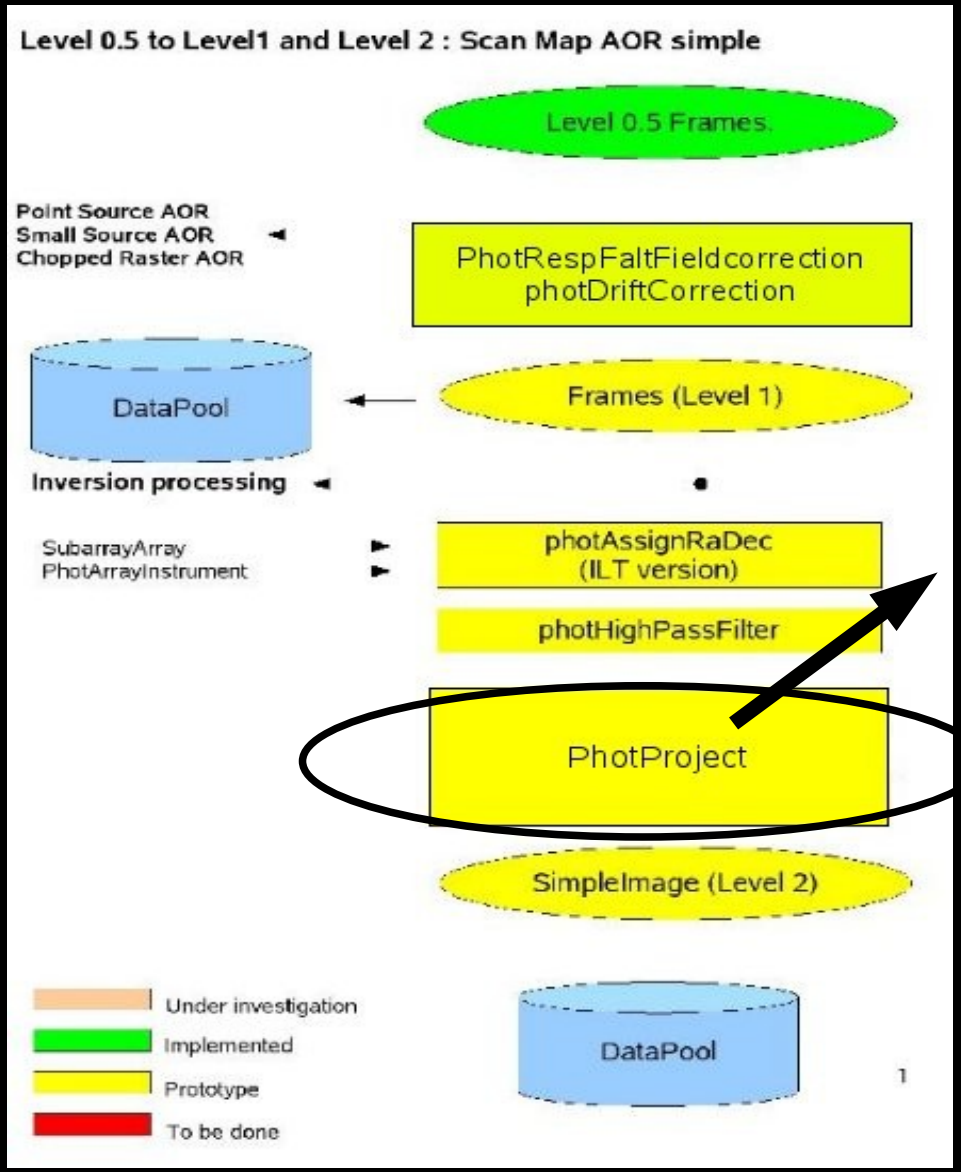


First Problem: 'Image sharpening'

High-Pass Filter:

- filtering away low frequency drift 1/f and let the high frequencies band pass
- the detail of the image is kept while the larger scale gradients are removed: (cutoff frequency?)

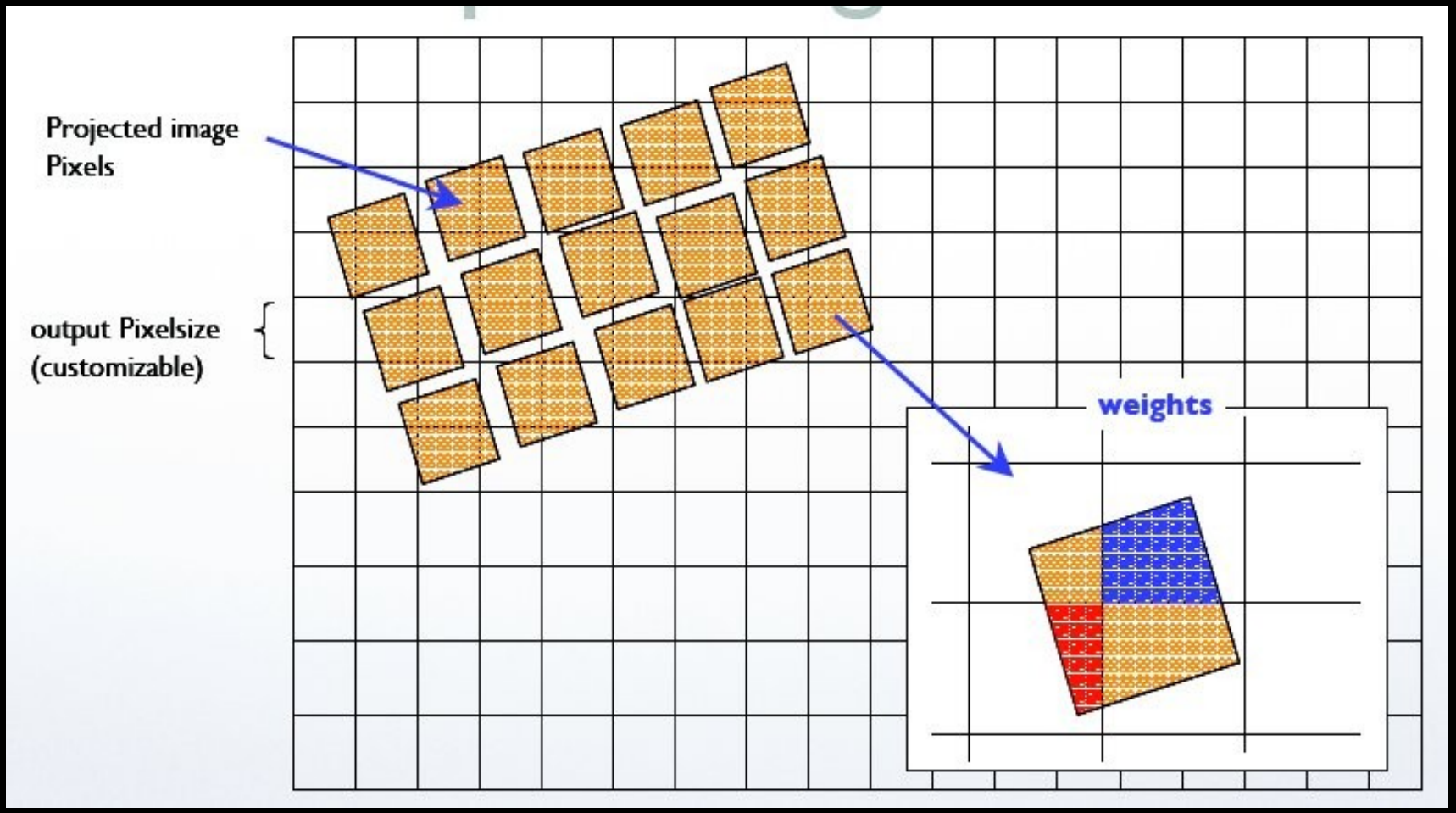
PhotProject



- data cube processed with simple geometrical projection:

$$W'_{xy} = a_{xy} w_{xy} + W_{xy}$$
$$I'_{xy} = \frac{a_{xy} i_{xy} w_{xy} + I_{xy} W_{xy}}{W'_{xy}}$$

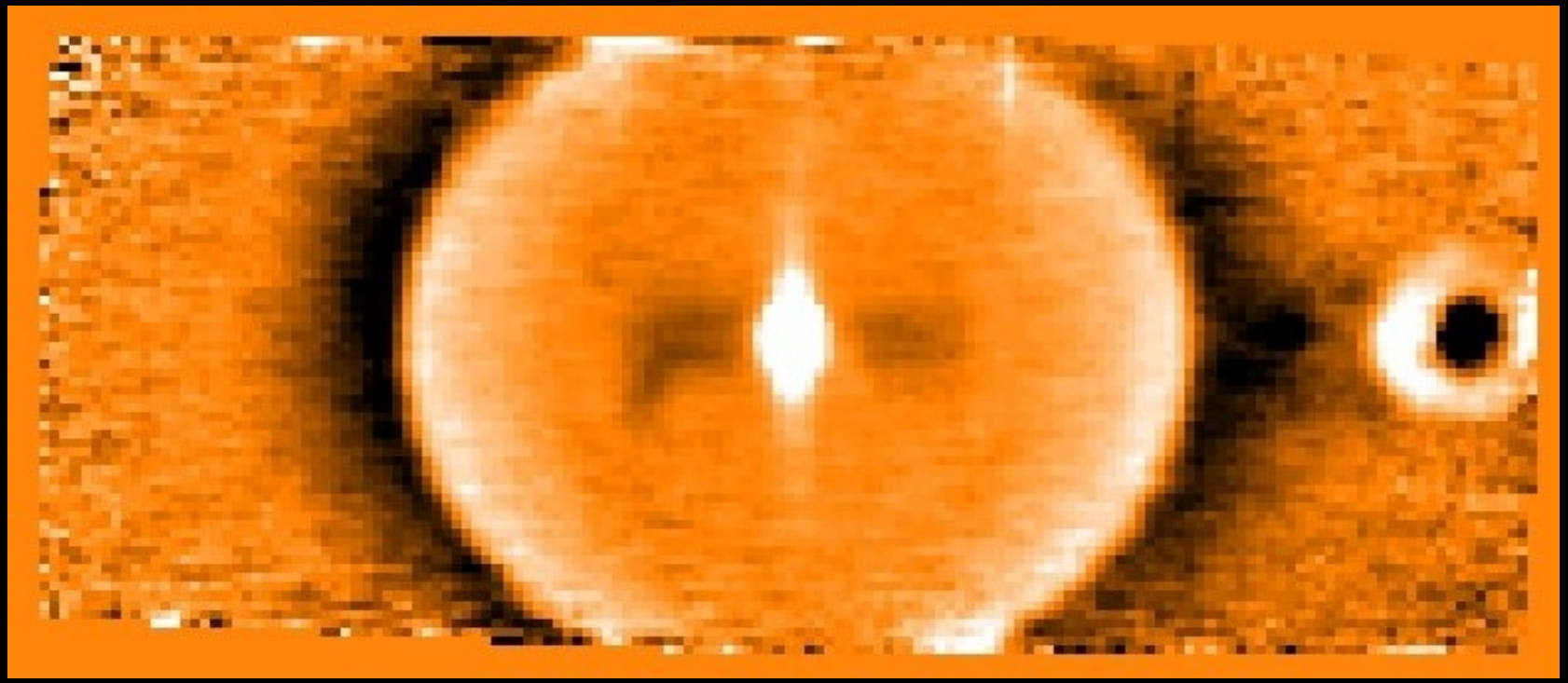
Tested on extended sources, raster data and scan map data



$$W'_{xy} = a_{xy} w_{xy} + W_{xy}$$
$$I'_{xy} = \frac{a_{xy} i_{xy} w_{xy} + I_{xy} W_{xy}}{W'_{xy}}$$

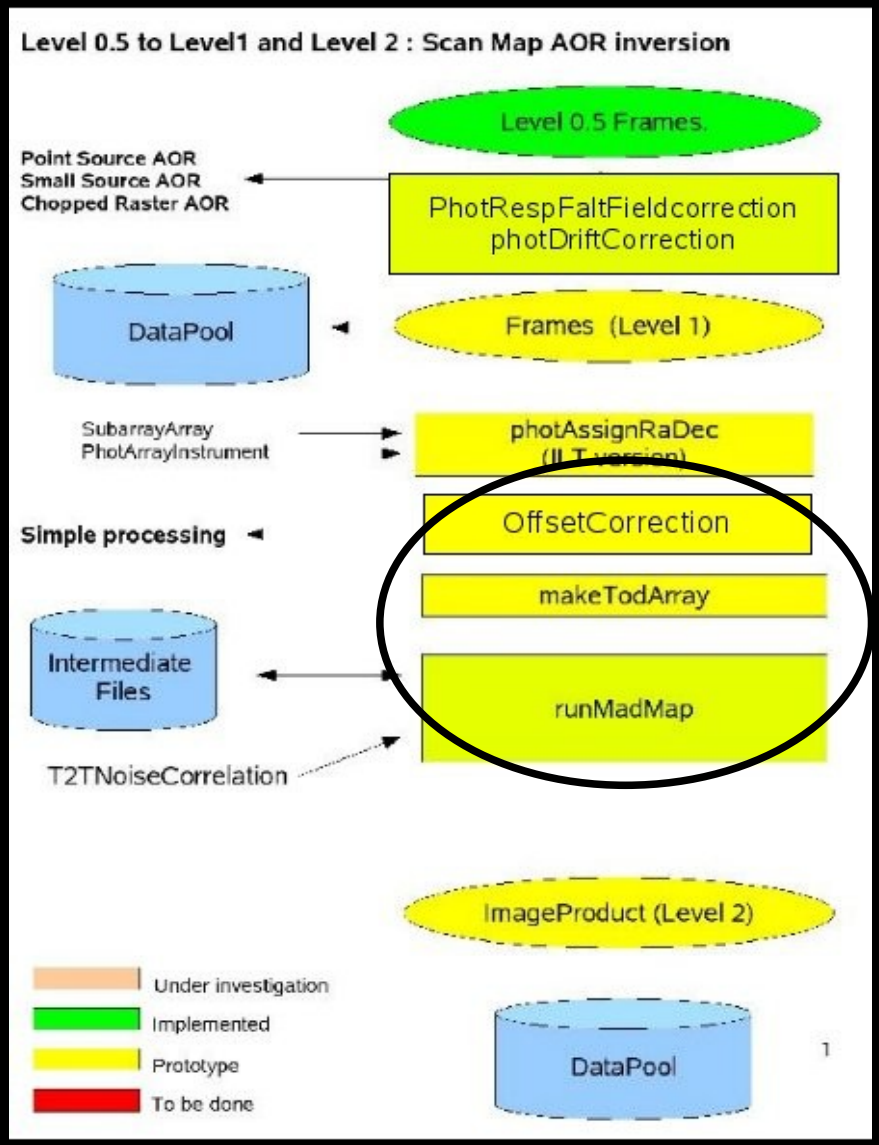
- only geometrical weights in the current version: w_{xy}=1 (w_{xy} = 0 for flagged pixels) in the scan map
- Exposure map based weights ($0 < w_{xy} < 1$) only for point/small source AOT

PhotProject Level 2 result



PhotProject result based on PACS test data:
output consists of map, noise map, coverage map

Pipeline MadMap



- Original C MadMap version (Wmap data)
- MADmap uses a maximum-likelihood technique to build a map from an input Time Order Data (TOD) set by solving a system of linear equations.
- It is used to remove low-frequency drift ("1/f") noise from bolometer data while recovering extended source flux
- Offset correction (Median filter) needed! detector signals need to be put on a similar scale before MADmap processing

Pipeline MadMap

• TOD:

$$d = n + As$$

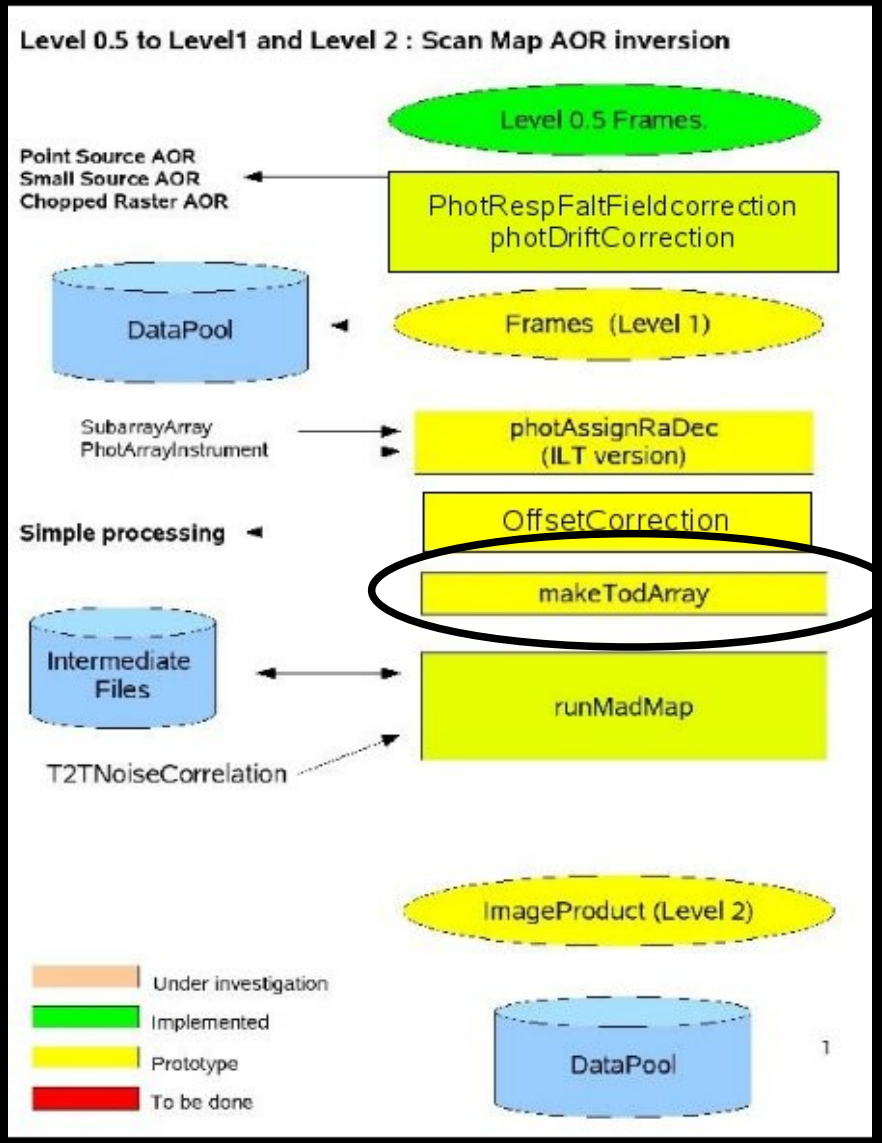
• where d is the time ordered data set, n is a time domain vector of piece wise stationary Gaussian noise, s is the signal in some basis other than the time domain, and A is the pointing matrix which is the linear operator which projects from the signal basis to the time domain. Once the data is formulated in this way the maximum likelihood map can be derived with MADmap.

•MADmap solves the map making equation:

$$m = \left(A^\top N^{-1} A\right)^{-1} A^\top N^{-1} d$$

•A is the n_t by n_p pointing matrix where n_t is the number of time samples and n_p is the number of pixels. N is the time time noise covariance matrix. d is the time ordered data set (the observation with noise). m is the pixel domain maximum likelihood estimate of the noiseless signal map given N and d.

Pipeline MadMap



- The implementation is currently based on two JAVA classes:
- MakeTodArray:

→ Builds time-ordered data (TOD) stream for input into MADmap

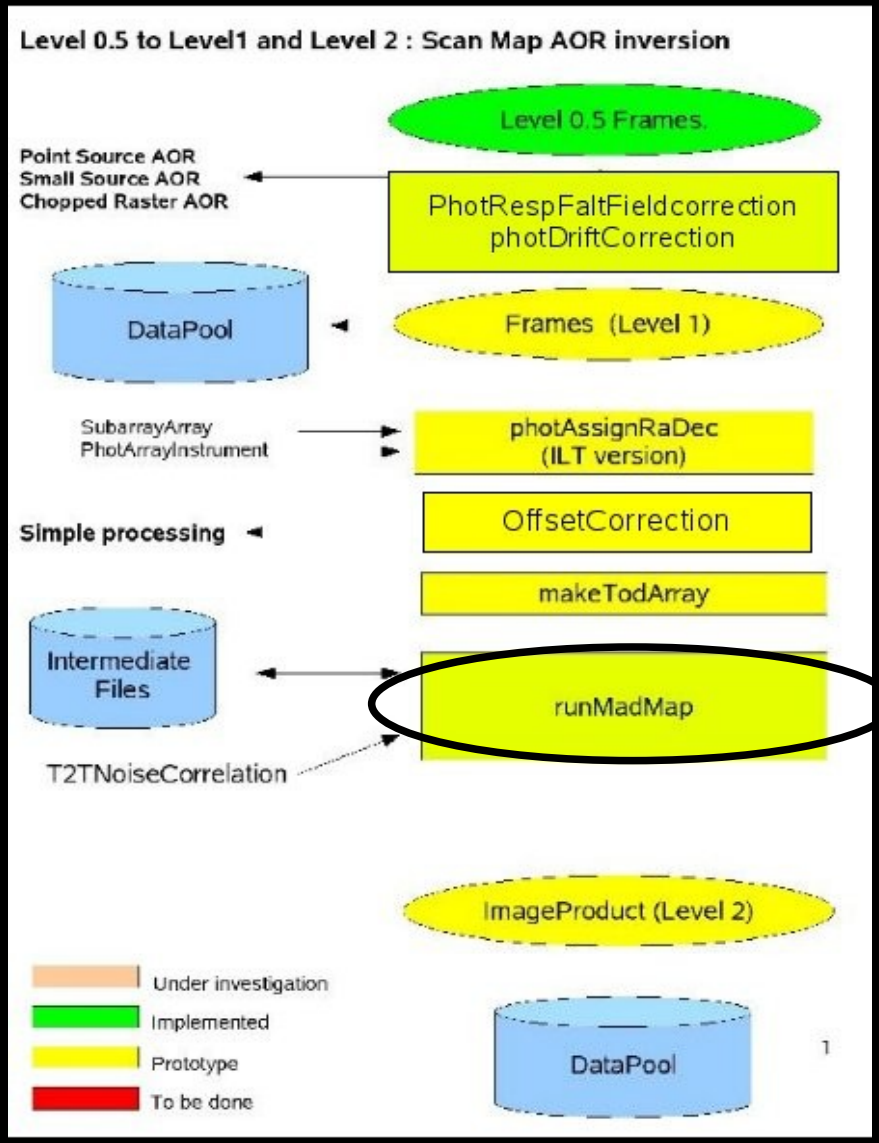
→ N, the time time noise covariance matrix, is given in a calibration file

→ in future version N will be calculated from input data?

→ MadCap routine not implemented

in java yet

Pipeline MadMap



- The implementation is currently based on two JAVA classes:
- RunMadMap: wrapper that runs MadMap module

→ Output product consisting of multiple images:

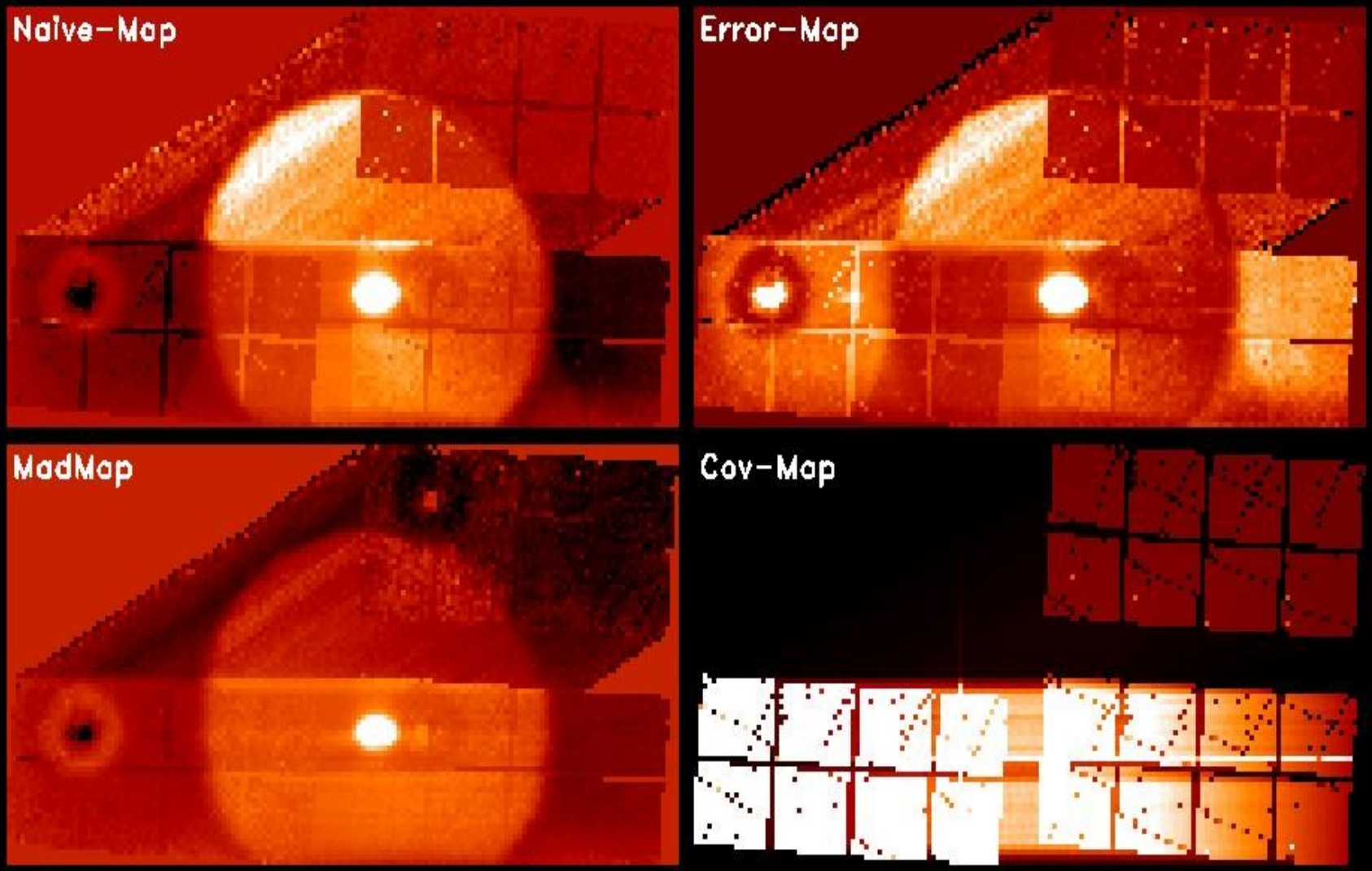
(1) map -- Sky map image

(2) naive map -- Sky map without corrections

(3) coverage map

(4) noise image

Pipeline MadMap



Results based on PACS test data

PhotProject vs. MadMap

PhotProject

- can run on chunked data without any problem

MadMap

- can run on chunked data without caution

PhotProject vs. MadMap

PhotProject

- can run on chunked data without any problem
- does not mind to used cross scan legs simultaneusly

MadMap

- can run on chunked data without caution
- requires to used cross scan legs simultaneusly
  → this implies memory problem (not feasible for large maps)

PhotProject vs. MadMap

PhotProject

- can run on chunked data without any problem
- does not mind to used cross scan legs simultaneusly

- needs data filtering for image sharpening (still under investigation)

MadMap

- can run on chunked data without caution
- requires to used cross scan legs simultaneusly
  → this implies memory problem (not feasible for large maps)
- needs to remove background offsets

PhotProject vs. MadMap

PhotProject

- can run on chunked data without any problem
- does not mind to used cross scan legs simultaneusly

- needs data filtering for image sharpening (still under investigation)
- appropriate for cosmological sourveys (point sources no structures on large scale)

MadMap

- can run on chunked data without caution
- requires to used cross scan legs simultaneusly
  → this implies memory problem (not feasible for large maps)
- needs to remove background offsets
- appropriate for extended sources (remove 1/f noise without removing large scale structures)

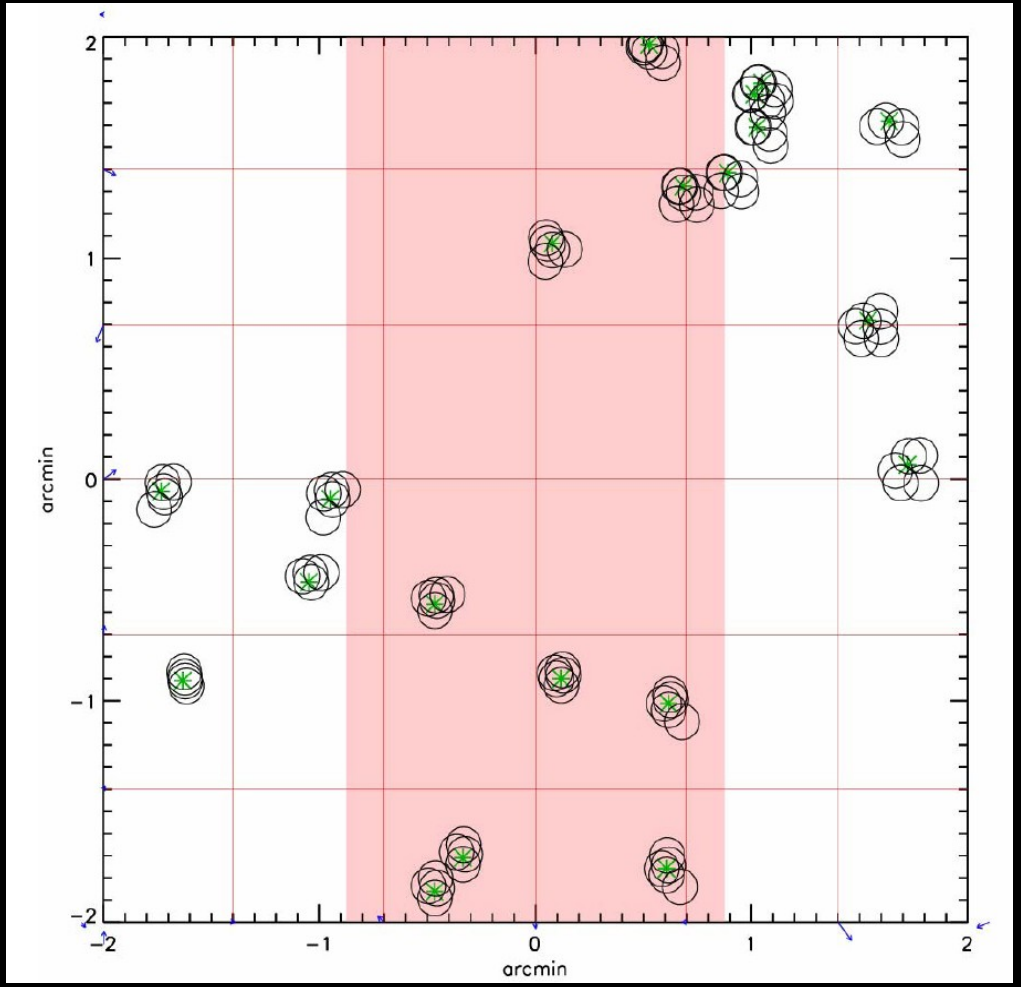
Open issues

- Pointing accuracy and re-centering

Open issues

- Pointing accuracy and re-centering

- ~6 arcsec FWHM point spread function at shortest wavelength
- ~2 arcsec 1sigma pointing accuracy (a posteriori)
- Offsets not stable over a long scanmap
- Smearing of the PSF
- recentering?



Open issues

- Pointing accuracy and re-centering

- Filtering:
  - how to filter the data
  - is High-Pass Filter a good option?
  - what about cross-correlated noise?

- data Chunking to reduce volume
  - Is that really necessary?
  - What's the consequences for MadMap?

Conclusions

- 80% of the pipeline modules are in place
- tests ongoing on real and simulated data
- huge effort on scientific validation
- still working on proper error propagation
- work in progress... but quite advanced stage ~4 months before launch