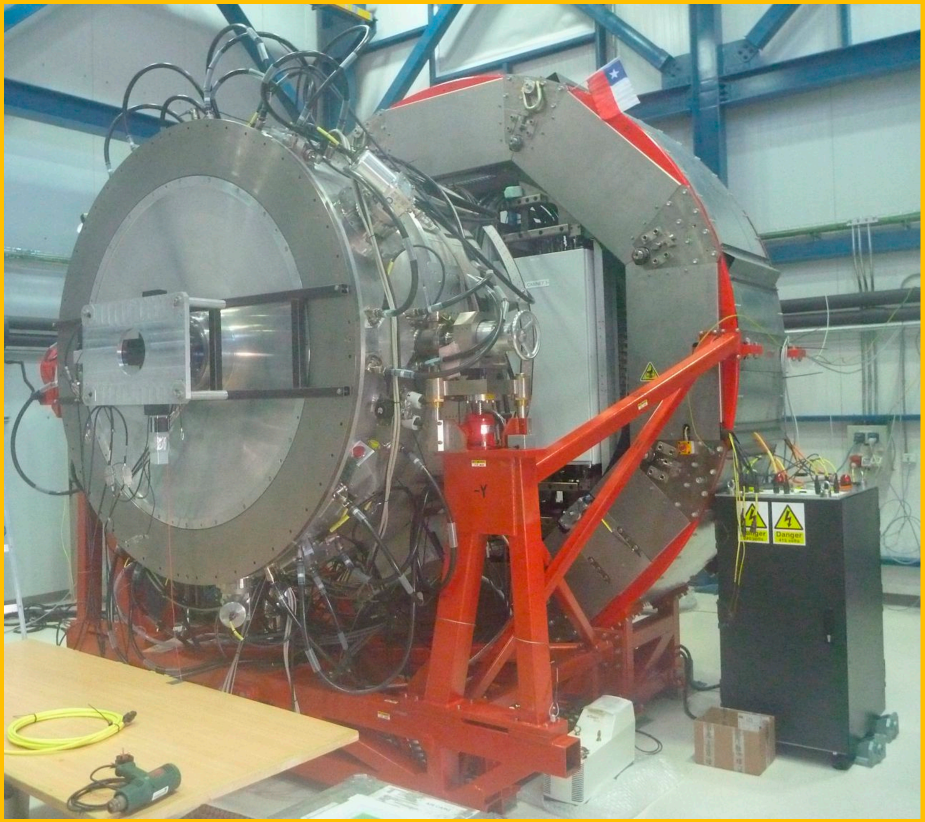


KMOS Pipeline Tutorial

Alex Agudo Berbel, Ric Davies, Natascha Förster-Schreiber, Erich Wiezorrek

26.10.2012

CONTENTS

What is KMOS?
- Some Pictures
- Specifications

Pipeline
- Data Format
- Flexure
- Calibration
- Reconstructing Data Cubes
- Combining Data Cubes

Observation Workflow at VLT
- Observation Modes

Software
- Data Viewers
- Karma
- Esorex
- Pipeline GUI

Recipes
- Calibration
- Overview

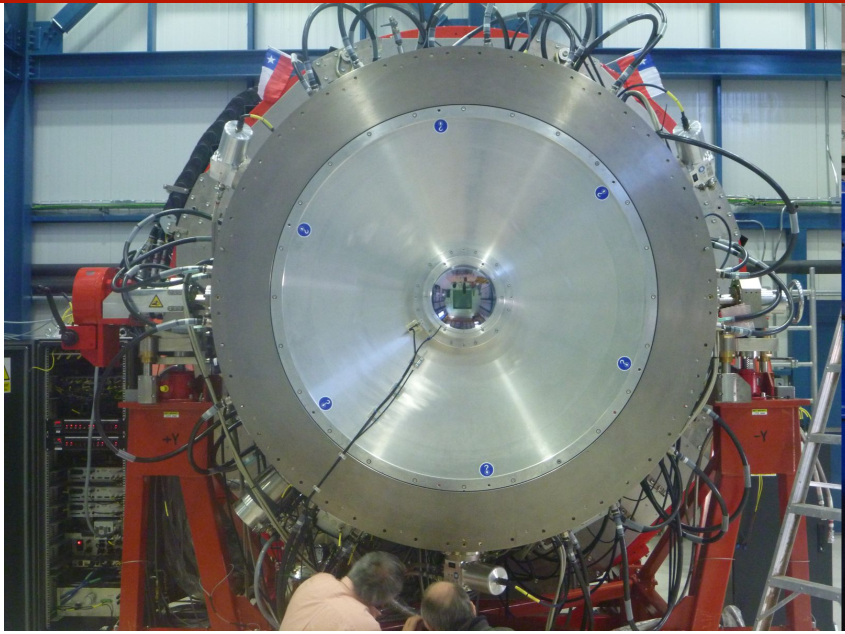
- **KMOS: K**-band **M**ultiple-**O**bject cryogenic integral field **S**pectrograph designed for intermediate resolution spectroscopy in the 0.8-2.5μm range

- First of its kind at a 8m class telescope
  - Similar project: SINFONI (single IFU)

- 24 robotic pickoff arms patrol a 7.2 arcmin diameter field each of which feeds 2.8x2.8 arcsec FoV sampled at 0.2 arcsec to an image slicing IFU

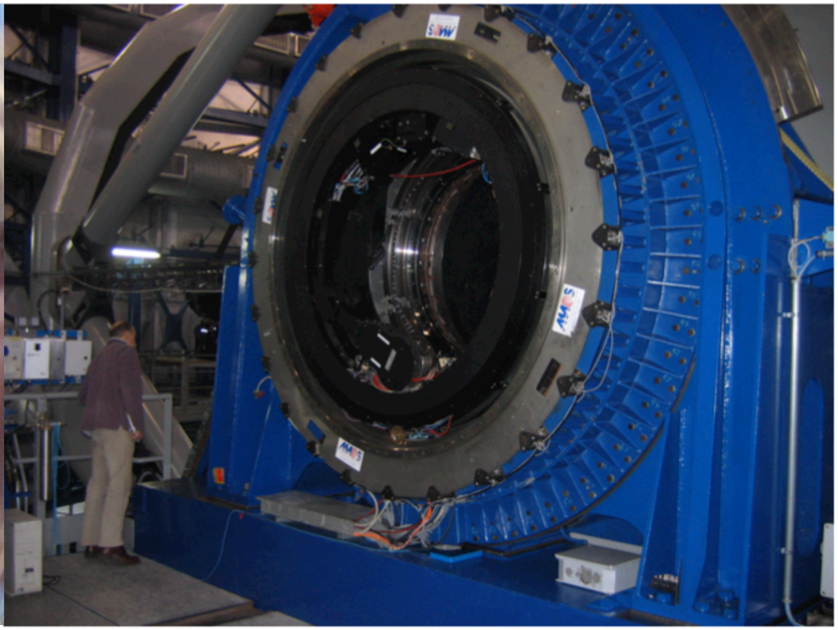
- The IFUs are consolidated in groups of 8 which feed one of 3 identical spectrographs providing R~3500 spectra in the H, HK, IZ, K & YJ bands

Outside KMOS

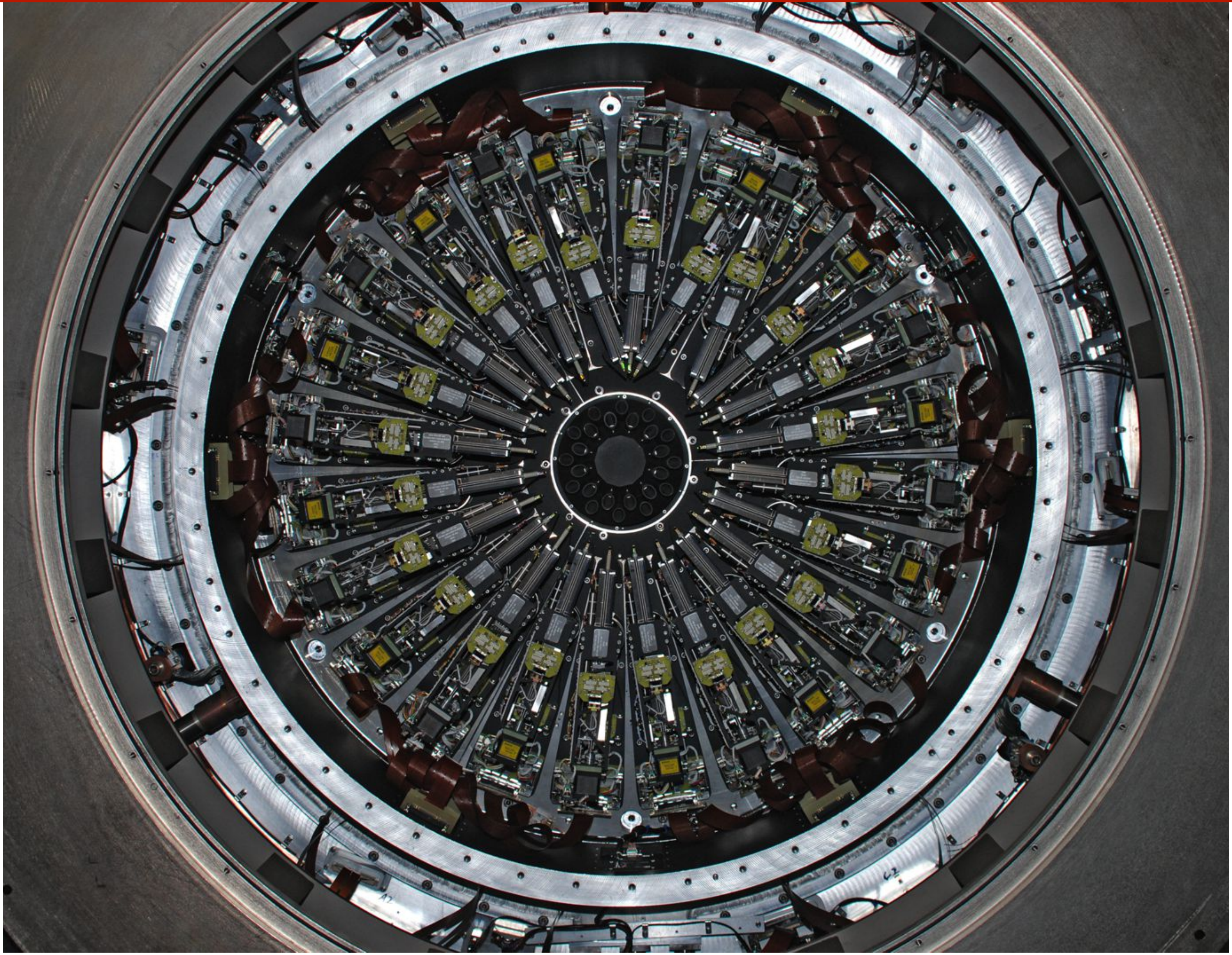

The front: cryostat, mounting plate & window


The back: cable derotator & electronic cabinets


All arms extracted

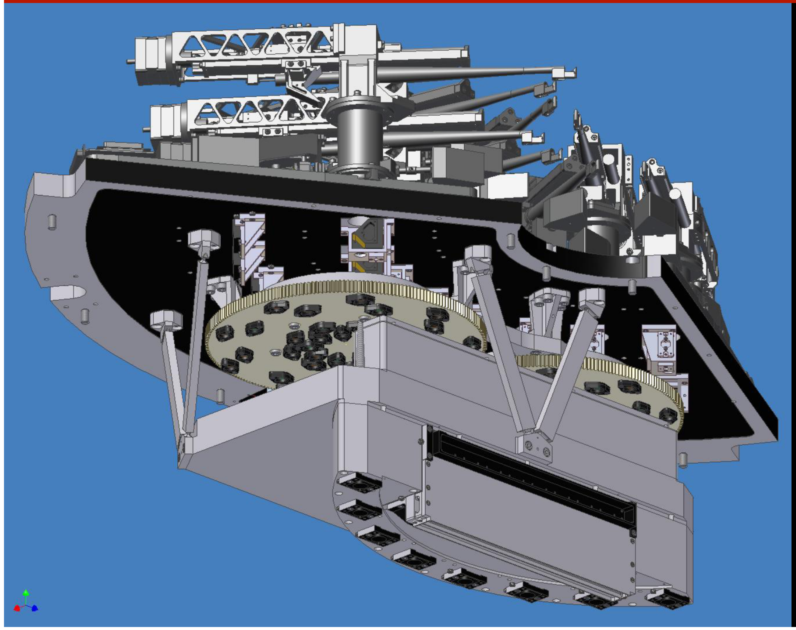

One of the Nasmyth focuses

Inside KMOS

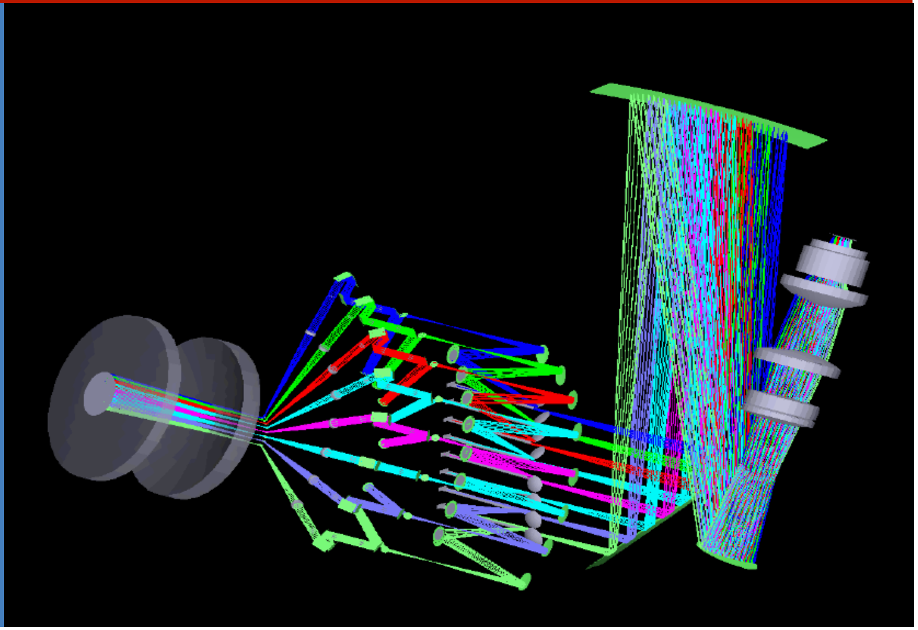


Inside the cryostat: the 24 robotic arms

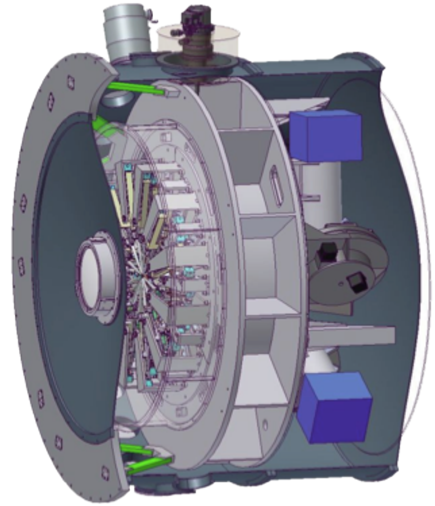
Inside KMOS



Pick-off arms, filter wheels, IFUs



Light path through the system (for 8 arms)



Pick-off arms, IFUs, spectrographs

Single pick-off arm

Set of mirrors (for 8 arms)

Key Specifications

General

- Rotating Mass 2.4 t
- Total Mass 9.5 t
- Operating temperature -140 °C
- Optical Throughput ~30%

Hardware

- 24 robotic pick-off arms
  - arranged in 2 layers
  - divided into 3 sections
- 3 spectrographs
- 3 Hawaii-2RG detectors 2048 x 2048 pix

Field of View, Resolution

- Telescope Patrol Field 7.2 arcmin
- IFU
  - FoV 2.8 arcsec
  - Size 14 x 14 pix
  - Pixel size 0.2 arcsec

Timeline

- Kick-off: 04.2004
- Final Design Review 07.2007
- Prelim. Acceptance Europe 04.2012
- 1st commissioning 11.2012

Wavelength

Total range 0.78 µm - 2.5 µm

- H-band: 1.425 - 1.867 µm
- HK-band: 1.460 - 2.410 µm
- IZ-band: 0.780 - 1.090 µm
- K-band: 1.925 - 2.500 µm
- YJ-band: 1.000 - 1.359 µm

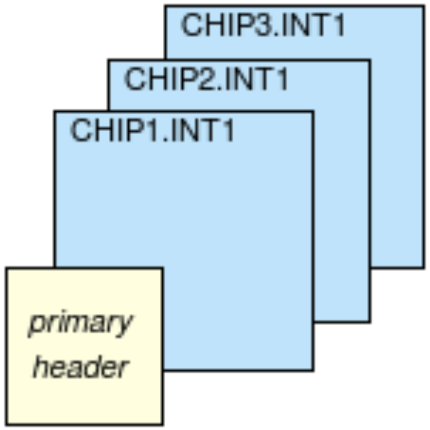
Spectral resolution

- H-band: R~4000
- HK-band: R~1900
- IZ-band: R~2800
- K-band: R~4100
- YJ-band: R~3400

Optical throughput

- H-band: ?
- HK-band: ?
- IZ-band: ?
- K-band: ?
- YJ-band: ? (to be measured on sky)

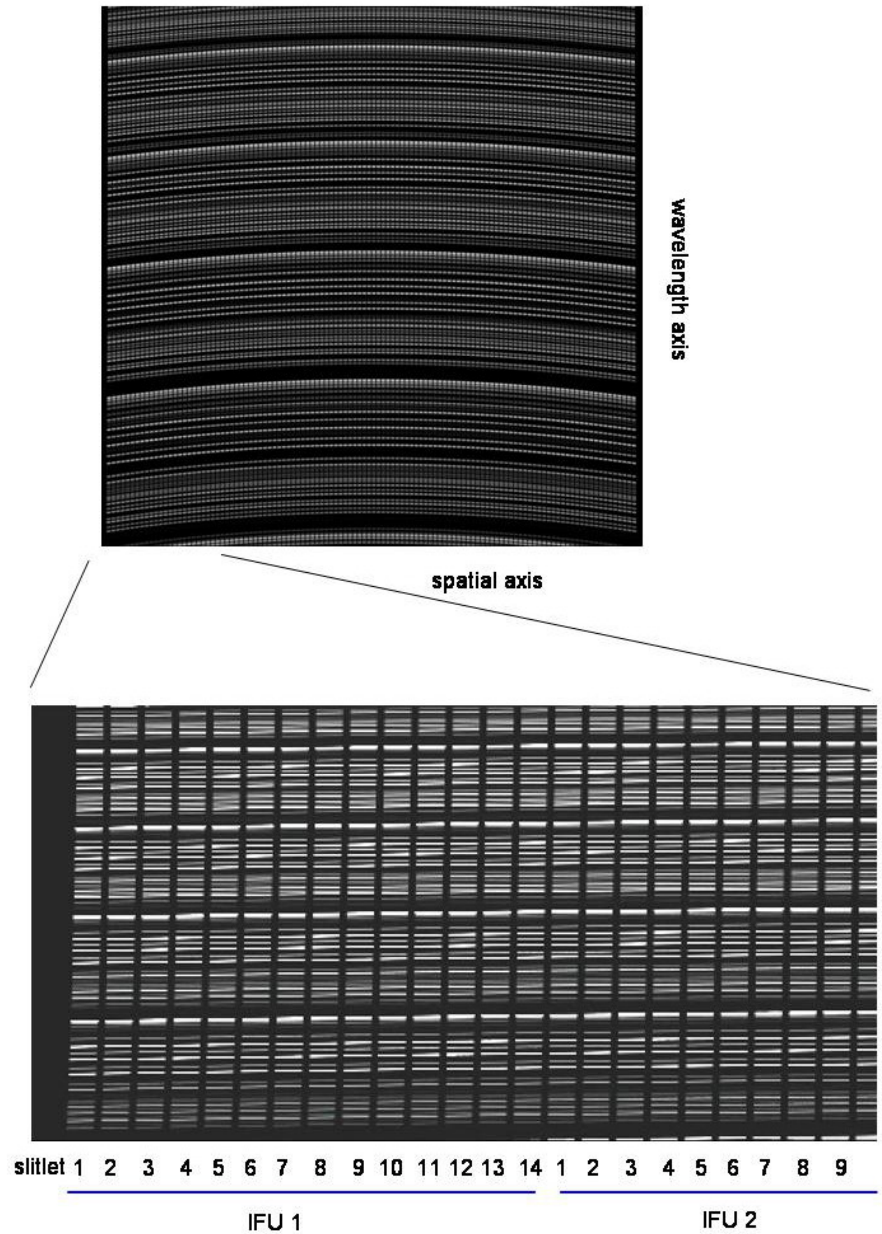
RAW detector frames



- Primary Header
  - Empty data section
- 3 data extensions

- 8 IFUs per detector
- Each IFU sliced into 14 slitlets, 14 pix width
- Each slitlet is a dispersed pseudo-longslit, 2040 pix length

- Exposure size: 48 MB
- 4 pix border around each frame reserved for detector readout electronics



Processed frames

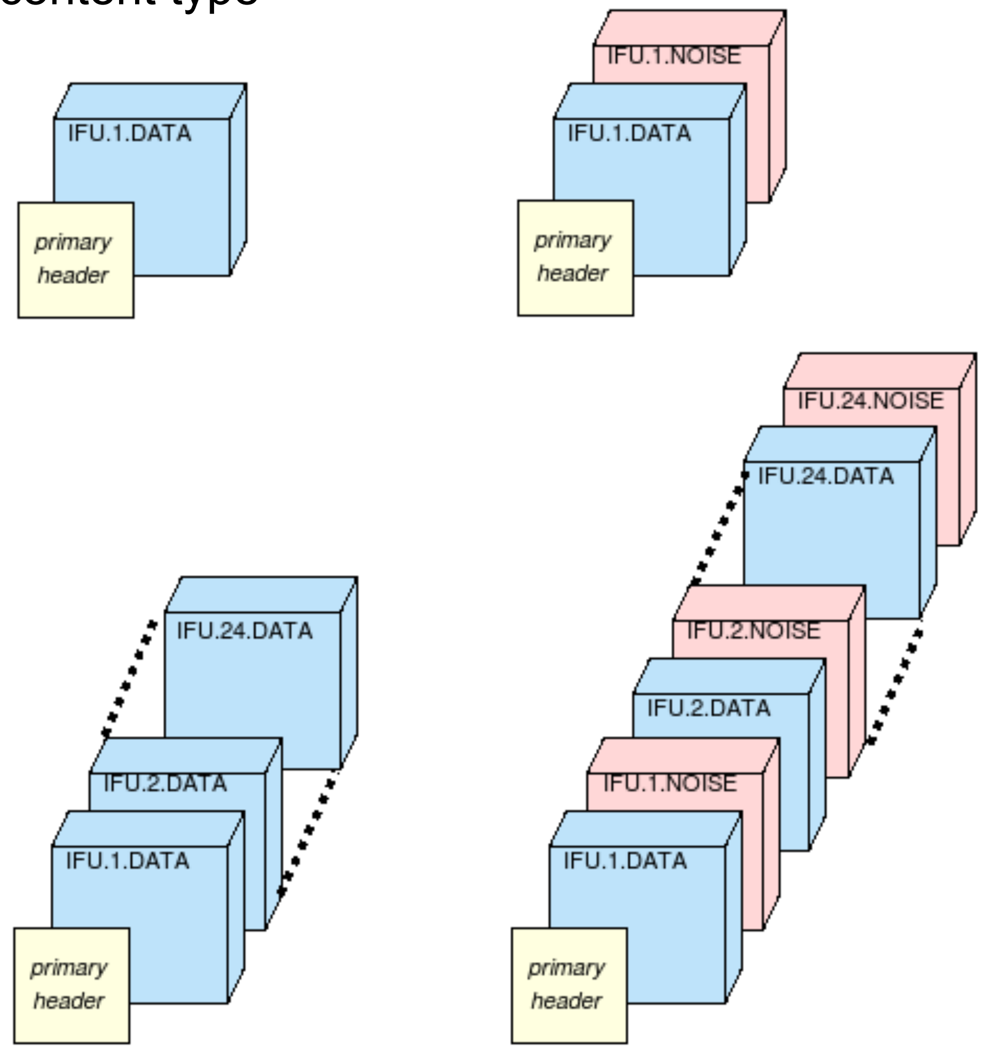
- Spectra, images, cubes are in extensions
- As a rule there are 1, 2, 24, or 48 extensions
- keyword EXTNAME indicates content type

F3I

F = floating point
B = binary

# dimensions
1 = spectrum
2 = image
3 = cube

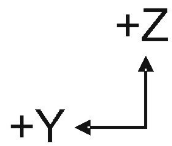
D = detector based
I = IFU based
S = any spectrum
L = any list



IFU orientation

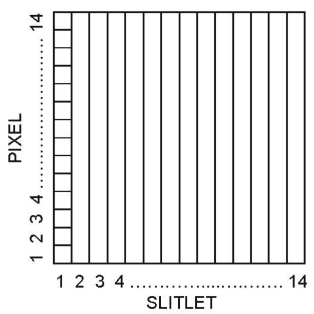
- Every 4 IFUs have a different orientation on the detector
- Calibration frames (as well raw frames) therefore miss a certain regularity

ON SKY

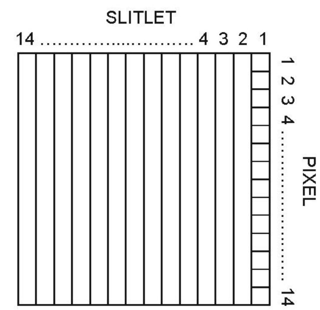


+Z corresponds to North and +Y to East when rotator offset angle is zero

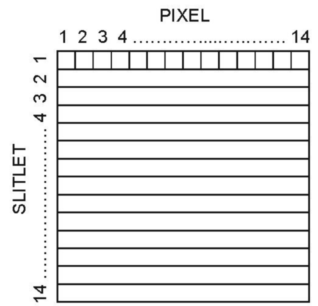
IFUs:
1, 2, 3, 4, 13, 14, 15, 16



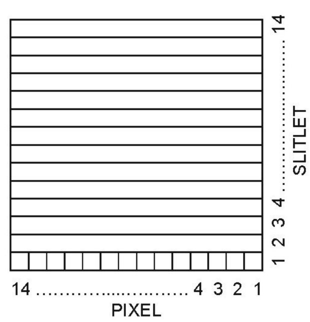
IFUs:
5, 6, 7, 8, 9, 10, 11, 12



IFUs:
17, 18, 19, 20

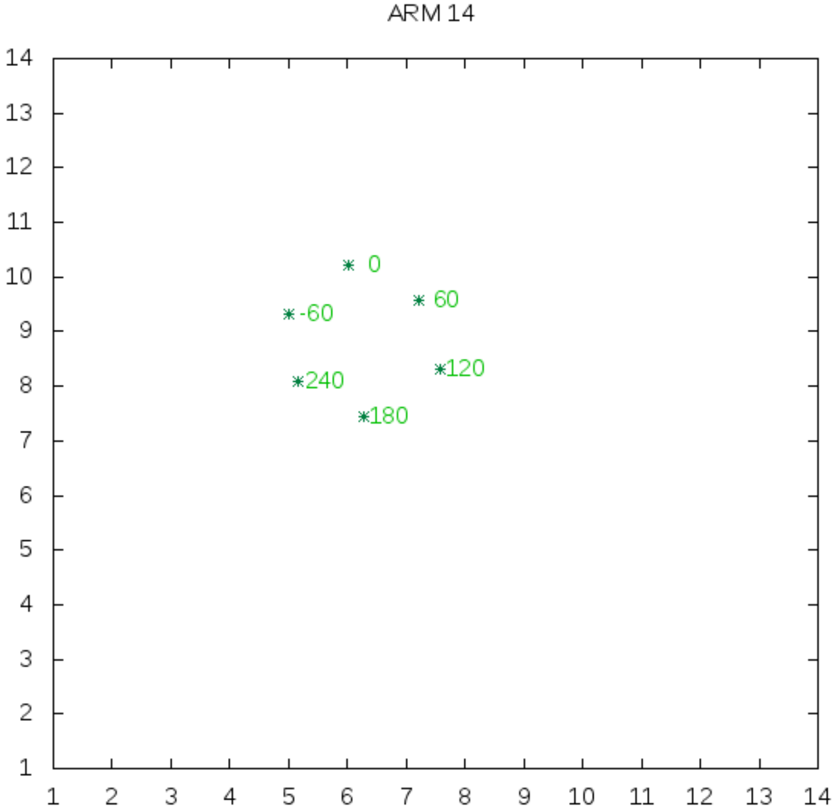


IFUs:
21, 22, 23, 24



Flexure

- Calibrations are done at 6 rotator angles (0, 60, 120, 180, 240, 300 deg)
- Measuring Flexure:
    - Place LED in the middle of the IFU
    - 6 exposures at different rotator angles
    - Reconstruct & measure centers



→ Whole instrument sags by a few microns wrt the optical axis

Calibration multitude

- 6 rotator angles @ 5 bands

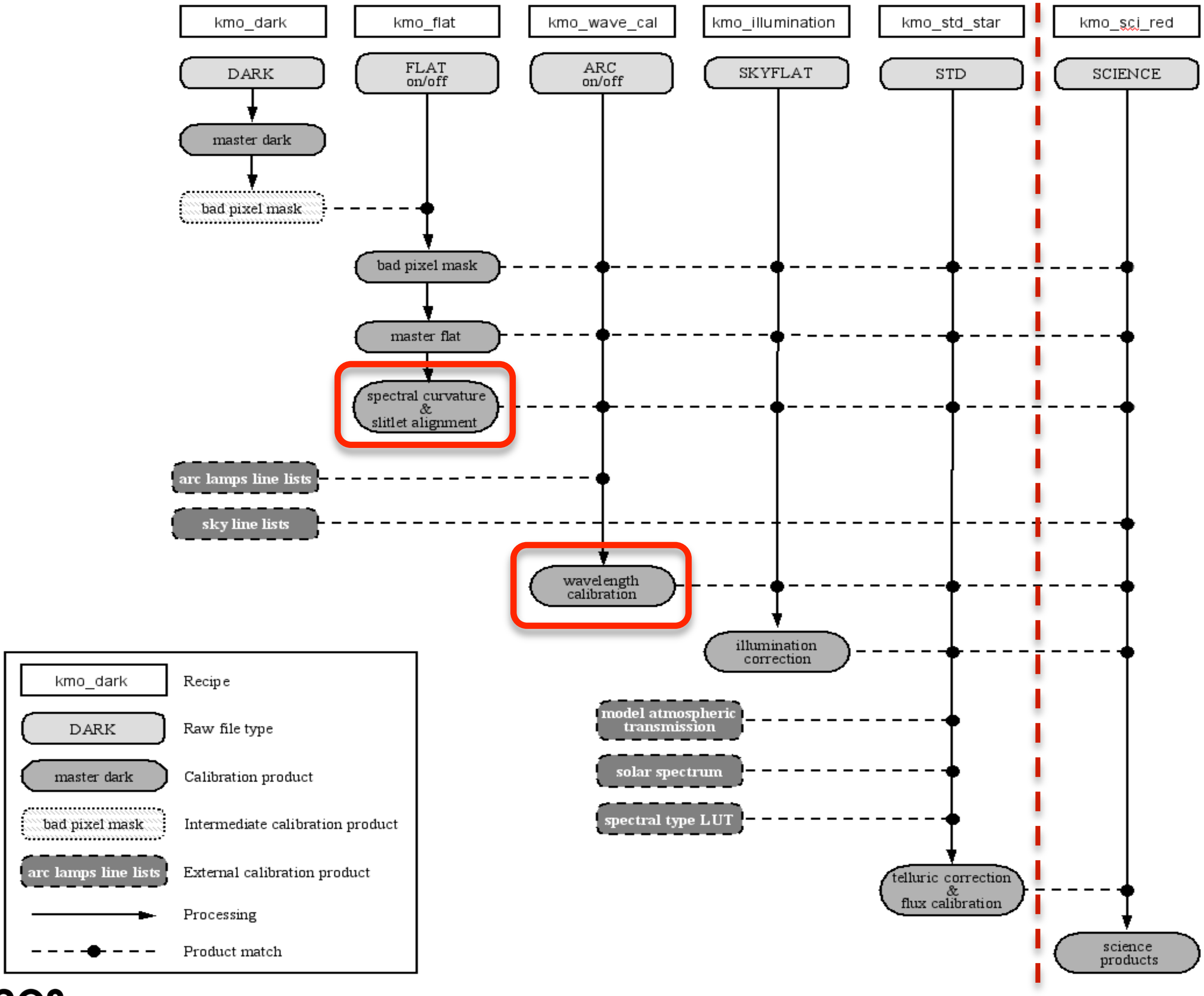
→ 30 sets of calibration frames

→ 8.8 GB calibration data for 200 files

- Calibration recipes ignoring bands and rotator angles:
    - kmo_dark
    - kmo_illumination

KMOS MPE

Calibration Cascade



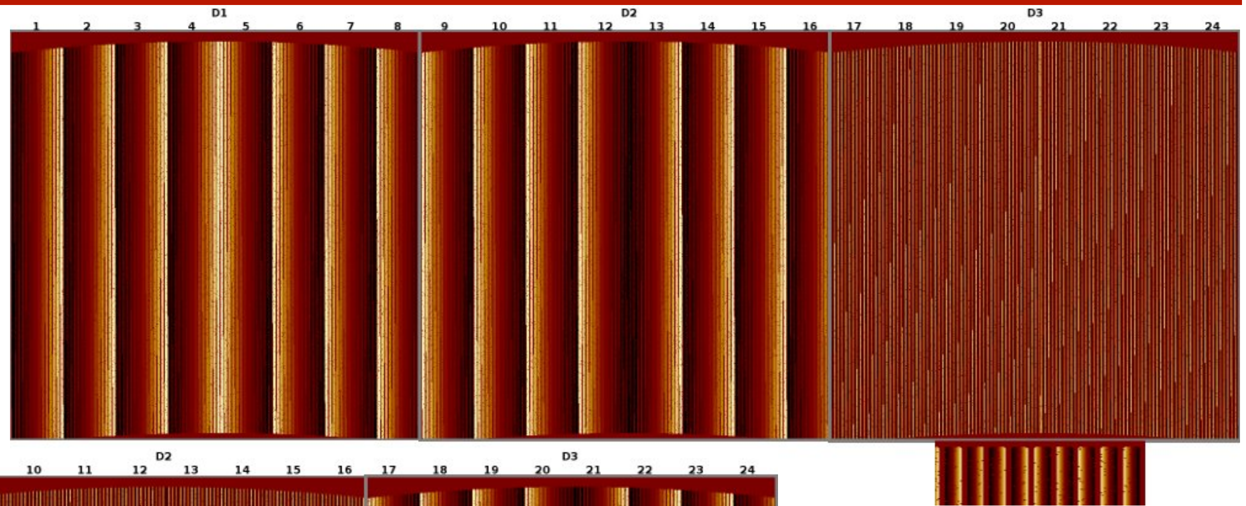
kmos

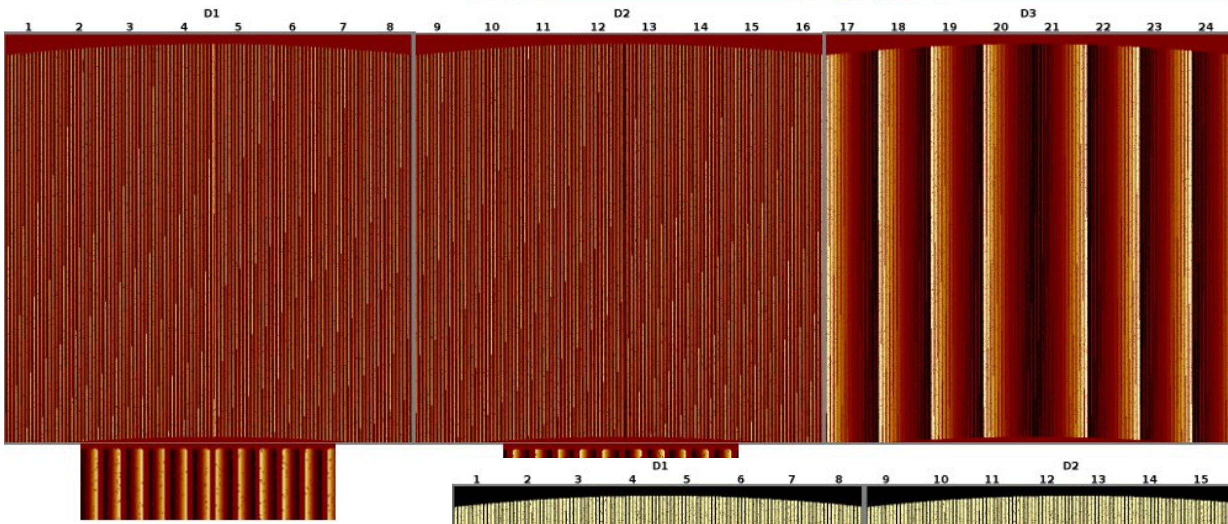
MPE

Calibration Concept: Spatial & Spectral Frames act as Lookup Tables (LUTs)

kmo_flat

XCAL frame

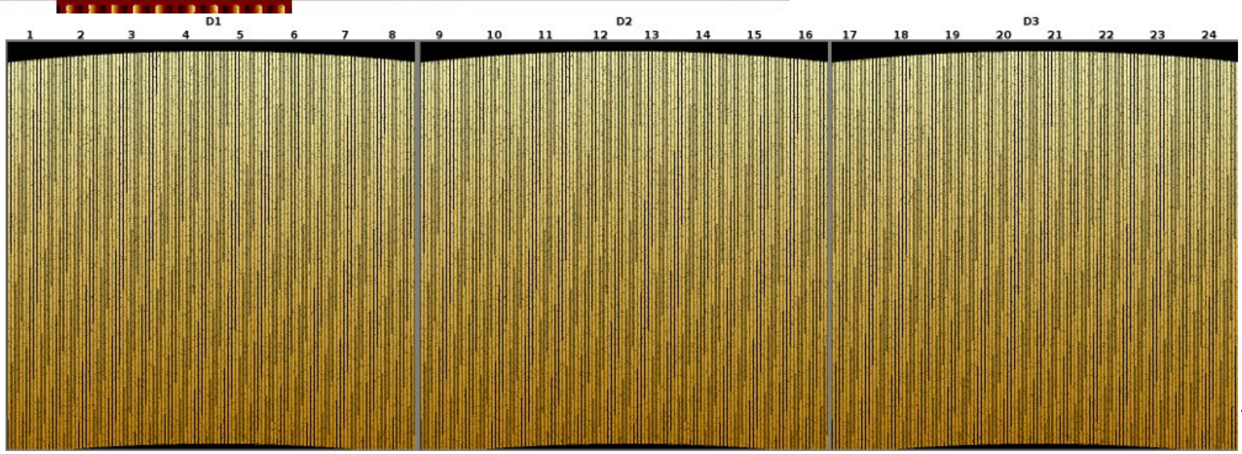




YCAL frame

kmo_wave_cal

LCAL frame

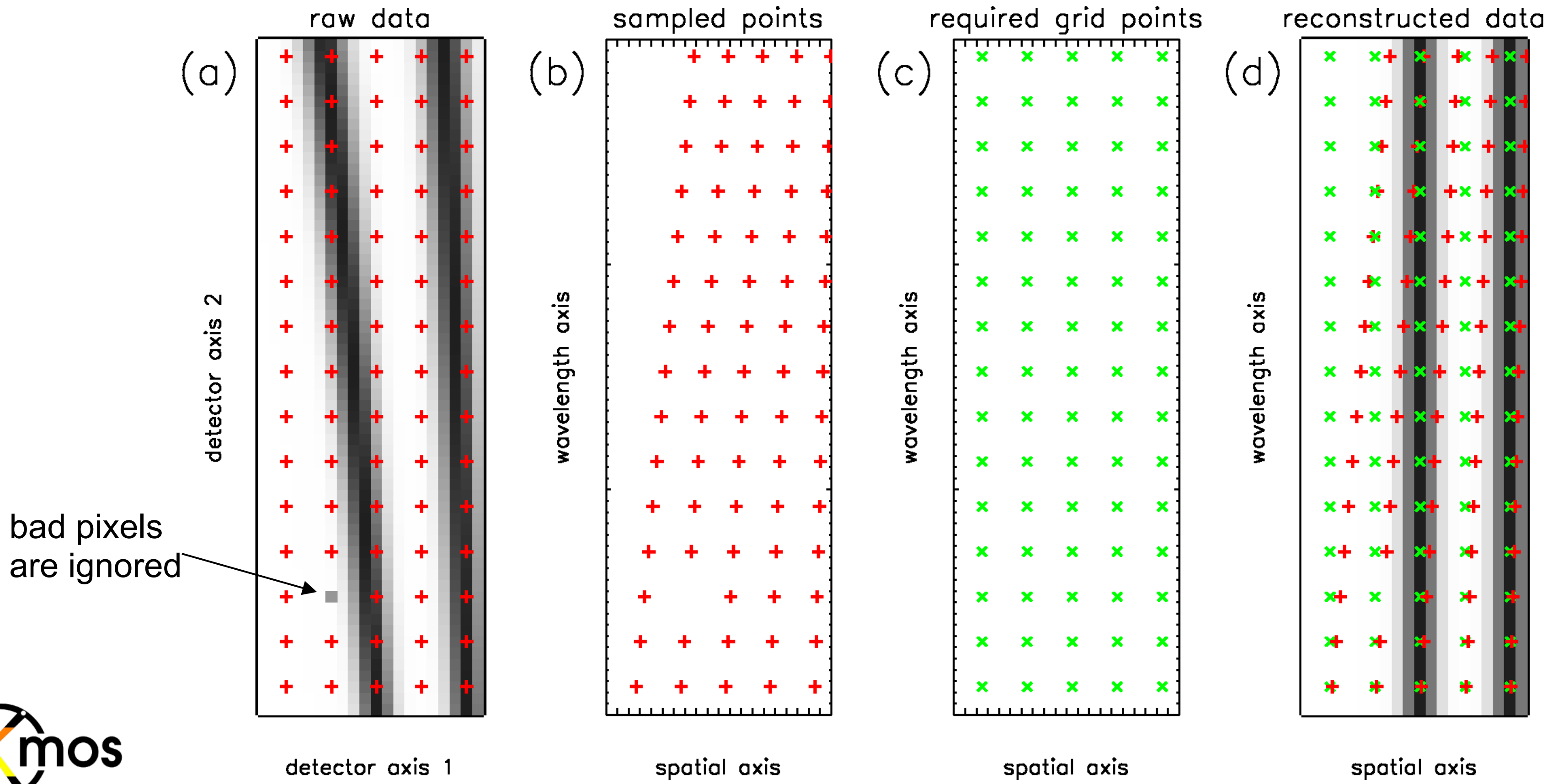


Interpolation

Final frame – *regularly* gridded x,y,λ positions where we want to know data values
Calibration frames – LUTs for *irregularly* spaced x,y,λ of each pixel on detector
Detector frame – data values for these *irregularly* spaced x,y,λ sampling positions

Create a list of sample points so that the data is dissociated totally from the detector frame.
i.e. the observations & calibrations simply give you: $value_0, x_0, y_0, \lambda_0$
...
$value_n, x_n, y_n, \lambda_n$

Each point in final frame is interpolated from sampled points in its local neighbourhood

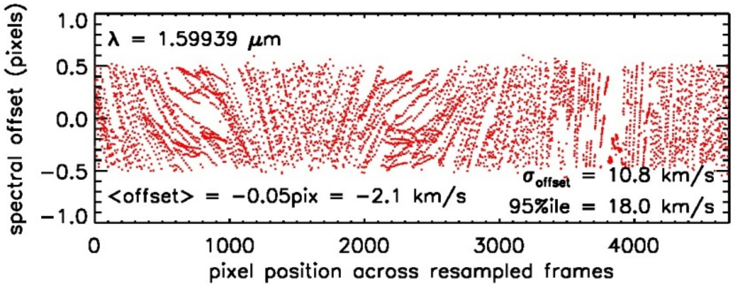


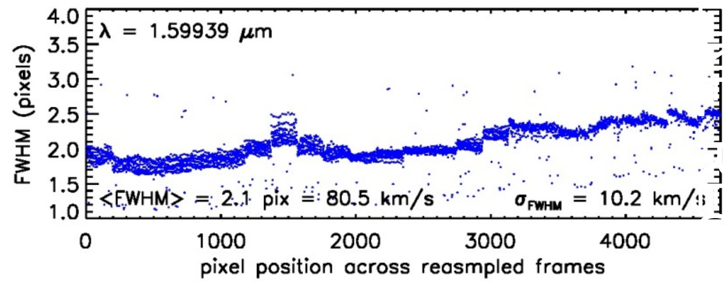
Comparison of Interpolation methods

The first three are true 3D methods, and are all based on weighted sums of the surrounding pixels.
The fourth method is the only one that uses real interpolation, and is a sequence of 1D interpolations.

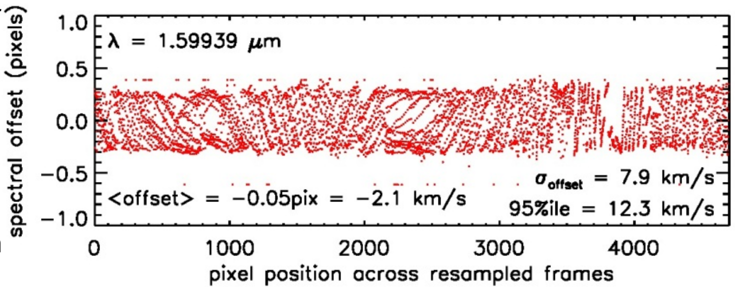
H-band (for a line @1.599μm, pixel ~820)

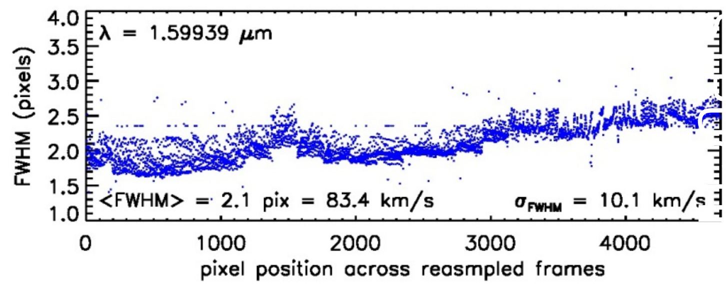
Nearest neighbour:
Offset up to 0.5 pixels since this is how far away the neighbour can be; the advantage is that one does not interpolate, but just re-sorts the measured data.



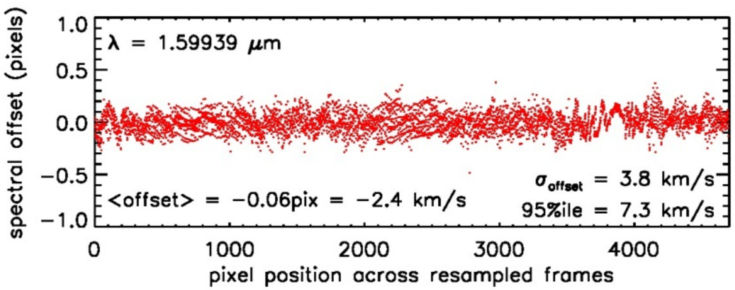


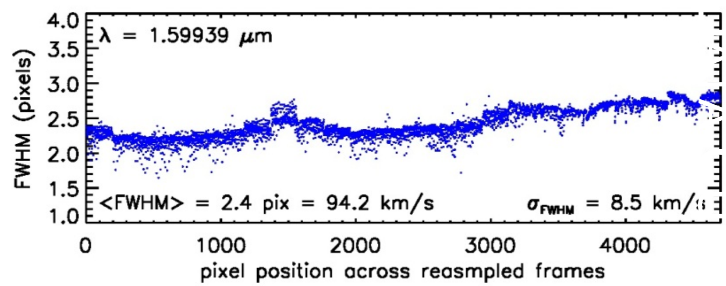
Modified Shepherd:
This can be a good method *if* the spatial/spectral resolution elements are well enough sampled (e.g. if one combines data from many exposures *during* the reconstruction). I expect drizzling to yield results much like this.



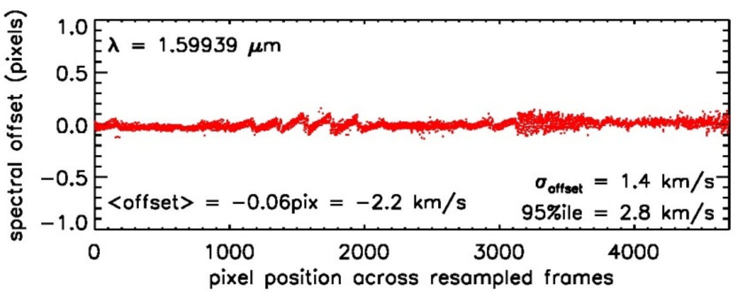


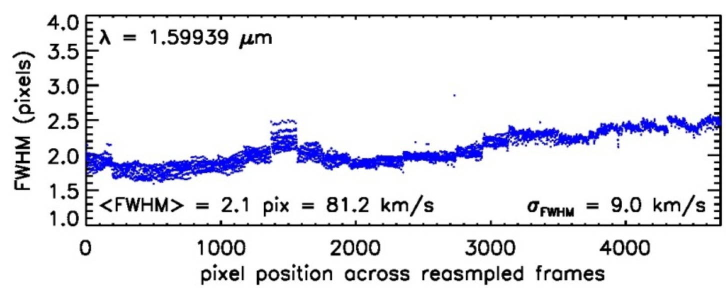
Linear Distance Weighting:
This is a simple method with simple error propagation, which yields reasonable results (which are within specification).





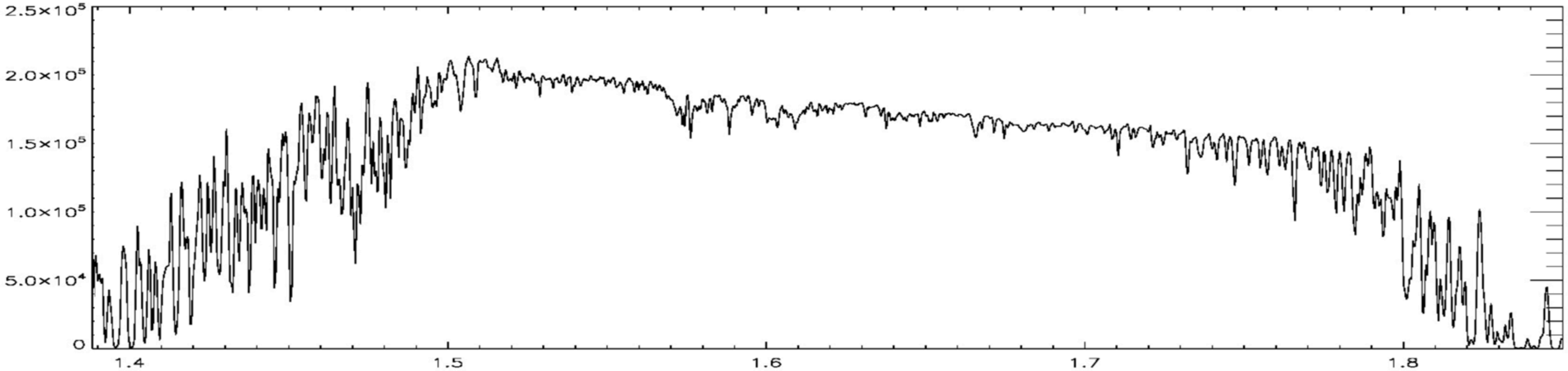
Cubic Spline:
(this uses cubic spline in spatial direction & polynomial in spectral direction). It gives the best results (precision & maintaining resolution). But error propagation is non-trivial.



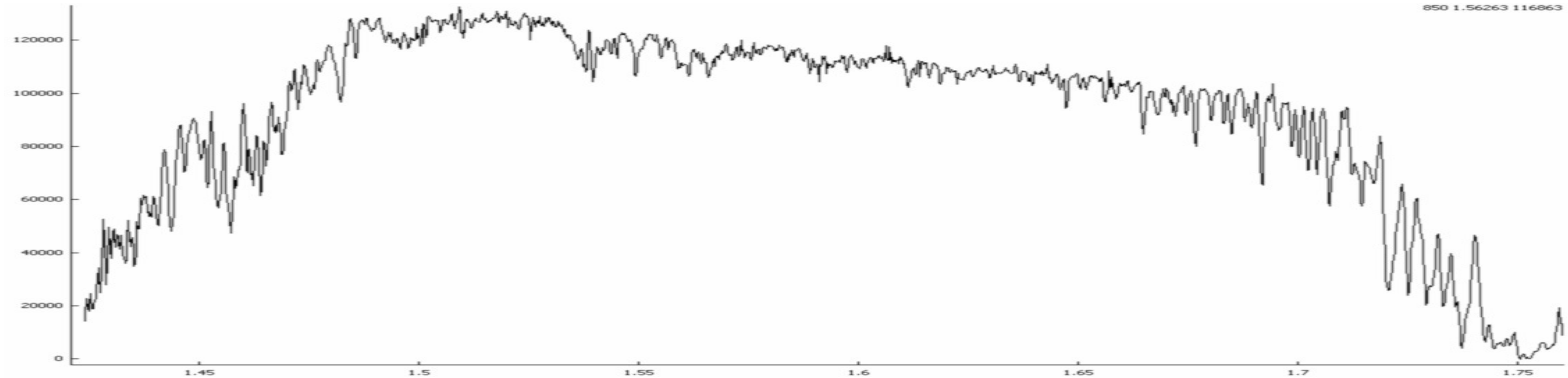


Comparison of Interpolation methods: Spectral Effects

Original spectrum

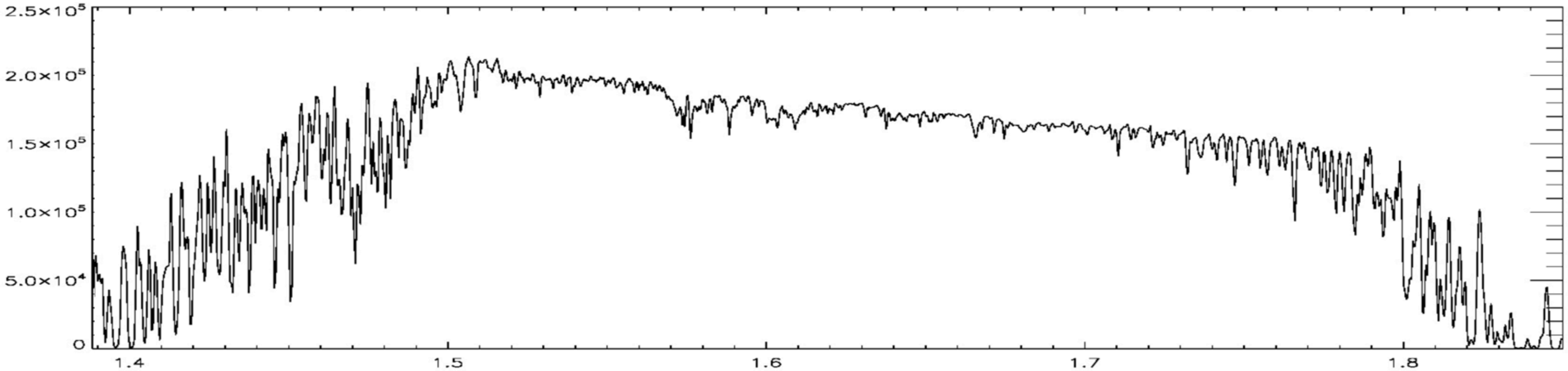


Cubic spline

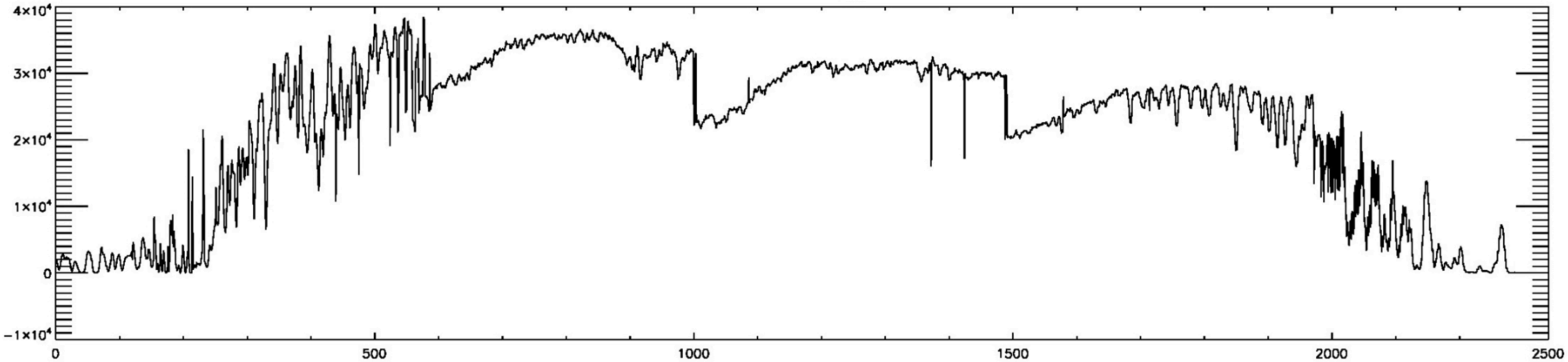


Comparison of Interpolation methods: Spectral Effects

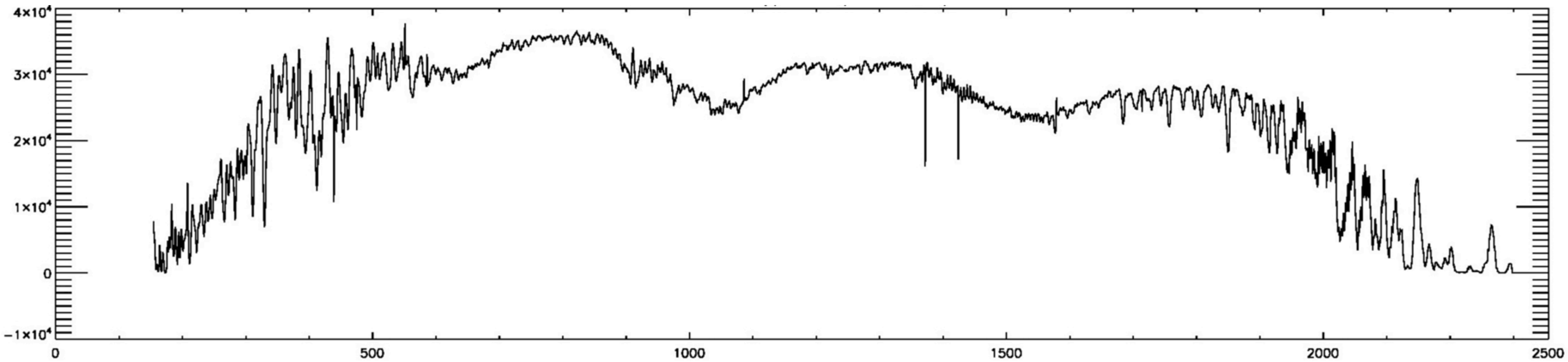
Original spectrum



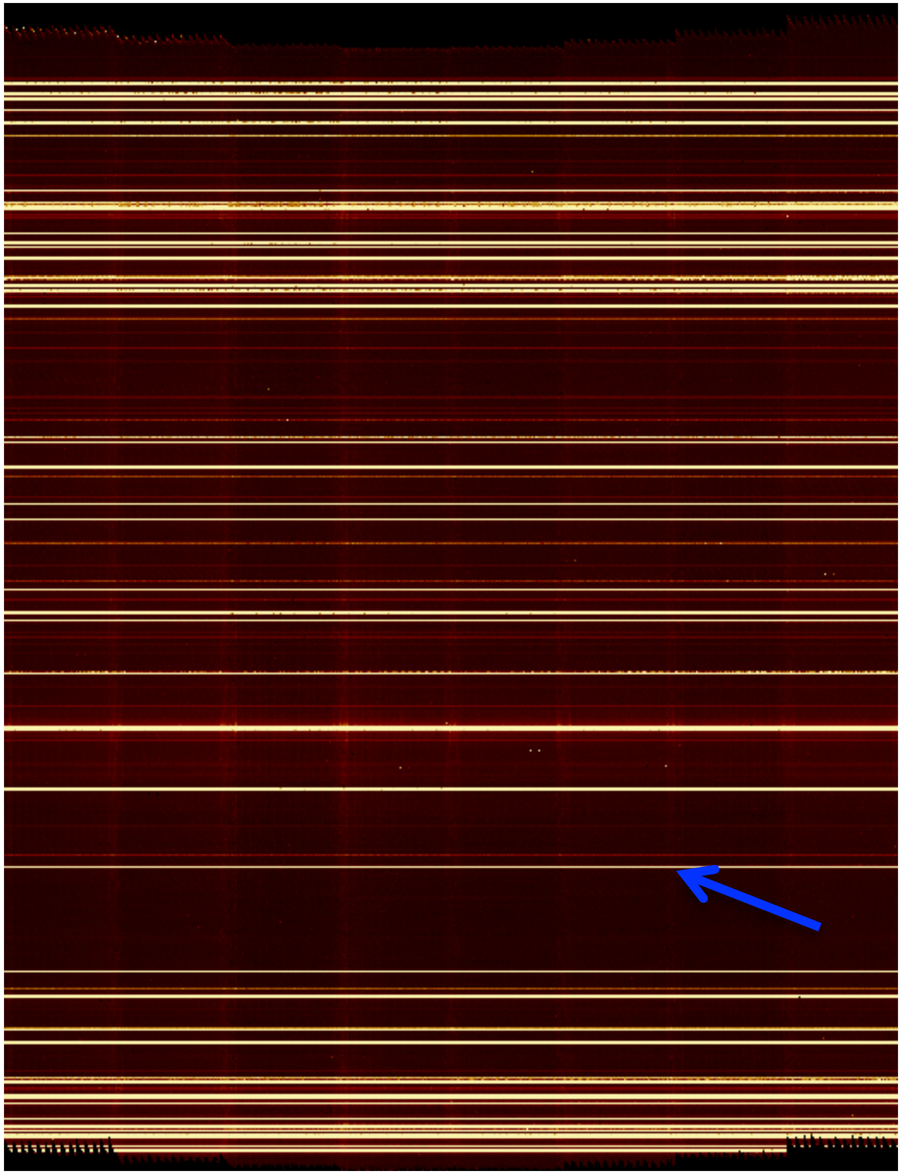
Nearest Neighbour



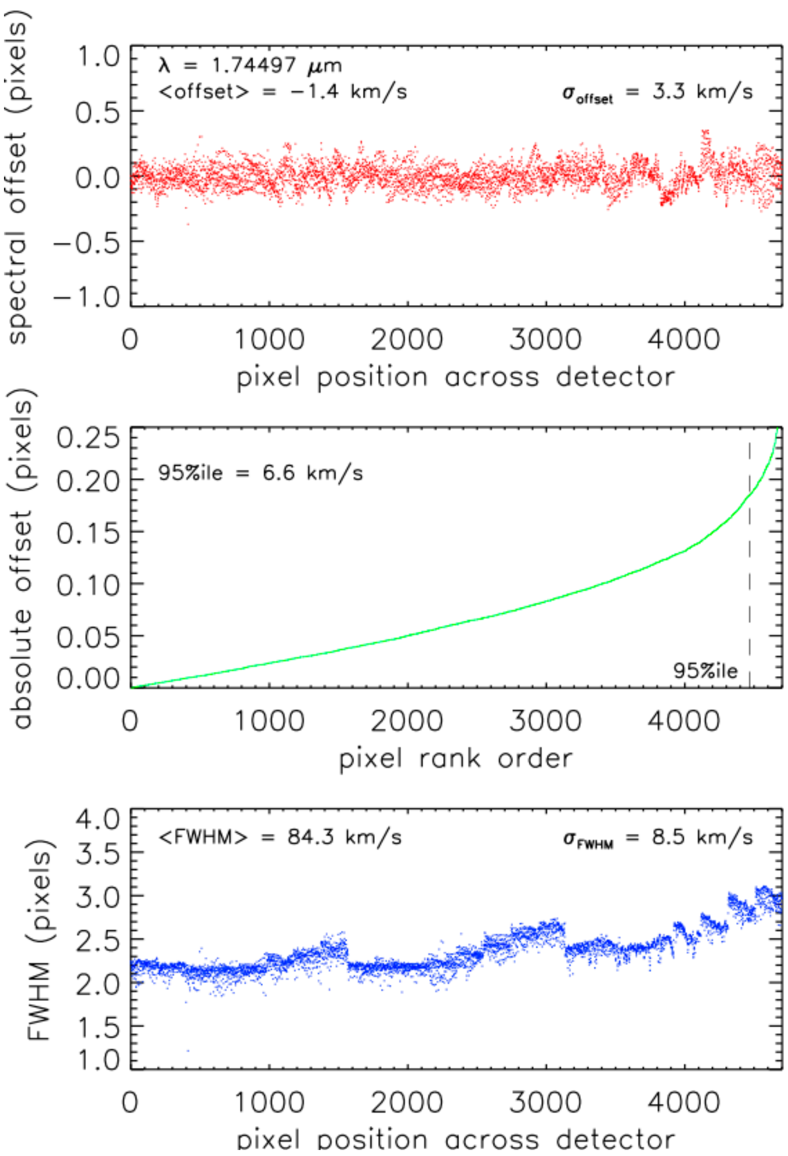
Modified Shepard's method



Calibration accuracy



Reconstructed H-band arc frame (lwNN)



Simple two-step Reconstruction & Combination

- **kmo_reconstruct**
  - Reconstruct every single IFU separately
- **kmo_combine**
  - All cubes combined together

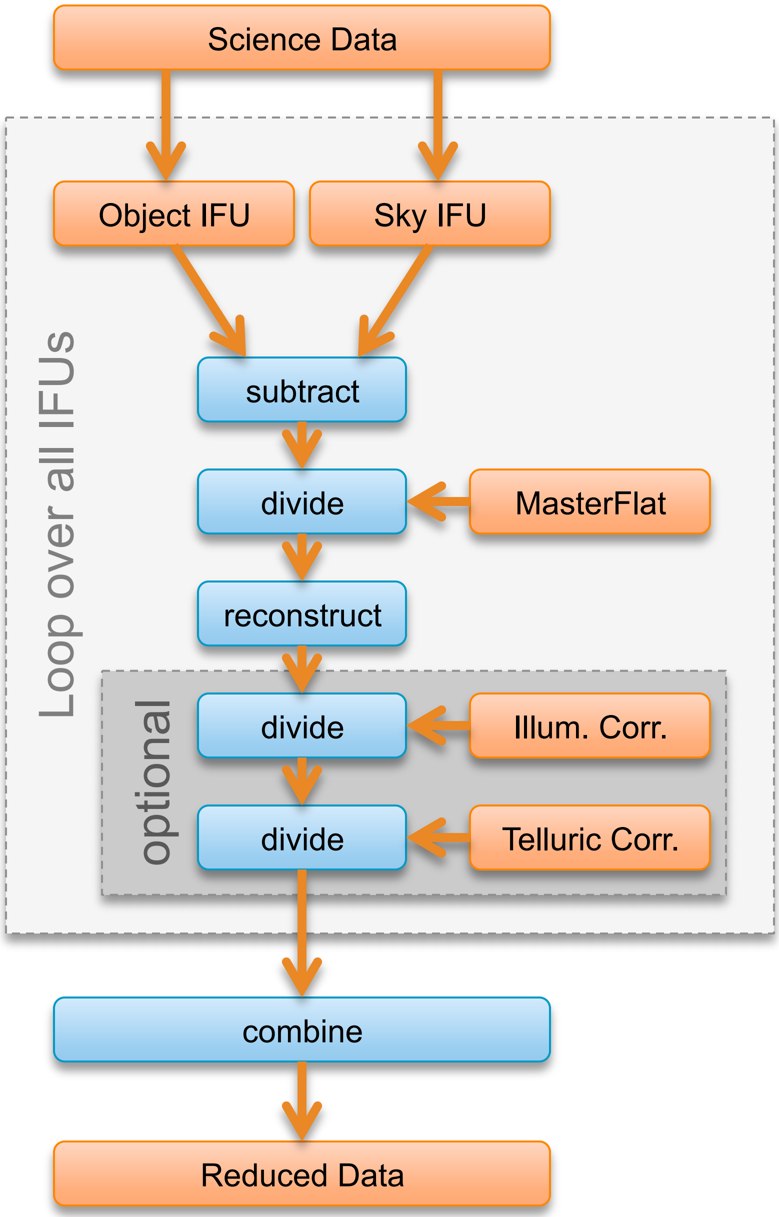
→ Problem: Data is interpolated twice

Advanced Simultaneous Reconstruction & Combination

- **kmo_multi_reconstruct**
  - Reconstruct every single IFU separately
  - Calculate shifts, create Super-LUT
  - Reconstruct all data in one step
  - For every IFU: sky-detection and –subtraction

→ Solution: Data is interpolated once!

Simple Science Reduction



Simple Science Reduction

- **kmo_sci_red**
  - For every IFU: sky-detection and –subtraction
  - Telluric and illumination correction

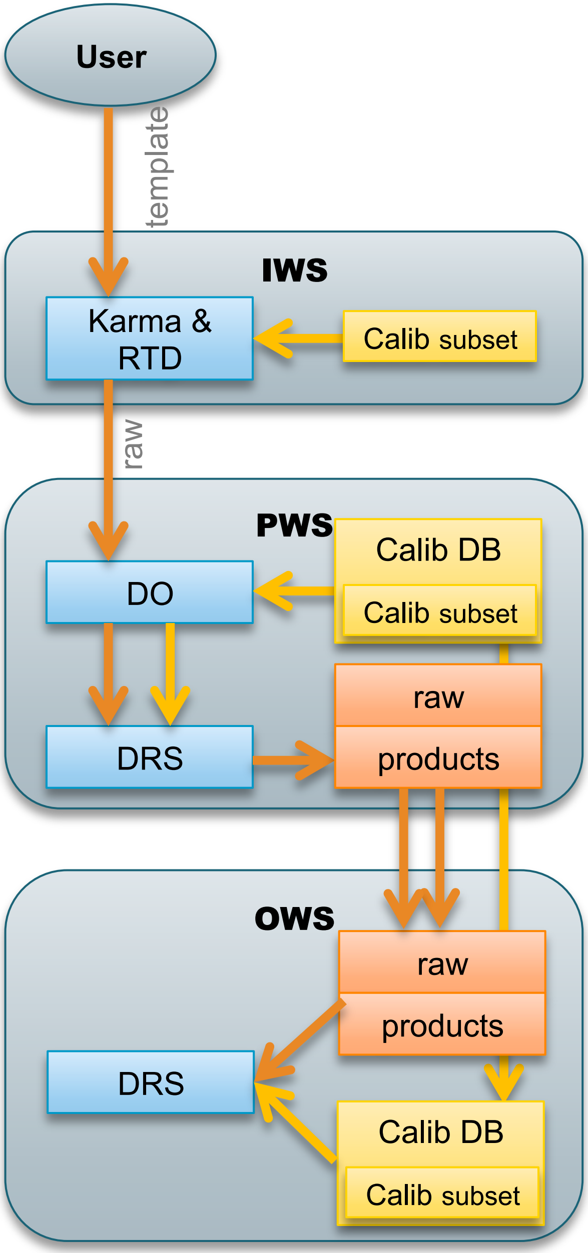
→ Problem: Data is interpolated twice

Advanced science reconstruction (to come)

- Apply "Simultaneous Reconstruction & Combination"
- Include sky tweaking

→ Solution: Data is interpolated once!

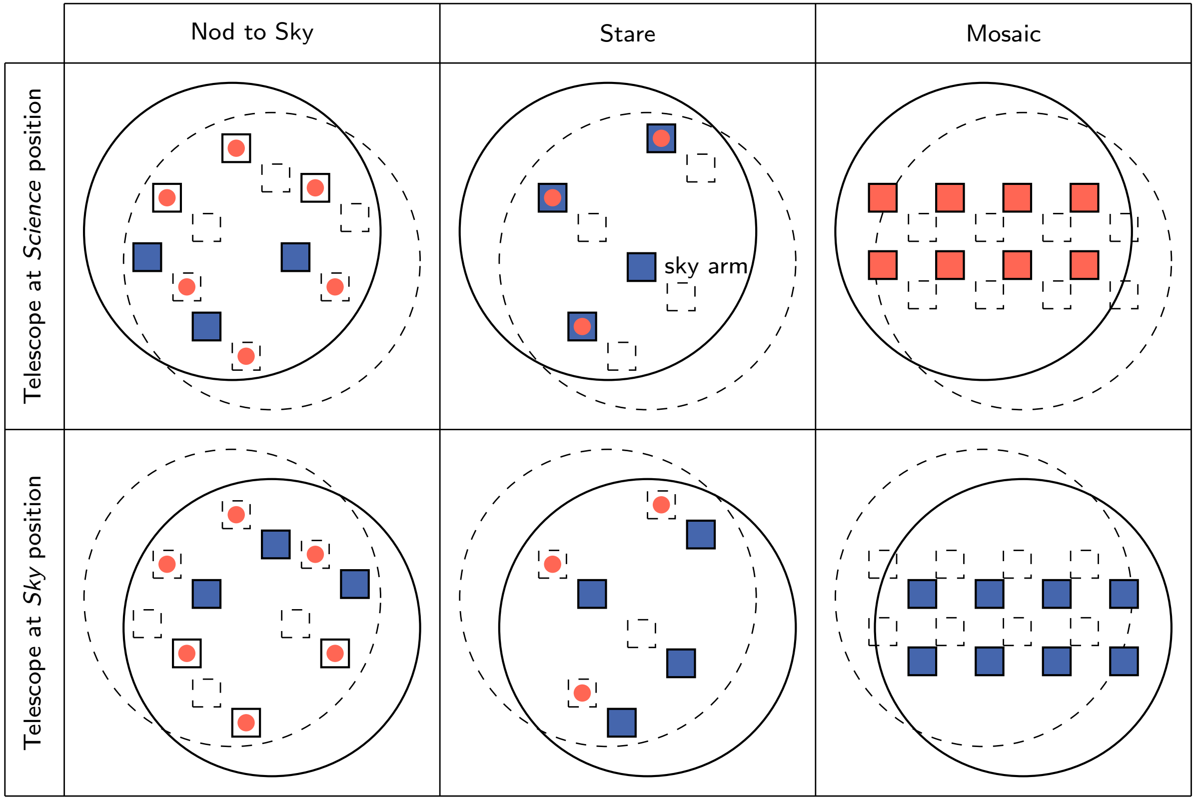
Observation workflow at VLT



IWS:	Instrument Workstation
PWS:	Pipeline Workstation
OWS:	Offline Workstation
Karma:	OB Preparation Tool
RTD:	Real Time Display
DO:	Data Organizer
DRS:	Data Reduction Software
Calib DB:	contains line lists etc AND a static set of calibration files

- User prepares OBs
- OBs are executed on IWS
- Exposures are sent to PWS
  - processed automatically with DO and static Calib DB
  - raw and product frames are archived (renamed): `master_flat.fits` will become `KMOS.2012-06-12T10:19:39.553.fits`
- Raw and product frames are copied to OWS
- On OWS observer can already do his own calculations or data reduction

3 Observation Modes



Nod to Sky

- Sky is obtained by moving telescope/rotating instrument
- Each IFU has its "own" sky
- Arm configuration remains unchanged
- Objects in Science AND Sky-Exposure

Stare

- Less frequent sky exposures (limit: One sky, many object exposures)
- May be possible to use Sky from dedicated IFUs

Mosaic

- Mapping 8 IFUs (4x2)
  - 9 exposures
  - 72 IFUs combined
  - 16x33 arcsec field
- Mapping 24 IFUs (6x4)
  - 16 exposures
  - 384 IFUs combined
  - 43x65 arcsec field
- Plus one sky exposure

The DO implemented in ESO software can't handle KMOS specific object/sky exposures objects and skies intermixed in one frame

Input frames

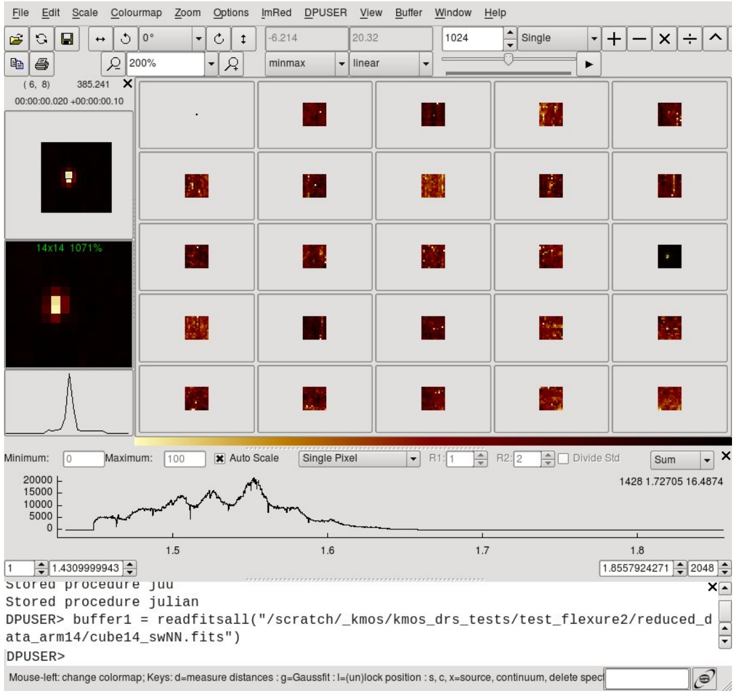
IFU #	1	2	3	4	5	6	7	8	Timestamp in "DATE-OBS" keyword
Frame 1	O	S	S	S	S	S	O	O	2011-01-13T01:00:00.0000
Frame 2	S	O	S	S	S	S	S	O	... 02 ...
Frame 3	S	S	S	O	S	O	S	O	... 10 ...
Frame 4	S	S	S	S	O	S	S	O	... 04 ...
Frame 5	O	S	S	O	S	S	S	O	... 05 ...

Output frames

Frame 1	X	.	.	.	.	.	X	.
Frame 2	.	X	.	.	.	.	.	.
Frame 3	.	.	.	X	.	X	.	.
Frame 4	.	.	.	.	X	.	.	.
Frame 5	X	.	.	X	.	.	.	.
IFU #	1	2	3	4	5	6	7	8
	X	X	.	X	X	X	X	.

Graphical Viewers

- **QFitsView** → MPE, handles extensions
- **CASAviewer** → ESO
- **ds9** → old school
- **fv** → editing tables



Console Applications

MPE (on afs or www.mpe.mpg.de/~ott)

- **dpuser**

Command line interface, same as in QFitsView

ESO (within Scisoft)

- **dfits** Print header information

`$ dfits cube.fits` display primary header

`$ dfits –x 1 cube.fits` display header of ext. 1

- **fitsort** Sort header information

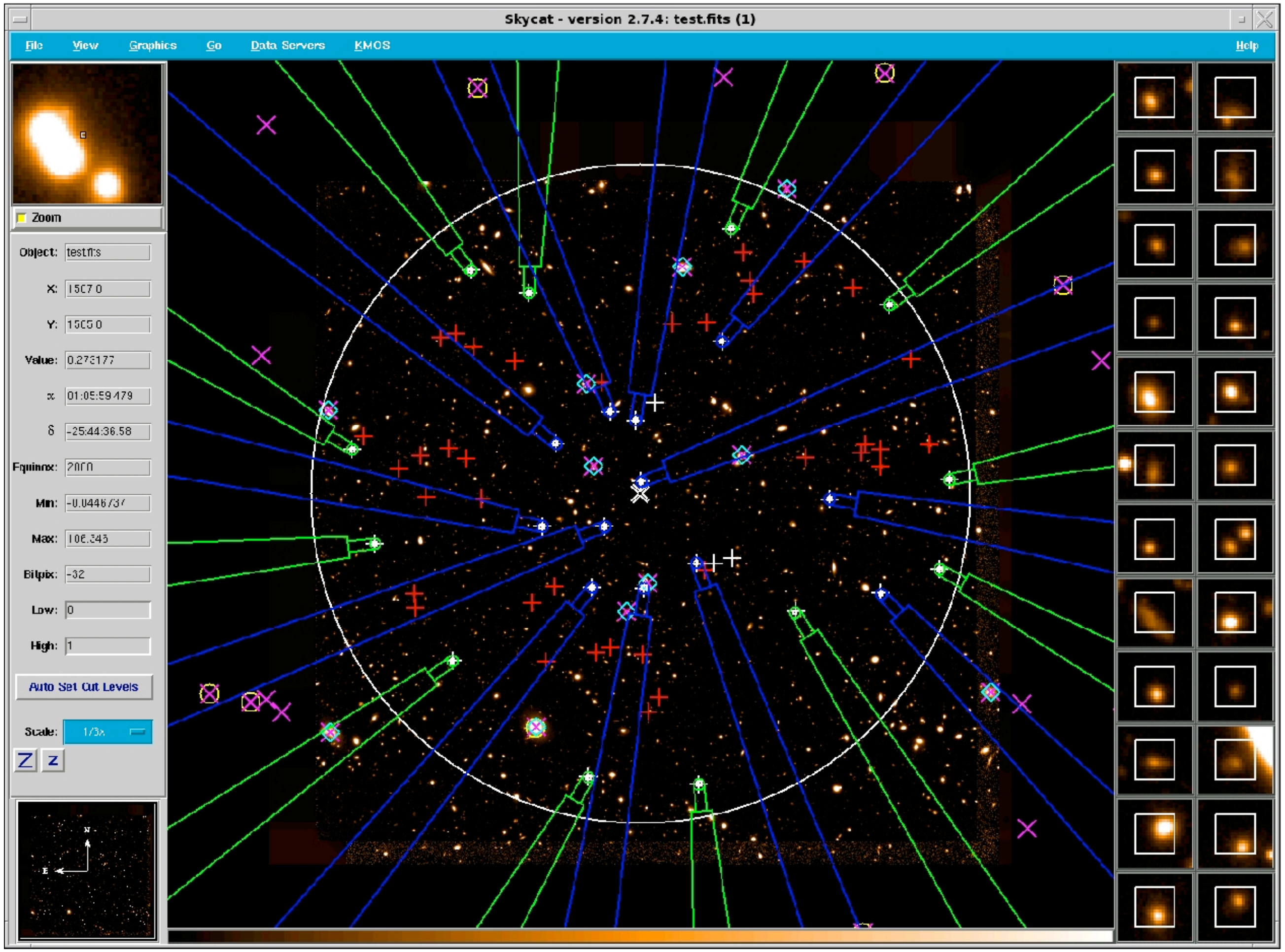
```
$ dfits *.fits | fitsort ocs.rot.naangle
FILE
OCS.ROT.NAANGLE
xcal_HHH_HHH_0.fits          0.
ycal_HHH_HHH_60.fits   60.
```

- **dtfits** Print tables

```
$ dtfits kmo_wave_band.fits
```

Observation preparation

- **KARMA** → USM
  - GUI
  - Avoid arm collision
  - Arm allocation, efficient usage of KMOS
  - Loading & visualizing catalogs



Pipeline package (kmos-kit)

- **The deliverable is a self-contained package**
  - cfitsio
  - wcslib
  - CPL
  - Esorex
  - KMOS
    - Pipeline
    - Manual
    - kmos-calib

Pipeline tools provided by ESO

- **Esorex**
  - Command line interface
- **Gasgano**
  - Data organizer
  - GUI frontend to Esorex
- **Reflex**
  - GUI
  - Workflow oriented
  - Successor of Gasgano

Esorex Basics

Running Esorex

```
$ esorex

     ***** ESO Recipe Execution Tool, version 3.9.6  *****

Libraries used: CPL = 6.1.1, CFITSIO = 3.29, WCSLIB = 4.13.4 (FFTW unavailable)
```

Help on Esorex

```
$ esorex --help

     ***** ESO Recipe Execution Tool, version 3.9.6  *****
Usage: esorex [esorex-options] recipe [recipe-options] sof

Options:
<...>
```

Help on recipes

```
$ esorex --man kmo_dark

     ***** ESO Recipe Execution Tool, version 3.9.6  *****

NAME
  kmo_dark -- Create master dark frame & bad pixel mask (for monitoring detector
              health) and derive mean dark current

SYNOPSIS
  esorex [esorex-options] kmo_dark [kmo_dark-options] sof

DESCRIPTION
<...>
```

Running recipes

```
$ esorex kmo_reconstruct --imethod="swNN" --flux=false reconstruct.sof
```

Where reconstruct.sof (set of files) contains:

```
kmos_data/KMOS_SPEC_TEC040_0008.fits  ARC_ON
xcal_HHH_HHH_0.fits                   XCAL
ycal_HHH_HHH_0.fits                   YCAL
lcal_HHH_HHH_0.fits                   LCAL
kmos-calib/kmos_wave_band.fits        WAVE_BAND
```

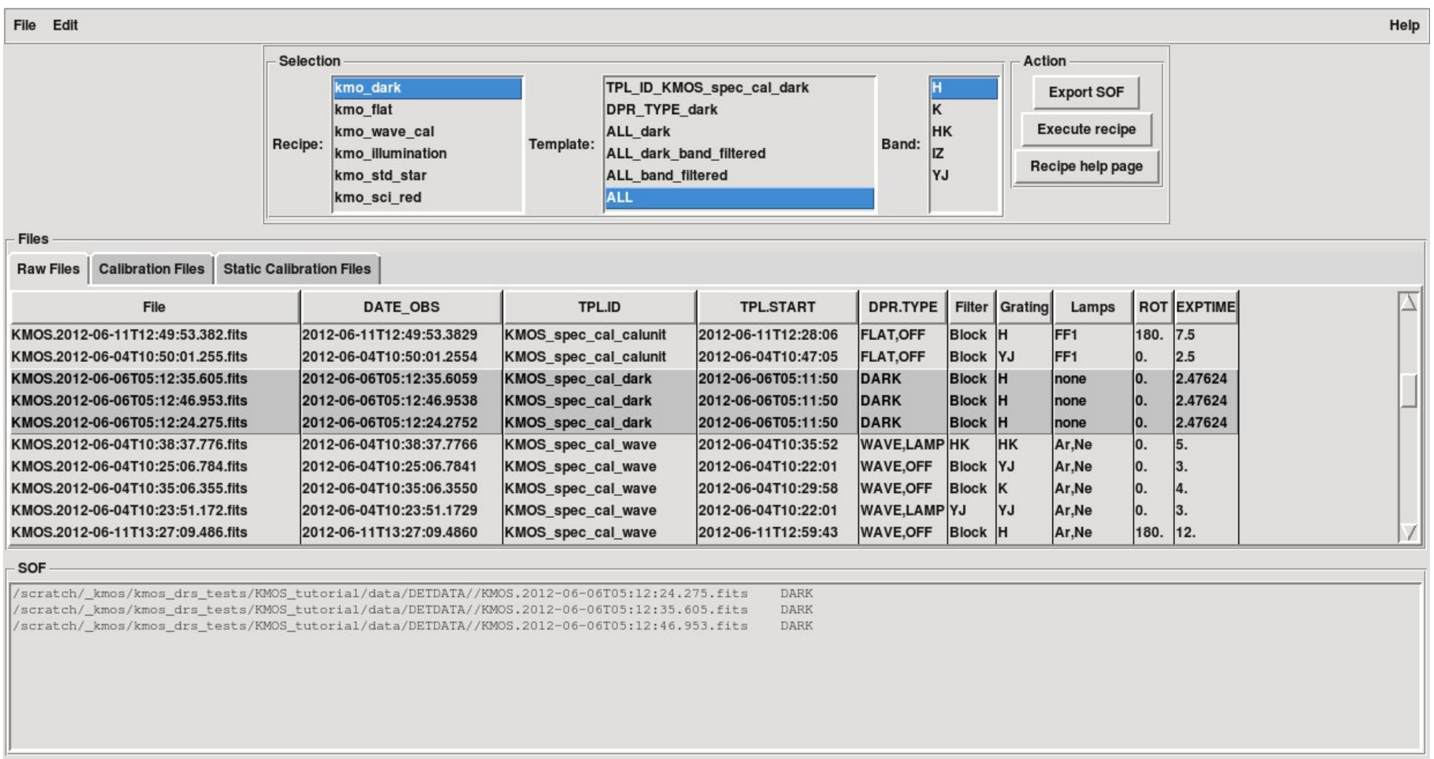
Problem:

Lots of exposures with different bands and rotator angles originating from different templates. How to find quickly the needed exposures?

Solution:

- Either make use of dfits and fitsort
- Or use the pipelineGUI-tool

```
$ pipeline-GUI.pl   --raw=<data_dir> \
                    --cal=<calib_dir> \
                    --static=<static_calib_dir>
```



Recipe: **kmo_dark**

Purpose:

Combine individual dark frames into a MasterDark frame and create a preliminary bad pixel mask.

Main Parameters:

--cmethod="ksigma" (default), "min_max", "median", "average", "sum" *
[others for rejection thresholds, number of iterations, etc.]

Example:

```
esorex  kmo_dark  dark.sof
```

with dark.sof containing:

```
dark_001.fits   DARK
dark_002.fits   DARK
dark_003.fits   DARK
```

It is recommended to supply at least 3 DARK frames

Output frames:

MASTER_DARK, BADPIXEL_DARK

* notes on cmethods: (i) ksigma is based on an initial estimate of sigma from a percentile clipping, and then an iterative estimate of the true sigma. (ii) not all methods are appropriate, but all are available since the combining operation is performed by a more generic recipe.

Recipe: **kmo_flat**

Purpose:

Combine individual flat frames into a MasterFlat frame, create the XCAL and YCAL spatial calibration frames and create the final bad pixel mask.

Main Parameters:

--surrounding_pixels

The amount of bad pixels to surround a specific pixel, to let it be marked bad as well.

Example:

```
esorex kmo_flat flat.sof
```

with flat.sof containing:

```
flat_001.fits        FLAT_ON
flat_002.fits        FLAT_ON
flat_003.fits        FLAT_ON
flat_004.fits        FLAT_OFF
flat_005.fits        FLAT_OFF
flat_006.fits        FLAT_OFF
badpixel_dark.fits   BADPIXEL_DARK
```

It is recommended to supply at least 3 FLAT_ON and FLAT_OFF frames

Output frames:

MASTER_FLAT, BADPIXEL_FLAT, XCAL, YCAL, FLAT_EDGE

Recipe: **kmo_wave_cal**

Purpose:
Create the LCAL wavelength calibration frame.

Main Parameters:
none

Example:

`esorex kmo_wave_cal arc.sof` with arc.sof containing:

```
frame_001.fits                ARC_ON
frame_002.fits                ARC_OFF
master_flat.fits              MASTER_FLAT
badpixel_flat.fits            BADPIXEL_FLAT
xcal.fits                     XCAL
ycal.fits                     YCAL
flat_edge.fits                FLAT_EDGE
kmos_wave_ref_table.fits      REF_LINES
kmos_wave_band.fits           WAVE_BAND
```

Output frames:
LCAL, DET_IMG_WAVE

Recipe: **kmo_reconstruct**

Purpose:
Reconstruct the 24 cubes of a single exposure. Note that no separate bad pixel mask is required because it is already encoded in the calibration frames.

Main Parameters:
--imethod="NN", "lwNN", "swNN", „MS", „CS" *
--flux=TRUE (apply flux conservation)

Example:
`esorex kmo_reconstruct reconstruct.sof` with reconstruct.sof containing:

```
frame_001.fits       OBJECT
xcal.fits            XCAL
ycal.fits            YCAL
lcal.fits            LCAL
kmos_wave_band.fits  WAVE_BAND
```

Instead of OBJECT as well DARK, FLAT_ON, ARC_ON, STD or SCIENCE can be provided

Output frames:
CUBE_DARK, CUBE_FLAT, CUBE_ARC, CUBE_OBJECT, CUBE_STD or CUBE_SCIENCE
DET_IMG_WAVE (optional)

* note on imethod: 3D drizzle will be added as additional interpolation method

Recipe: **kmo_illumination**

Purpose:
Create a calibration frame to correct spatial non-uniformity of flatfield.

Main Parameters:
--imethod="NN", "lwNN", "swNN", „MS", „CS"
--flux=TRUE (apply flux conservation)

Example:

```
esorex kmo_illumination illum.sof
```

with illum.sof containing:

```
frame_001.fits       FLAT_SKY
frame_002.fits       FLAT_SKY
frame_003.fits       FLAT_SKY
master_dark.fits     MASTER_DARK
master_flat.fits     MASTER_FLAT
xcal.fits            XCAL
ycal.fits            YCAL
lcal.fits            LCAL
kmos_wave_band.fits  WAVE_BAND
```

It is recommended to supply at least 3 FLAT_SKY frames

Output frames:
ILLUM_CORR

Recipe: **kmo_std_star**

Purpose:

Create the telluric correction frame

Main Parameters:

--imethod=“NN”, “lwNN”, “swNN”, „MS“, „CS“

--startype=“B7III“

--magnitude=7

Example:

`esorex kmo_std_star std.sof`

with std.sof containing:

```
frame_001.fits            STD
frame_002.fits            STD
master_flat.fits          MASTER_FLAT
xcal.fits                 XCAL
ycal.fits                 YCAL
lcal.fits                 LCAL
kmos_wave_band.fits       WAVE_BAND
illum_corr.fits           ILLUM_CORR
kmos_solar_h_2400.fits    SOLAR_SPEC
kmos_atmos_h.fits         ATMOS_MODEL
kmos_spec_type.fits       SPEC_TYPE_LOOKUP
```

At least 2 STD frames must be supplied.

Output frames:

TELLURIC, STD_IMAGE

Calibration Recipes

- kmo_dark
- kmo_flat
- kmo_wave_cal
- kmo_illumination
- kmo_std_star

Science Recipes

- kmo_sci_red

- kmo_reconstruct
- kmo_multi_reconstruct
- kmo_combine

Basic tools

- kmo_arithmetic
- kmo_copy
- kmo_extract_spec
- kmo_fit_profile
- kmo_make_image
- kmo_noise_map
- kmo_rotate
- kmo_shift
- kmo_sky_mask
- kmo_stats