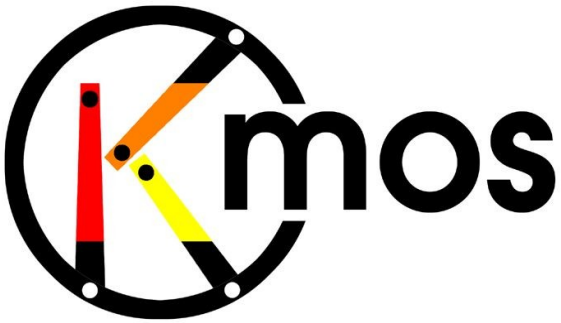

Document Title: **KMOS**
Data Reduction Library Design & User Manual

Document Number: **VLT-MAN-KMO-146611-007**

Issue: **3.6**

Date: **03.06.2020**

Document Prepared By:	A. Agudo Berbel E. Wiezorrek R. Davies N. Förster Schreiber Y. Jung J.A. Escartin Vigo	Signature and Date:	
Document Approved By:	P. Rees	Signature and Date:	
Document Released By:	A. Fairley	Signature and Date:	






Change Record

Issue	Date	Section(s) Affected	Description of Change/Change Request Reference/Remarks
1.1	07.09.07	All	Merging KMOS Data Reduction Library and User Manual
1.2	04.02.08	updated 3.2.1, 5.3, 7.3.4, 9.3.1, 9.3.4, 9.3.5, 9.3.11 inserted & updated 5.1	updated after implementing basic tools 1
1.3	28.10.08	updated 7.3.7, 7.3.9, 7.3.15, 7.3.17, 9.3.7, 9.3.9, 9.3.15, 9.3.17	updated after implementing basic tools 2
1.4.	08.05.09	updated 6.1.1, 6.1.2, 6.1.3, 6.1.4, 7.1.1, 7.1.1, 7.1.2, 7.1.3, 8.2, 9.1.1, 9.1.2, 9.1.3 deleted 8.4 (kmos_split_frame)	updated after implementing calibration recipes 2
1.5.	16.07.09	updated 7.3.8, 9.3.8	updated after implementing basic tools 3
1.6	10.08.10	added sections 4.1, 4.2, 5.5	updated after implementing basic tools 4
1.7	13.04.10	updated 4.3 updated 5.3.3 updated 6.1	templates to change F2L can contain multiple extensions Spec_align recipe obsolete
1.8	10.11.11	All	Issue prepared for TRR
2.2	30.03.12	All	Issue prepared for PAE
2.3		updated 0	
2.4		updated 3.3	
2.5	10.06.13	whole document	Issue prepared for software verification
2.6	02.08.13	updated 7.2.3.8 and 0	kmos_sky_tweak now implemented
2.7.	07.08.13	updated 9.3	
2.8	20.09.13	updated 0	enhanced kmos_multi_reconstruct functionality
2.9	29.10.13	added section 7.1.5	kmos_illumination_flat implemented
2.10	25.11.13	updated sections 0, 0 and 7.2.3 released with pipeline v1.2.7	- added new product frame NOISE_SPEC in kmos_std_star - added new product frame EXP_MASK in kmos_combine and kmos_sci_red
2.11	11.12.13	updated section 7 and 0	- updated association map - kmos_sci_red: velocity offset and arbitrary IFU association
2.12	05.03.14	removed section on kmos_dev_setup updated 7.2.2, 7.3.8.5 and 11.3	removed obsolete parameters, recipe and environment variable, added comment in 7.2.2
2.13	25.07.14	Added Release Notes section at the beginning	Renamed recipes kmos_dark/flat/wave_cal/illumination in kmos_dark/...

2.15	13.02.15	All	Renamed recipes kmos_sci_red in kmos_sci_red. The recipe has 2 additional parameters and produces optionally extra products
2.16	24.03.15	All	Renamed kmos_reconstruct and kmos_std_star New parameters for kmos_std_star. kmos_sci_red products in physical units
2.17	08.02.16	All	Update for version 1.3.17
2.18	15.04.16	All	Update for version 1.3.17
3.0	01.10.16	All	kmos_combine, sky stretching, sky tweaking – For version 1.4.0
3.1	24.11.16	All	kmos_illumination recipe supports FLAT_ON data. kmos_illumination_flat deprecated
3.2	24.04.18	All	Updated association map kmos_gen_telluric kmos_molecfit_model kmos_molecfit_calctrans kmos_molecfit_correct
3.3	20.12.18	All	Example kmos_combine with different rotation
3.4	22.02.19	All	For release 2.3.0
3.5	02.04.19	All	For release 2.4.0
3.6	03.06.20	Updated esoreflex canvas	For release 3.0.0

Table of Contents

Acronyms and Abbreviations

ADU	Analog to Digital Unit – unit used to quantify CCD signal intensity
CLIP	C Library for Image Processing
CPL	Common Pipeline Library
DFO	Data Flow Operations Group (ESO Garching)
DFS	Data Flow System
DIT	Detector Integration Time
DO	Data Organiser
DR	Data Reduction
DRL	Data Reduction Library
DRS	Data Reduction Software
ESO	European Southern Observatory
FITS	Flexible Image Transport System
IFU	Integral Field Unit
IPSRV	Image Processing Server
KMOS	K-band Multi Object Spectrometer
LUT	Look-up Table
MPE	Max-Planck-Institut für extraterrestrische Physik
OB	Observation Block
OS	Observing Software
PSF	Point Spread Function
RTD	Real Time Display
QC	Quality Control
UK ATC	United Kingdom Astronomy Technology Centre
USM	Universitäts-Sternwarte der Ludwig-Maximilians-Universität München
WCS	World Coordinate System

Applicable and Referenced Documents

[AD01] KMOS technical specification, VLT-SPE-ESO-14660-3190, issue 1.0

[AD02] KMOS Data Reduction Library Specification, VLT-SPE-KMO-146611-001, issue 1.1

[RD01] KMOS Instrument Software Design Description, VLT-SPE-KMO-146606-003, issue 1.0

[RD02] Bentley J., Friedman J., 1979, "Data Structures for range searching", ACM Computing Surveys, 11, 397-409

[RD03] Clark I., Harper W., 2000, "Practical Geostatistics 2000", pub. Geostokos

[RD04] Yang C.-S. et al., 2004, "12 Different Interpolation Methods", in *Geo-Imagery Bridging Continents*, XXth ISPRS Congress

[RD05] Lekien F., Marsden J., 2005, "Tricubic interpolation in 3 dimensions", Int. J. Numer. Meth. Engang, 63, 455-471

[RD06] Renka R., 1988, "Multivariate interpolation of large sets of scattered data", ACM Trans. Math. Software, 14, 139-148

[RD07] Shepard D., 1968, "A 2-dimensional interpolation function for irregularly spaced data", Proc. 23rd Nat. Conf. ACM, 517-523

[RD08] Farage C., Pimbblet K., 2005, PASA, 22, 249

[RD09] van Dokkum P., 2001, PASP, 113, 1420

[RD10] Davies R., 2007, MNRAS, 375, 1099

[RD11] Cappellari M., Copin Y., 2003, MNRAS, 342, 345

Scope of this Document

This document defines the design of the data reduction library for the KMOS pipeline, including all modules of the DRL to process KMOS data as well as the additional DFS tools. It provides a technical description of the instrument modes, data formats and data processing required for scientific observations, calibrations, and instrument monitoring tasks for KMOS. It is based on the DRL Specification [AD02] and supersedes that document.

Release Notes

This **version 3.00** of the KMOS pipeline implements:

- Reflex canvas & tutorial updates
- Compatibility with Telluriccorr 3.0.X, including parameter renaming
- An improved method for the calculation of ABMAGLIM
- Workflow Update
- Add extracted IDP for kmos_spec_extract when input is IDP
- Fixed kmos_gen_telluric fails in computing the QCZPOINT in some cases
- Save outputs with a new cpl_frameset that have only the input cpl_frame in the cpl_frameset
- The product generated by kmos_gen_telluric should have the MJD-OBS of the input TELLURIC.fits file if provided
- Added another parameters in the kmos_molecfit_save function in order to take into account all the input frameset to add the output products in kmos_molecfit_correct
- Fix Headers of various products of various recipes
- fixed kmos_reconstruct: wrong MJD-OBS in output files
- fixed wrong input to kmos_combine in the kmos workflow

- Fixed Select_reference.py not compatible with numpy 1.7
- Fix PIXNOISE computation
- Reflex interactive actors made more robust agains missing input data
- Backward compatible python 3 support
- suppress_extension parameter in lower case
- Produce combined frame if 1 input

PART I: DRS DESIGN

1 Instrument Description

1.1 Brief Description

KMOS is a multi-object near infrared spectrograph with a spectral resolution of R~3000, depending on bandpass observed. It comprises 24 arms which can be positioned so as to cover almost any combination of objects within a 7.2arcmin patrol field. Each arm is an integral field spectrometer with a field of view of 2.8arcsec × 2.8arcsec and a sampling of 0.2arcsec per pixel. So that the light can be dispersed in the conventional way, each field is sliced by a suite of mirrors into 14 slitlets, each 14 pixels long. These are then rearranged by a second suite of mirrors into a single pseudo-longslit. The primary aim of the data processing software is to reconstruct the 3D data cubes from the 2D data on the detectors.

KMOS is designed so that 8 IFU arms are fed into a single spectrograph and have their light dispersed onto a single detector. Thus, in total there are 3 spectrographs and 3 detectors. Each section is identical with all the others. Hence, the format of the data on each detector is, modulo optical alignment and manufacturing tolerances, identical.

KMOS generates its own internal flatfields. For this it uses 2 lamps mounted in an integrating sphere outside the instrument. The light is directed through a sealed tube to another integration sphere in the centre of the cryostat, and thence to each arm. In order to detect light from the flatfield lamps, the arms must be positioned correctly outside the patrol field. It is possible that for some configurations, parts of some arms may be vignetted. In addition, there may be unexpected spatial non-uniformities in the flatfield. As a result it will be possible to make an illumination correction by observing a blank sky field during twilight. This will provide a correction to the spatial (rather than spectral) component of the flatfield.

KMOS has also internal lamps (Argon and Neon) which will be used for wavelength calibration. As an example, these are estimated to produce 35 lines in the K-band with more than 100 counts in a 150-second integration.

1.2 Modes and Configurations

Although KMOS itself is a complex instrument, the only observing mode available is multiple integral field spectroscopy.

The only instrument configuration that the observer can make (and which has an impact on the subsequent data reduction, with respect to the appropriate calibration data) involves the wavebands – for each of which there is a single fixed spectral format and range, and a fixed filter. The wavebands offered cover near-infrared wavelengths from 0.8μm to 2.5μm, and hence the observing strategy is the same for all bandpasses.

1.2.1 Instrument Flexure

KMOS is mounted at a Nasmyth focus of the VLT and hence rotates. It is therefore inevitable that there will be at least some flexure. For individual exposures, the most noticeable impact (i.e. elongated PSF) will be when the telescope is pointing close to zenith and the parallactic angle is changing rather quickly. However, calculations suggest that spatial flexure will be very small (less than 1 pixel). On a scheme how to handle this can be found in Sec. 4.5.1.

On the other hand, spectral flexure is expected to be significant: exceeding the Technical Specification on wavelength accuracy. Although mechanical solutions have been investigated, it has been decided that it is more reliable, more accurate, and simpler to correct this in software rather than hardware. Since science exposures will typically have integration times of at least a few minutes, the OH sky lines will be bright and clear in individual frames. The processing will reconstruct an initial cube from each science frame using the wavelength solution derived from the arc lamp. It will then measure the wavelength offset of the frame by comparing the observed wavelengths of the OH lines with respect to their theoretical wavelengths. This offset will be folded back into the wavelength solution and the cube reconstructed anew from the raw data (and the initial reconstruction will be deleted). Thus correcting the spectral flexure will not compromise the quality of the data by requiring additional interpolation steps.

1.2.2 Inputs

The DRS pipeline receives as input:

- Raw images from KMOS as a single file with 3 extensions
- Calibration data, of which there are two types:
    - master calibrations, generated by the pipeline, typically from daytime calibrations
    - ancillary data such as reference line catalogues

1.2.3 Outputs

The KMOS DRS pipeline creates the following data:

- 3D cubes, which are calibrated in wavelength, spatial position, and flux.
- associated error cubes (as FITS extension)
- QC1 parameters and performance monitoring values.

It should be noted that spectra will be extracted for standard star observations, in order to generate the necessary telluric corrections. But in general spectra will not be extracted from science observations, although it would in principle be possible to do this using exactly the same technique and recipe as for standard stars. The reason is that often it is not obvious from which spatial pixels the spectrum should be taken. This is particularly true for observations of high redshift galaxies (one of the primary science drivers of KMOS), where continuum emission is either very weak or even undetected. Attempting to extract spectra automatically from fields where either the object of interest is very faint or there are multiple objects, can lead to misleading and confusing results. On the other hand, extracting a spectrum manually is very quick and easy to do within QFitsView. As one moves the cursor across the displayed image of the spatial field of view, it enables one to see *in real time* integrated spectra from different groups of spaxels. This tool is already available at Paranal, and users are recommended to use it to do exactly this. QFitsView also enables the user to create a collapsed image from the cube (or even a linemap) across any wavelength range in an equally straightforward and speedy manner.

While bad pixel masks are generated during the processing, these are not part of the output. The main reason is that due to the necessary interpolation step, there does not exist a one-to-one correspondence between pixels in the final cube and pixels on the detector. However, the impact of bad pixels is reflected in the noise cube which is created along with the data cube (see Section 2.2). Bad pixels are simply ignored during the interpolation. This will result in a local increase in the noise, which will be apparent in the noise cube.

1.2.4 Data Formats

Only standard FITS data formats with extensions are used for tables, 2D and 3D images.
ASCII files are used for parameter files (e.g. EsoRex or Gasgano configuration files).

1.2.5 Pipeline Modes

The DRS pipeline will be able to run in 3 specific default modes which are built from the same set of recipes but with different input parameters, and 1 more general mode. These are:

Acquisition pipeline mode: this will run on Paranal in real time to aid in acquiring targets. In order to achieve the maximum speed, a number of stages will be omitted and the reconstructed data will be approximate (although sufficient for the task in hand); the final output will be a set of images.

On-line pipeline mode: this will run on Paranal in quasi real time in an automated manner with a primary goal of monitoring the scientific results from execution of an OB, and generating initial QC parameters.

Off-line pipeline mode: this will be run by the DFO in Garching in order to generate all necessary calibration products, which will be certified by the DFO and archived. It will also be used to generate reduced frames from service mode observations, which are then sent to the proposer.

Desktop Processing: the pipeline can be run by an observer at their home institution using the EsoRex and Gasgano tools. External software such as QFitsView can be used to view intermediate and final data products; the observer can freely select all parameters; and if required add in their own processing steps.

2 Mathematical Description

2.1 Interpolation

In KMOS, reconstruction of a (rectilinear) 3D datacube from raw 2D data will be performed in a single step. This is no more risky or difficult than interpolating in 2-dimensions. However, being able to conceptualise it requires that the calibrations are viewed in a particular way. Traditionally, calibrations are considered to be the mathematical functions (polynomials) which allow one to correct the curvature in the recorded data. Instead, calibrations should be considered as a look-up table associating each data value in the raw frame with its (x,y,λ) position in the reconstructed cube. This is shown graphically in Figure 1, where the calibration look-up tables would allow one to go from (a) to (b).

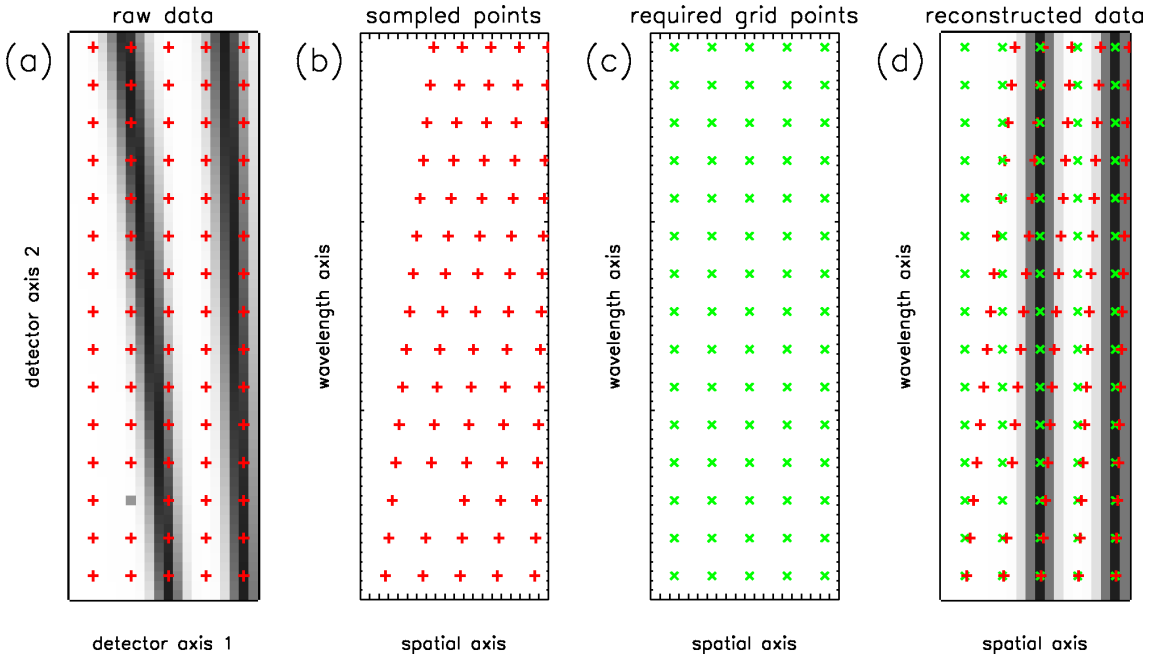

Figure 1: Illustrative example of the perspective required in order to interpolate in 3D. (a) Observed data are sampled regularly in the reference frame of the detector. (b) This sampling is irregular in the reference frame of the reconstructed cube; bad pixels can simply be omitted from the set of sampled points. (c) One can freely specify the required gridding (i.e. spatial/spectral pixel scale) for the reconstructed data; it is independent of the actual sampling. (d) Each required grid point is interpolated from the sampled points which lie in its neighbourhood. Any suitable algorithm (see below) can be used for the interpolation.

The recorded data on the detector can then be considered as a set of values at irregularly spaced sampling positions in the final cube. Once this is done, one can dissociate the data completely from the detector frame and simply generate a list of values and positions:

$value_0, x_0, y_0, \lambda_0$
$value_1, x_1, y_1, \lambda_1$
…
$value_n, x_n, y_n, \lambda_n$

Each grid position in the reconstructed cube is interpolated from its nearby neighbours, which are selected from this list of data values. Bad pixels are simply excluded from the list. Doing this brings a number of advantages:

- The 3D datacube can be reconstructed in a single step, improving the noise properties of the final dataset
- One can combine frames during the interpolation by concatenating as many lists as required from various raw frames; this simply increases the number of sample points close to each interpolated grid point.
- One can choose the sampling of the reconstructed cube arbitrarily. This is useful if one wishes to compare the data to that from another instrument: the KMOS data can be directly reconstructed at a matching pixel scale.
- The data can be smoothed during the reconstruction (for some algorithms), simply by increasing the size of the local neighbourhood from which sampling points are taken.

It is fortunate that there are many different schemes available for interpolating points in 3-dimensional space, since no single one is optimal for every situation. Each has its advantages and disadvantages. It is for this reason that we will make several schemes available. In this section the methods we propose to include within the KMOS data reduction software are described. While these are all standard methods, few have actually been applied extensively to astronomical data. It is not practical to provide a full description of each here, and so only the salient points are described. The reader is referred to various references for further details.

2.1.1 Nearest Neighbour

This is the simplest, and also one of the fastest, methods imaginable for interpolation: one simply adopts the value of the nearest data point. This method is included since no additional noise is added during the interpolation process, and as a result there may be instances when an observer wishes to use this method: e.g. when signal-to-noise is more critical than optimal spatial/spectral accuracy. The efficiency of this method can be enhanced using the cell method developed by Bentley & Friedman (1979) [RD02]. A script called `ngp.pro` which performs this interpolation is available from the IDL Astronomy User's Library.

This method is available in the KMOS pipeline as value *"NN"* in the corresponding parameter settings.

2.1.2 Cubic Spline Interpolation

Cubic spline interpolation is a standard technique which is discussed in detail in, amongst others, Numerical Recipes. As far as we are aware, it is applied commonly throughout astrophysical data. The goal of a cubic spline is to get a formula that is smooth in the first derivative and continuous in the second derivative, not only within an interval but also at its boundaries. We will use the natural cubic spline, which has zero second derivative at its boundaries.

The issue here is how to apply it in 3 dimensions. A method has been developed by Lekien & Marsden (2005) [RD05] which does this; but it requires that the data are gridded regularly. While the KMOS data are gridded regularly on the detector, their position (x,y,λ) is not uniform and therefore it would be quite difficult to apply this method – indeed to do so one would need to calculate accurately where on the detector any particular point in (x,y,λ) would fall.

The alternative most commonly employed is to perform multiple 1-dimensional interpolations. This makes the cubic spline method relatively straight forward mathematically. One useful characteristic of the data in this respect is the fact that the pixel spacing perpendicular to the slitlets in each IFU is regular – which, due to optical distortions, is not the case either along each slitlet or along the spectral axis. One can then perform the first set of interpolations along this axis and then propagate the regular spacing to the other dimensions.

This method is available in the KMOS pipeline as value *"BCS"* in the corresponding parameter settings.

2.1.3 Modified Shepard's Method

This fits a smooth function to a set of data points scattered in 3 dimensions using a modification by Renka (1988) [RD06] of a method developed by Shepard (1968) [RD07]. The necessary algorithms are part of the NAG library (their `nag_3d_shep_interp` and `nag_3d_shep_eval` routines). It is also available in IDL as the `grid3.pro` routine.

The original basic method constructs a function $Q(x,y,z)$ which interpolates a set of m scattered data points at positions (x_i,y_i,z_i) and having values f_i with a weighted mean:

$$Q(x,y,z) = \frac{\sum_{i=1}^{m} w_i(x,y,z) f_i}{\sum_{i=1}^{m} w_i(x,y,z)}$$

where the weights are simply

$$w_i(x,y,z) = \frac{1}{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}$$

The modification is that the method is made local by truncating the weights w_i beyond a specified distance R_w.

This method is available in the KMOS pipeline as value *"swNN"* in the corresponding parameter settings, where the truncation radius can be specified (recommended box size is 1.1 pixels). An analogous linear distance weighted scheme is also available under the name *"lwNN"*.

We note that in the full Modified Shepard's method, the performance is improve by replacing each f_r by $q_r(x,y,z)$ which is a quadratic fitted by weighted least-squares to local data (i.e. within a radius R_q). The resulting surface is continuous and has continuous first partial derivatives. It is the calculation of each $q_r(x,y,z)$ that takes most of the processing time, but nevertheless the method is remarkably fast, as shown by Yan et al. (2004). The radii R_w and R_q are chosen to be large enough to include N_w and N_q data points respectively, and it is these latter numbers that define how localised the interpolant is. For smaller numbers, the interpolation only uses local data and so is faster but possibly less accurate; for larger numbers the computational cost is higher. The method is not thought to be particularly sensitive to the choice of these parameters and typical values of $N_w = 32$ and $N_q = 17$ seem to work well, based on experimental results reported by Renka (1988).

2.2 Error Propagation

One of the goals of the pipeline is to produce (at least a reasonable approximation to) an error cube to complement the final reduced and combined data cube. This is an important consideration since the noise is strongly wavelength dependent – being affected most by the presence of OH lines and the thermal background. In addition, in a combined cube, the noise will be spatially dependent.

In principle creating a noise cube ought to be straight forward since the basic mathematics of error propagation are straight forward and well known. In practice, this is not so, most notably due to systematic effects when combining different datasets. Any useful estimate of the error should include these, and as a result our methods assess the noise from the data themselves rather than simply propagating a formal estimate.

2.2.1 Initial Noise Estimate

It is assumed that the gain (e-/ADU) and the readnoise (e-) are either known or can be measured. In this case the noise in any raw 2D frame can be found (or strictly, only estimated, because the counts measured are themselves subject to noise) simply as

$$\sigma(ADU) = \frac{\sqrt{counts \times gain + readnoise^2}}{gain}$$

This relation can be tested as follows: for a large number (e.g. 20) identical exposures, the standard deviation between the values at each position on the detector should be equal to σ as estimated above. Alternatively, since the readnoise is approximated by the noise in a frame with exposure time of MINDIT, this same method can be used to derive the gain.

2.2.2 Mathematical Manipulations

The recipe kmo_arithmetic allows one to perform mathematical manipulations on the data. For these cases, the errors can be propagated in a strictly mathematical way. This applies similarly to the recipes kmo_rotate and kmo_shift. We have ignored covariance terms since they are expected to be small for uncorrelated data.

For example, if one adds (or subtracts) two frames then (ignoring cross terms) the noise adds in quadrature.

if x = au+bv then $\sigma_x = \sqrt{a^2\sigma_u{}^2 + b^2\sigma_v{}^2}$

And if one multiples (or divides) two frames, then (again ignoring cross terms) the noise combines as:

if x = auv then $\frac{\sigma_x}{x} = \sqrt{\frac{\sigma_u{}^2}{u^2} + \frac{\sigma_v{}^2}{v^2}}$

Similarly, raising a number to some power

if $x=au^b$ then $\frac{\sigma_x}{x} = b\frac{\sigma_u}{u}$

And lastly, for exponentials and logarithms one has

if $x=ae^{bu}$ then $\frac{\sigma_x}{x} = b\sigma_u$

and

if x=a ln(bu) then $\sigma_x = a\frac{\sigma_u}{u}$

2.2.3 Combining Datasets

We described two methods for estimating the noise in the result when multiple cubes are combined. Both of these options will be available; the latter will be the default.

If one is combining cubes which have either small spatial dithers between them (i.e. multiple exposures of the same field) or large dithers (i.e. in order to mosaic a larger field) one can in principle use the formal relations above to combine the individual error estimates. Thus

$$\sigma_{combine} = \frac{1}{n}\sqrt{(\sigma_1^{\ 2} + \sigma_2^{\ 2} + ... + \sigma_n^{\ 2})}$$

where there are pixels overlapping. For all image regions where there is no overlap one simply propagates the noise estimate directly.

While this can always be applied, it has a disadvantage in that it does not take into account systematic effects between the different data sets being combined (e.g. offsets in the background level). Thus an alternative method which will be offered is to estimate the noise directly from the standard deviation of the pixel values at each spatial/spectral position. This has the advantage that one can iteratively reject values which lie outside a threshold defined in terms of the standard deviation of the (remaining) pixels – thus yielding a better mean value in the combined cube.
The only restriction is that such a noise estimate can only be made if there are at least 3 values available at any given spatial/spectral position; in practice positions where this criterion is not met will simply be assigned a noise of NaN.

2.2.4 Extracting Spectra

The process of extracting a spectrum from a datacube is simply adding up spectra within a given aperture (possibly weighted appropriately). The noise can therefore be propagated from the cube to the spectrum very simply, by using the relation for a weighted sum given in Section 2.2.2.

2.2.5 Creating Images

Images are created simply by collapsing the cube along its spectral axis within specified wavelength ranges (and perhaps also excluding some intermediate wavelength ranges). As for spectra, the noise can therefore easily be propagated using the relation for a weighted sum in Section 2.2.2.

3 Instrument Data Description

The aim of this section is to describe the structure of the raw data produced by KMOS, which corresponds to the RAW format.

KMOS comprises 24 IFUs, each of which has 14×14 spatial pixels and approximately 2000 spectral pixels. The data from these are recorded by three 2k×2k HAWAII 2RG detectors, with 8 IFUs assigned to each detector. The field of each IFU is sliced into 14 slitlets which are rearranged along a pseudo-longslit and then dispersed. The raw data for each IFU therefore consists of 14 sets of standard 2-dimensional (1 spatial, 1 spectral) slit spectra, which is arranged next to each other on the detector, separated by a few blank pixels. The same pattern is repeated 8 times for each of the 3 detectors. A single exposure therefore produces approximately 50Mb data. Figure 3 illustrates how the data appears on each detector. See also Figure 19 for an illustration of how the raw data appears in the RTD.

A single integration with KMOS will produce three 2-dimensional frames, each 2048×2048 pixels, stacked in 3 extensions of a single fits file with an empty primary header.

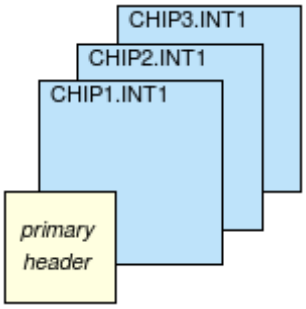

Figure 2 Format of a RAW file as the instrumentation software delivers it. The value for the EXTNAME keyword can be seen in the blue rectangles.

Calibration observations are performed in a standard way and typically yield data with a similar format: darks, flats, wavelength calibration and spectral curvature. The exceptions are the illumination correction and standard stars.

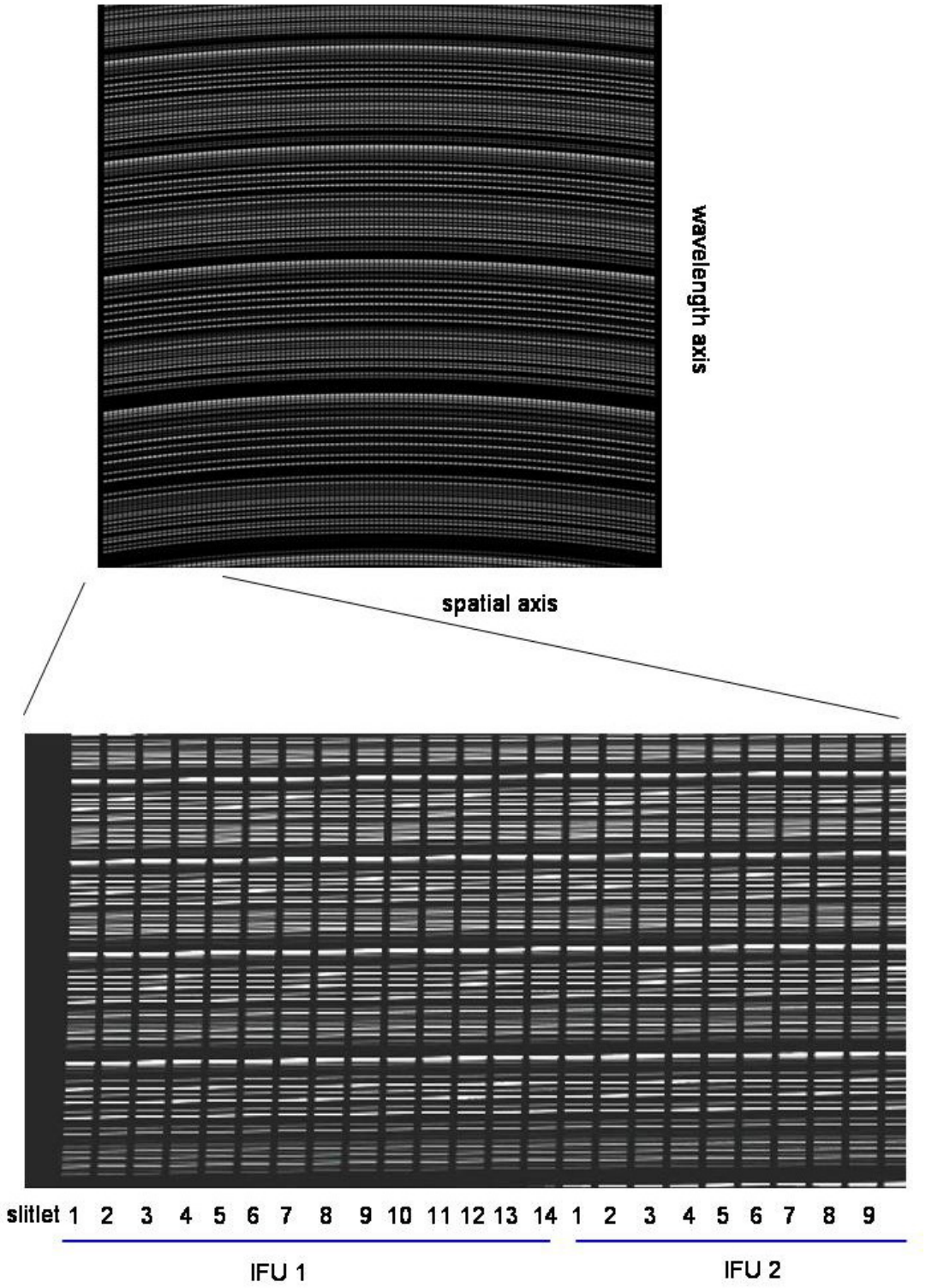

Figure 3 - illustrative layout of the data format on each detector (curvature has been enhanced for visual purposes). Upper panel: full detector showing OH emission lines on the H-band; Lower panel: left side, stretched to show individual slitlets within each IFU are arranged.

3.1 Orientation of the IFUs on the detectors

Due to the optical path realised in the KMOS instrument the spatial orientation of the IFUs on the detector frames isn't the same for all of them, as one would expect intuitively. The orientation of a reconstructed slitlet of an IFU can be flipped or rotated. The orientation of the wavelength axis

never changes. The wavelength is always lowest at the bottom and highest at the top of the detector frame as depicted in Figure 3.

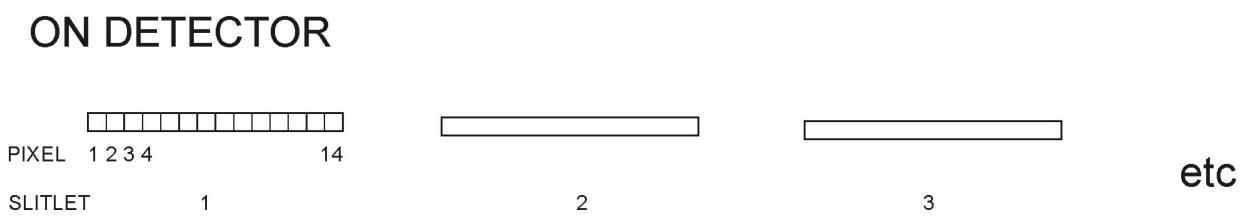

Figure 4 Numbering of pixels and slitlets as they are referenced to in **Figure 5**

For IFUs 17, 18, 19, and 20 the pixels in a slitlet are orientated from left to right and the slitlets are stacked from top to bottom.
For IFUs 21, 22, 23 and 24 the pixels in the slitlet are oriented just the other way round, from right to left. As well the stack orientation is flipped, it goes from bottom to top.
Whereas in IFUs 1, 2, 3, 4, 13, 14, 15 and 16 the slitlets are oriented vertically from bottom to top. The stacks are stacked from left to right.
Finally in IFUs 5, 6, 7, 8, 9, 10, 11 and 12 the slitlets are also vertical but go from top to bottom and they are stacked from right to left.

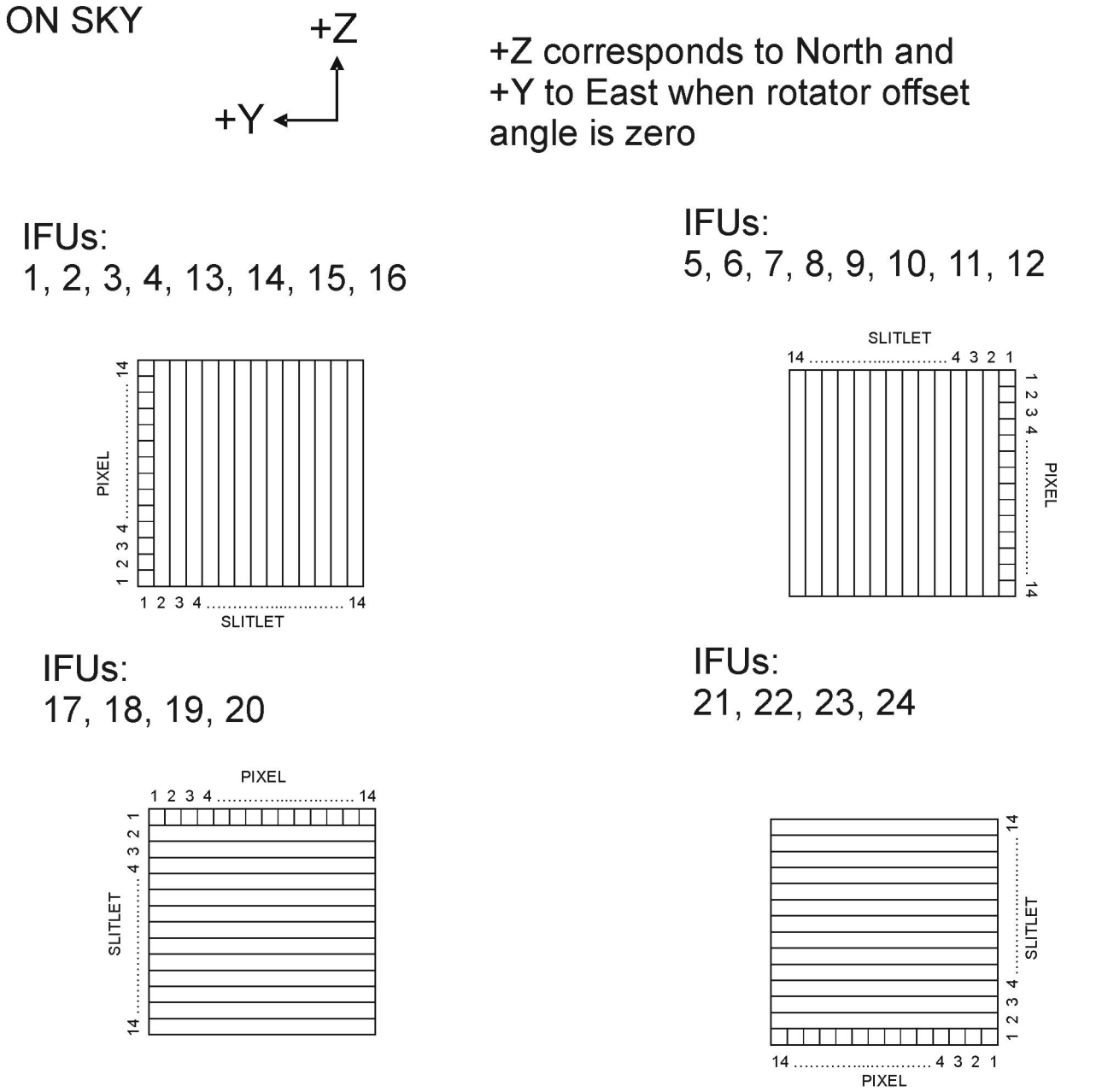

Figure 5 Orientation of the slitlets for the different IFUs

3.2 FITS header keywords

The tables below define the FITS header keywords which are required by the data reduction pipeline. The Instrumentation Software will provide these keywords in the headers of raw frames – see KMOS Instrument Software Design Description [RD01].

3.2.1 Primary header

Keyword	value	comment
DATE	string	Date the file was written
DATE-OBS	string	Observing date
EXTEND	bool	There may be FITS extensions
NAXIS	int	number of array dimensions
NAXIS1	int	# of pixels in axis1

NAXIS2	int	# of pixels in axis2
HIERARCH ESO DET NDIT	int	Number of detector integrations
HIERARCH ESO DET SEQ1 MINDIT	double	Minimum DIT
HIERARCH ESO OBS ID	int	Observation block ID
HIERARCH ESO INS FILTi ID (i=1-3)	string	Filter unique id
HIERARCH ESO INS GRATi ID (i=1-3)	string	Grating unique ID
HIERARCH ESO INS LAMPi ST (i=1-4)	bool	arc lamp status (on/off), i=1,2 flatfield lamp status (on/off) , i=3,4
HIERARCH ESO OCS ARMi ALPHA (i=1-24)	double	RA centre of arm i (J2000)
HIERARCH ESO OCS ARMi DELTA (i=1-24)	double	Dec centre of arm i (J2000)
HIERARCH ESO OCS ARMi NAME (i=1-24)	string	Target name hosted by arm i
HIERARCH ESO OCS ARMi NOTUSED (i=1-24)	string	String containing error message. If keyword isn't present, then the arm is functional
HIERARCH ESO OCS ARMi TYPE (i=1-24)		
HIERARCH ESO OCS ROT OFFANGLE	double	Rotator offset angle
HIERARCH ESO OCS TARG DITHA	double	Telescope dither in ALPHA [arcsec]
HIERARCH ESO OCS TARG DITHD	double	Telescope dither in DELTA [arcsec]

3.2.2 Extension header

Keyword	value	comment
EXPTIME	double	Integration time
EXTNAME	string	string describing the extension
NAXIS	int	number of data axes
NAXIS1	int	length of data axis 1
NAXIS2	int	length of data axis 2
XTENSION	string	IMAGE extension
HIERARCH ESO DET CHIP GAIN	double	Gain in e-/ADU
HIERARCH ESO DET CHIP INDEX	int	Chip index
HIERARCH ESO DET CHIP RON	double	Read-out noise in e-

The reduction pipeline updates the headers in a way that information applying to all frames is stored in the empty primary header. Detector or IFU specific information is stored in the subsequent headers (see also section 4).

3.3 Raw file types

The raw files are generated using different templates that represent the available modes to use KMOS with. When a template is executed the following keywords are written into all generated files:

- HIERARCH ESO DPR TYPE (unique identifiers to perform DO categorisation)
- HIERARCH ESO DPR CATG (qualitative category of the file)
- HIERARCH ESO DPR TECH (technical category of the file)

- HIERARCH ESO OCS TEMPL ID (the applied template)

With these keywords it is possible for the Data Organiser (DO) to classify the files and provide the corresponding DO category that is needed to run the KMOS pipeline properly.

The raw files with DPR.TECH equal IMAGE or SPECTRUM require no reconstruction of the data cubes. In these cases the data will be treated as the simple 2D frame that it is.

DO category	DPR TYPE	DPR CATG	DPR TECH	OCS TEMPL ID
DARK	DARK	CALIB	IMAGE	KMOS_spec_cal_dark
FLAT_ON FLAT_OFF	FLAT,LAMP FLAT,OFF	CALIB CALIB	SPECTRUM IMAGE	KMOS_spec_cal_calunitflat
ARC_ON ARC_OFF	WAVE,LAMP WAVE,OFF	CALIB CALIB	SPECTRUM IMAGE	KMOS_spec_cal_wave
FLAT_SKY	FLAT,SKY	CALIB	IFU	KMOS_spec_cal_skyflat
STD	OBJECT,SKY,STD,FLUX	CALIB	IFU	KMOS_spec_cal_stdstar KMOS_spec_cal_stdstarscipatt
SCIENCE	OBJECT,SKY	SCIENCE	IFU	KMOS_spec_obs_nodtosky KMOS_spec_obs_stare KMOS_spec_obs_mapping8 KMOS_spec_obs_mapping24 KMOS_spec_obs_freedither

The following DO categories are not used in the pipeline itself.

Although acquisition frames will need to be processed in order to reconstruct the acquisition images needed for the real time display, the recipe will be triggered by CLIP rather than any header keywords (because the frames do not have headers at this stage).
For acquisition frames one exposure will have objects in (some) arms and the subsequent exposure will be of blank sky fields. However, for most science observations, this will not be the case: in any single exposure some arms will be on sky and some arms will be on objects.

DO category	DPR TYPE	DPR CATG	DPR TECH	OCS TEMPL ID
ACQ_OBJ	OBJECT	ACQUISITION	IFU	KMOS_spec_acq KMOS_spec_acq_lutatcfstars
ACQ_SKY	SKY	ACQUISITION	IFU	KMOS_spec_acq KMOS_spec_acq_lutatcfstars
ACQ_STD	OBJECT,SKY	ACQUISITION	IFU	KMOS_spec_acq_stdstar KMOS_spec_acq_stdstarscipatt

The technical templates do not require specific data processing other than reconstructing the cubes. All measurements of the source size and position will be done afterwards manually.

DO category	DPR TYPE	DPR CATG	DPR TECH	OCS TEMPL ID
FOCUS	LAMP,FOCUS	TECHNICAL	SPECTRUM	KMOS_spec_tec_focus
LOOKUP	OBJECT, LOOKUP	TECHNICAL	IMAGE	KMOS_spec_tec_lutatcfstars

3.3.1 Dark

File types: DARK
These frames are observed with the filter wheel in a ‘blocked’ position. Dark frames are used as the OFF frames for the illumination correction.

3.3.2 Flatfields

File types: FLAT_ON, FLAT_OFF, FLAT_SKY

The standard flatfield is illuminated by a pair of lamps via an integrating sphere. Each set of flatfields FLAT_ON has an associated set of FLAT_OFF frames, taken immediately before (although in principle a standard dark frame could suffice). In case there are spatial non-uniformities in the flatfield, and also to take into account vignetting further upstream in the light path, an illumination correction can be performed. Since this is taken on sky, a dark frame is used as the corresponding OFF frame. The edges of the illuminated regions of the flatfields can also be used to trace the spectral curvature.
In order to measure the spectral curvature and calibrate KMOS while it is mounted, the edges of the illuminated regions in the flatfields will be traced. This provides 2 traces per slitlet. As a result one has to assume that the magnification as a function of wavelength is uniform across the slitlet.

3.3.3 Wavelength

File types: `ARC_ON, ARC_OFF`
These frames are illuminated simultaneously by Ar and Ne arc lamps. Each `ARC_ON` frame has an associated `ARC_OFF` frame, taken immediately before (although in principle a standard dark frame could suffice).

3.3.4 Standard Star

File types: `STD` (object and sky)
This type identifies observations of a telluric standard star. In addition, for the many such stars where the magnitude is well known, these also provide the photometric calibration. Because the standard stars are observed in an IFU, there are no issues associated with limited slit width, seeing corrections, etc.

3.3.5 Science Object

File types: `SCIENCE` (object and sky)
These frames are illuminated by a science target. It should be noted that in most cases, for any particular exposure only some of the 24 IFUs will be on objects and the rest will be on sky. The necessary keywords `OCS.ARMi.TYPE` indicating whether each individual IFU is on sky or on object in any particular frame are written into the header by the OS.

3.4 Processing Table

The different recipes for generating calibration and science products are listed in the Data Processing Tables in Appendix A. These relate the various calibration recipes to their respective raw data types. The tables connect the classification keywords, the DO category, and the observing template. Required input from the calibration database is indicated, as are the final products. A summary of the main processing steps is given, as are the FITS header keywords needed by the recipe.

3.5 IFU Layout in the Mapping Templates

The mapping modes of KMOS have specific templates to perform the observations. But the data are treated by the pipeline in exactly the same way as for any other science observation.

It is often useful to know which IFUs in which exposures makes up the various parts of the patchwork mosaic. Figure **6** and Figure **7** show this information for the 8-arm and 24-arm mapping modes respectively.

21	2	3	8
20	15	14	9

A B C
D E F
G H I

Figure 8 Left – Arrangement of the IFUs used in the template KMOS_spec_obs_mapping8 used for the Mapping8 mosaic mode. Right – order (from A to I) of the 9 dithers performed during the Mapping8 mode. The IFUs are separated by 8.1" and each dither is 2.7" so that, at the end, there is a 0.1" (half-pixel) overlap between adjacent pieces.

23	24	1	3	5	6
21	22	2	4	8	7
19	20	16	14	10	9
18	17	15	13	12	11

A B C D
E F G H
I J K L
M N O P

Figure 9 Left – Arrangement of the IFUs used in the template KMOS_spec_obs_mapping24. Right – order (from A to P) of the 16 dithers performed during the Mapping24 mode. The IFUs are separated by 10.8" and each dither is 2.7" so that, at the end, there is a 0.1" (half-pixel) overlap between adjacent pieces.

4 Data Reduction Library Data Structures

During the different processing steps, the raw data is modified and associated with additional information, which is either produced during the reduction or originates externally. The resulting data types are described in this section.

Note that in both of these tables, the file type is given as a 3-character identification:

- the first character refers to whether the data in the file is stored as a floating point number ('F', number of bits unspecified) or a binary digit ('B');
- the second character indicates the dimension of the data (1, 2, or 3);
- the third character indicates whether the data refers to a complete detector array ('D'), an individual IFU ('I'), a look-up table or list ('L'), or a spectrum of arbitrary size ('S').

4.1 Classification Tags

The classification of intermediate and final data products that will be generated by the calibration recipes and pipeline is given below, together with the recipe which generates them and a brief description of the product:

PCATG	*type*	*recipe*	*description*
MASTER_DARK	F2D	kmos_dark	- dark frame (including noise map)
MASTER_FLAT	F2D	kmos_flat	- flatfield frame (including noise map)
XCAL	F2D		- spatial solution lookup frame
YCAL	F2D		- spatial solution lookup frame
LCAL	F2D	kmos_wave_cal	- wavelength solution lookup frame
LEVEL_CORRECTED	F2D	kmos_level_correct	- level corrected frame
ILLUM_CORR	F2I	kmos_illumination	- illumination correction to flatfield
TELLURIC	F1I	kmos_std_star	- normalised telluric spectrum (including noise map)
STAR_SPEC	F1I		- extracted star spectrum
STD_IMAGE	F2I		- images from a standard star cube collapsed along the spectral axis
STD_MASK	F2I		- mask used for extracting the spectra
EXTRACT_SPEC	F1I	kmos_extract_spec	- extracted spectrum
COMBINED_CUBE	F3I	kmos_sci_red	- reconstructed and combined science cubes (including noise map)
SCI_RECONSTRUCTED	F3I		- intermediate reconstructed science cubes (including noise map)
ATMOS_PARM	F1L	kmos_molecfit_model	- atmospheric parameters
BEST_FIT_PARM	F1L		- best fitting parameters
BEST_FIT_MODEL	F1L		- best fit model and intermediate products
TELLURIC_CORR	F1I	kmos_molecfit_calctrans	- telluric spectrum
TELLURIC_DATA	F1L		- telluric spectrum and intermediate products
SINGLE_SPECTRA	F1I	kmos_molecfit_correct	- 1D data corrected for tellurc absortion
SINGLE_CUBES	F3I		- 3D data corrected for tellurc absortion

The classification of ancillary external data files is given below:

PCATG	*file type*	*description*
ARC_LIST	F1L	list of arc line wavelengths & strengths
OH_LIST	F1S	spectrum of OH line wavelengths & strengths
ATMOS_MODEL	F1S	high resolution model spectrum of atmospheric transmission
SOLAR_SPEC	F1S	high resolution solar spectrum
SPEC_TYPE_LOOKUP	F2L	lookup table to find stellar effective temperature from spectral type and luminosity class
KERNEL_LIBRARY	F2I	Library of instrumental spectral resolution as function of wavelength, IFU, rotator angle and Grating.
RESPONSE	F1S	Library of instrument response as function of wavelength, IFU, Rotator angle and Grating.

The various formats are detailed in the following subsections.

4.2 Intermediate Data Formats

All files have an empty primary header, data and noise maps are stored in extensions as described below.

4.2.1 Detector based floating point products

File Type: F2D
PCTAG: MASTER_DARK, MASTER_FLAT
For these files, the detector pixel space (i.e. 2048×2048 pixels) is still the reference frame in which the data are stored. The data of each detector is stored in an extension of the FITS file. for the dark and flat frames), these will be stored in extensions of the same FITS file. In this case the first extensions will contain the data of the first detector, the second extension will contain the associated noise map and so on.

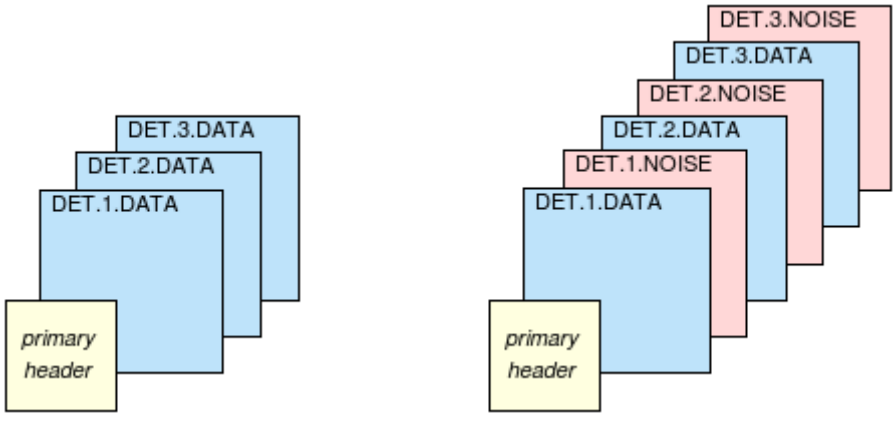

Figure 10 The two valid configurations of a F2D-frame either with or without noise maps. The value for the EXTNAME keyword can be seen in the blue and red rectangles.

4.2.2 1-dimensional detector based products

File Type: F1D
PCTAG: –
These files can be created by some intermediate recipes, e.g. kmo_stats. When statistics are to be calculated from a detector based frame, then the output frame follows the same naming

convention. F1D frames can either have one or three extensions. With noise it will be two or six extensions.

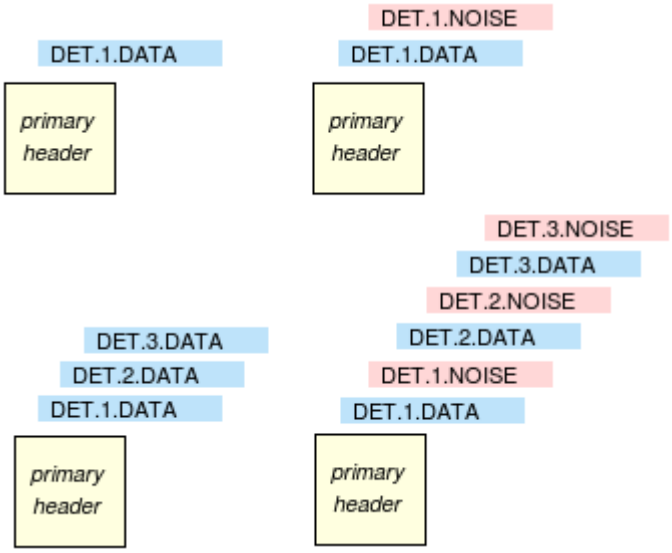

Figure 11 The valid configurations of F1D-frames either with or without noise maps.

4.2.3 Detector based binary digit products

File Type: B2D
PCTAG: `BADPIXEL_DARK, BADPIXEL_FLAT`
These files also have the detector as the reference frame in which the data are stored, in fact they are almost identical to F2D frames. But the data stored has another meaning: i.e. '0' stands for a bad pixel, '1' for a good pixel. The FITS files will have extensions corresponding to the 3 detectors (like in Figure 10 on the left side). A B2D frame can't contain any noise frames.
Note that although a list of bad pixels would require less file space, it requires additional processing and does not allow for an easy way to visually check the bad pixel map.
To distinguish F2D from B2D frames the EXTNAME keyword contains DET.1.BADPIX, DET.2.BADPIX and DET.3.BADPIX.

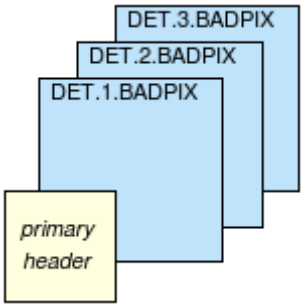

Figure 12 The valid configuration of a B2D-frame either with or without noise maps.

4.2.4 1-dimensional IFU based products

File Type: F1I
PCTAG: `TELLURIC, STAR_SPEC, SINGLE_SPECTRA`
The IFU spectral domain is the reference for the storage of these data – i.e. the data is a simple spectrum, the length and sampling of which correspond exactly to those of the spectral axis of a reconstructed cube. The same telluric correction will be used for all IFUs, and so the only extension in the FITS file will correspond to the noise spectrum. A F1I-frame can either contain the spectrum of just one IFU or of up to 24 IFUs. For inactive IFUs (for which hence no data exists) an empty extension is inserted for data as well for the noise map.

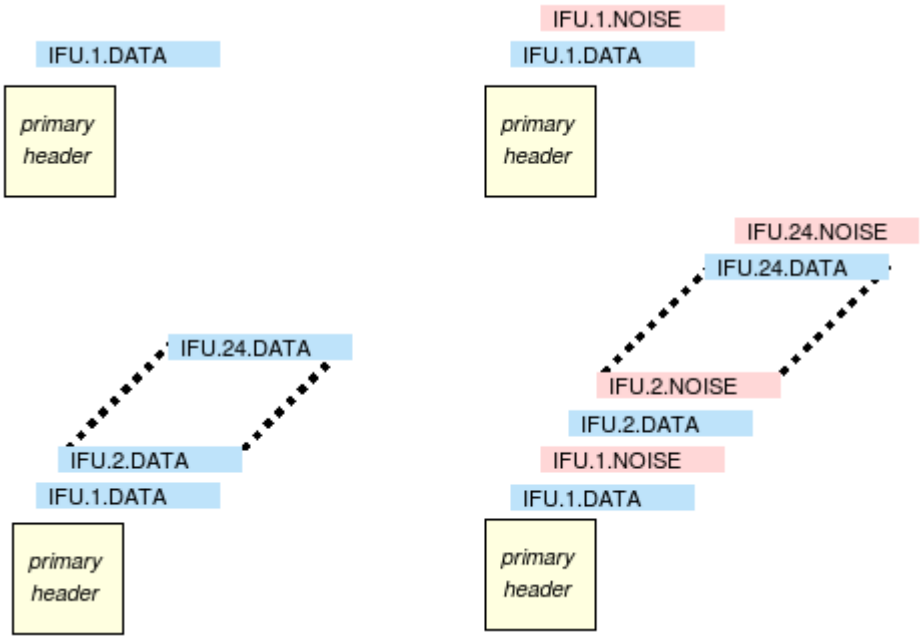

Figure 13 All valid configurations of a F1I-frame either with or without noise maps. The value for the EXTNAME keyword can be seen in the blue and red rectangles.

4.2.5 2-dimensional IFU based products

File Type: F2I
PCTAG: `ILLUM_CORR, STD_IMAGE`
The IFU spatial field is the reference for the storage of these data (i.e. 14×14 pixels) – i.e. the data correspond to a cube which is collapsed along the spectral axis. Since KMOS has 24 IFUs, the data will be stored in up to 24 extensions or in 48 extensions with noise maps in a single FITS file. All extensions will be presenting every file produced; those for which no data exist will be left empty. A F2I-frame can either contain images of just one IFU or of up to 24 IFUs.

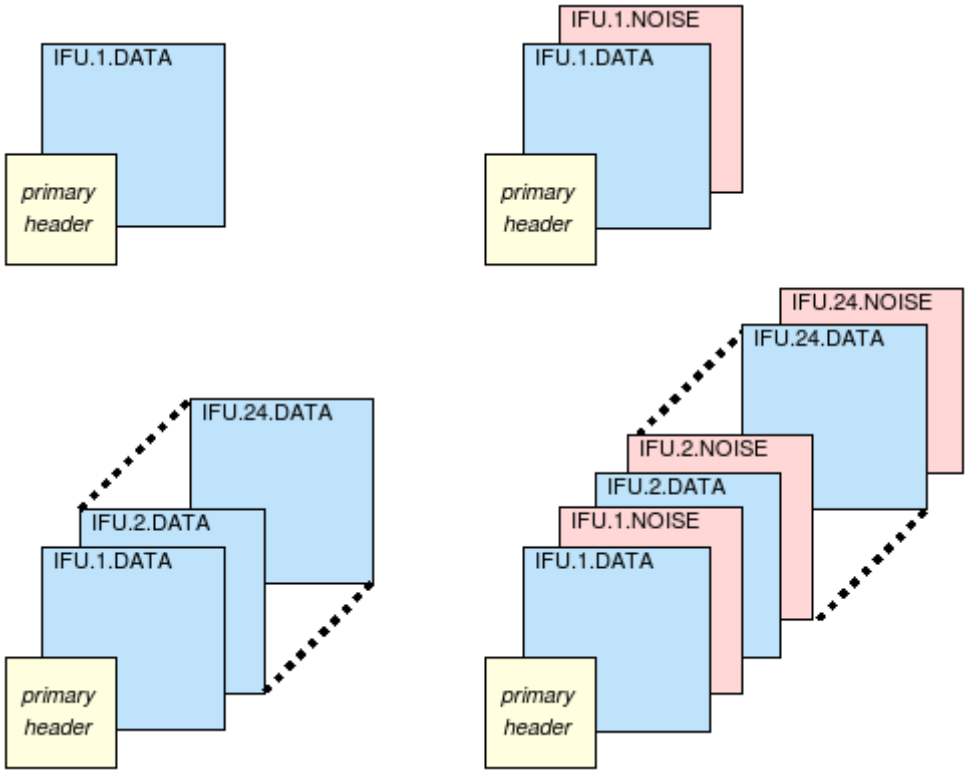

Figure 14 All valid configurations of a F2I-frame either with or without noise maps.

4.2.6 Naming convention

For all intermediate data formats described in section 4.2 (and also for F3I in section 4.4.1) the convention is followed that in all extensions the EXTNAME keyword is describing its origin and content. The format is **"TYPE.NR.CONTENT"**,
where TYPE can be DET or IFU,
where NR can be a number between 1 to 24 and
where CONTENT can be DATA, NOISE or BADPIX.

This convention is modified when cubes are combined using the recipe kmos_combine. Since the cubes to be combined needn't to stem from the same IFU (for example an object is observed in the first OB on IFU #2 and in the second OB on IFU #13), the format will be changed to **"TYPE.CONTENT"**.The user will have to keep track himself of the history of the IFUs if he desires so. kmos_combine will take the header of the first fits file in the sof-file and modify it accordingly.

4.3 External Data Formats

All files have an empty primary header, data and noise is stored in extensions as described below.

4.3.1 Lists

File Type: F1L
PCTAG: ARC_LIST, ATMOS_PARM, BEST_FIT_PARM, BEST_FIT_MODEL, TELLURIC_DATA
These file types will be stored as a binary fits table. The EXTNAME keyword contains the string "LIST".

The line list will have three columns: the first column will contain a list of wavelengths corresponding to the positions of the lines; the second column will contain a corresponding list of approximate line strengths. The third column contains a string, either "Ar" or "Ne" depending to which gas the line belongs to. With this information, it will be possible both to generate a spectrum at the appropriate resolution to match that of the bandpass and also to unambiguously identify particular lines in an observed spectrum. Note that because two different arc lamps are used, there is uncertainty in the relative strengths of the lines between these two lamps. Therefore the arc line strengths will not be used by the automatic pipeline. However, the information will be retained in the data file for the astronomer and possible future upgrades or other unforeseen uses.

4.3.2 1-dimensional spectra

File Type: F1S
PCATG: ATMOS_MODEL, SOLAR_SPEC, OH_LIST, RESPONSE
These data formats will be stored as linearly sampled spectra, with the standard parameters defining the wavelength sampling given in the header. These spectra are at very high resolution and cover the entire wavelength range of all the bandpasses used within KMOS. When needed, the appropriate section of the spectrum can be convolved to the required resolution. The structure of a F1S file follows the definition of a F1I file, except that there can only be one data extension without noise and the EXTNAME keyword contains the string "SPEC".

4.3.3 Lookup tables

File Type: F2L
PCATG: SPEC_TYPE_LOOKUP, FLAT_EDGE, REF_LINES
A lookup table is by definition 2-dimensional. Therefore this data format will consist of a binary fits table with an appropriate number of rows and columns. The EXTNAME keyword contains the string "LIST".
In the case of SPEC_TYPE_LOOKUP, the aim is to cover the most common MK spectral types so that the effective temperature of any telluric star (typically a B or G2V star) can be estimated:
luminosity classes: I, II, III, IV, V
spectral type: O5, O9, B0, B2, B5, B8, A0, A2, A5, F0, F2, F5, F8, G0, G2, G5, G8
This file type can either have one or 24 extensions.

4.4 Final Output Data Formats

4.4.1 3-dimensional IFU based products

File Type: F3I
PCATG: CUBE_DARK, CUBE_FLAT, CUBE_ARC, CUBE_OBJECT, CUBE_STD, REDUCED_CUBE, SINGLE_CUBES
The processed datacubes (i.e. 14×14×2048 pixels), one corresponding to each of the 24 IFUs is stored in a F3I fits file. As the other formats described above, F3I has as well an empty primary header and data and noise maps are stored alternately. Extensions for inactive IFUs are left empty.

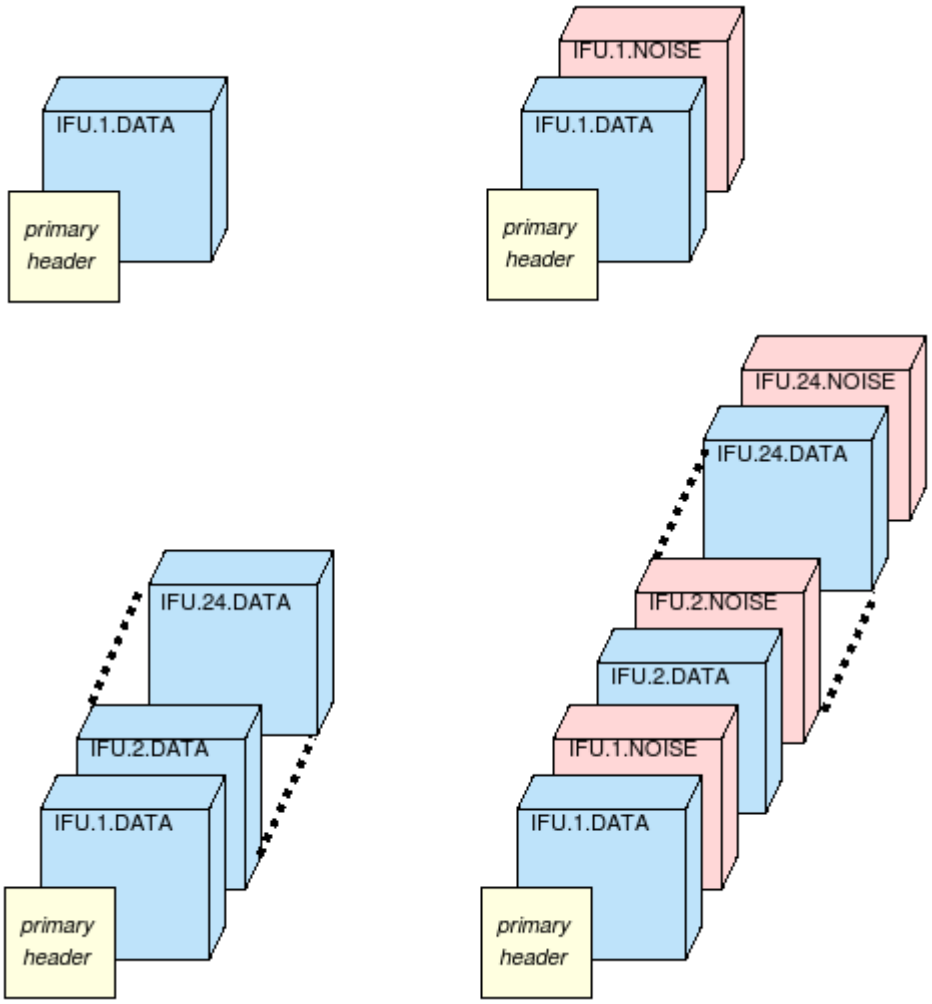

Figure 15 All valid configurations of a F3I-frame either with or without noise maps.

4.5 Calibration Data Formats

Since the orientation of all IFUs isn't the same due to the optical path of the KMOS instrument the spatial solution lookup frames XCAL and YCAL (see section 4.1) are intermixed. The assembly of the RAW frames in respect to the IFUs is explained in section 3.1 in detail.

Following figures show the setup of the three calibration frames XCAL, YCAL and LCAL:

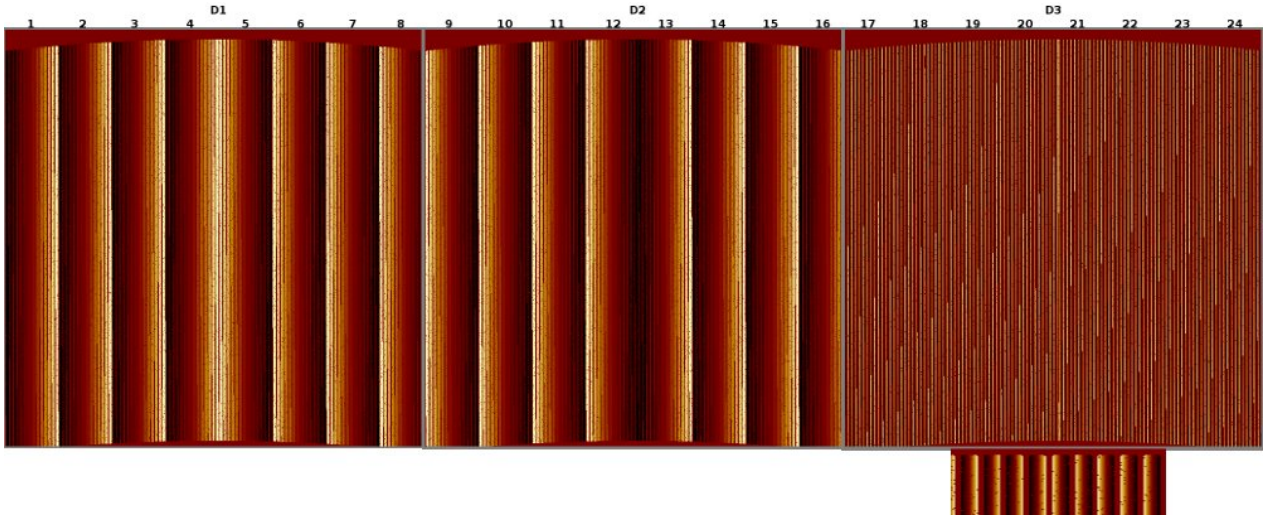

Figure 16 XCAL: In the first two detector frames there is the same data value inside each slitlet. So the visible gradient extends over the whole IFU. In the third detector frame the extends over each slitlet individually (see magnification)

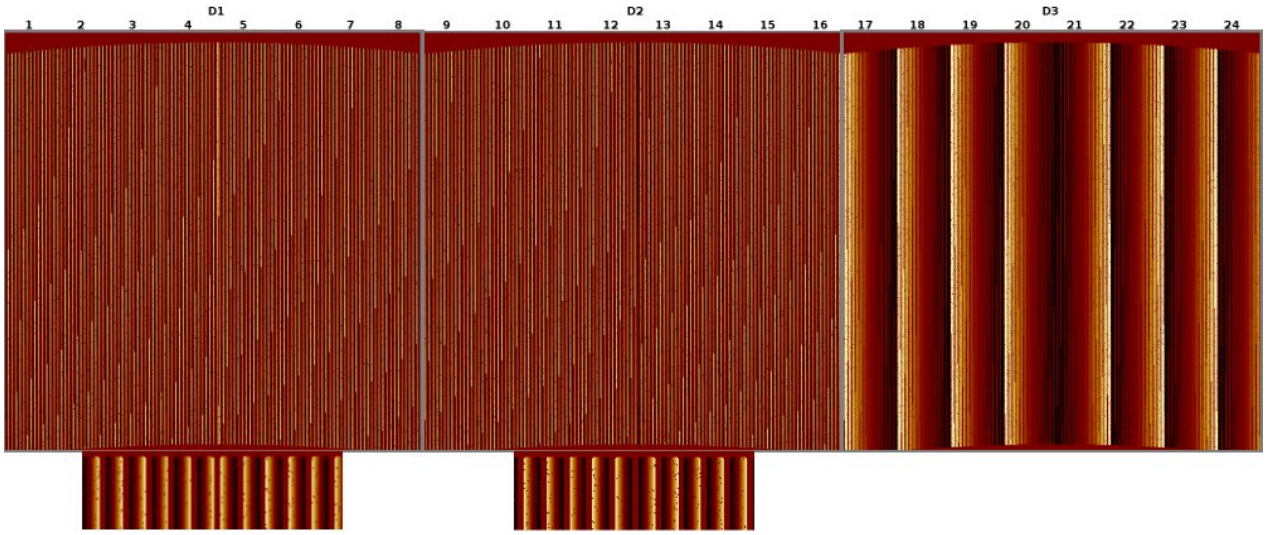

Figure 17 YCAL: The same pattern as above is observed but just switched between the detectors.

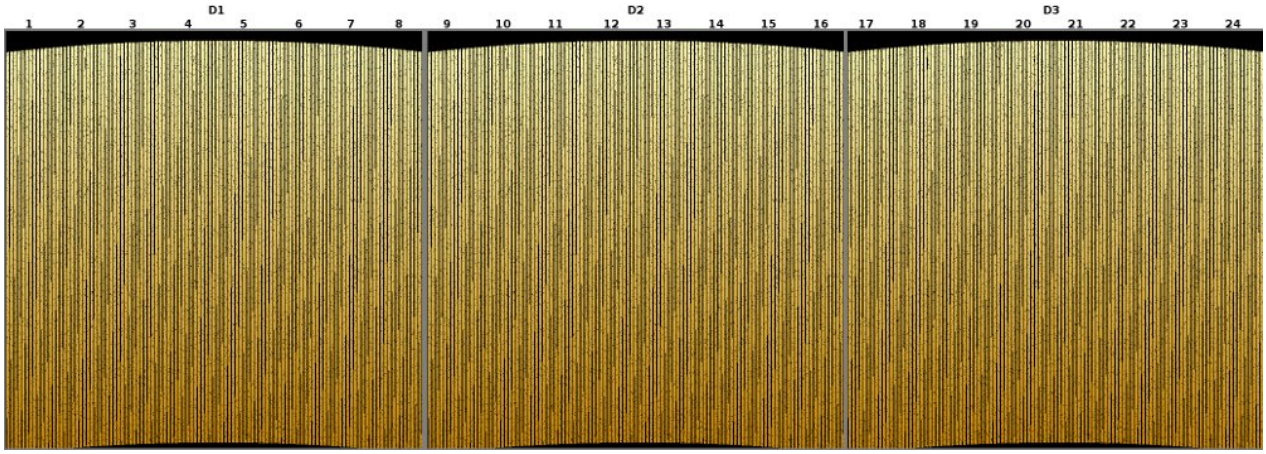

Figure 18 LCAL: The gradiant extends over the wavelength axis in the same way for all detectors, IFUs and slitlets.

4.5.1 Calibration at multiple rotator angles

Calibrations should as well compensate as much as possible for the flexure. There are two main sources of flexure. On one hand the flexure of the whole instrument as such and on the other hand the flexure introduced inside the instrument (spectrographs, arms etc.).

Compensating the outer flexure hasn't been solved satisfactory until now since the characteristics are quite inpredictable. The flexure results in offset of about 1 pix, which lie in the specifications of the instrument.

To compensate for the inner flexure the KMOS DRS pipline allows to take calibration exposures at several rotator angles and to process them in a single run of the associated recipes. The recipes applying reconstruction will then choose the closest calibration frames regarding the rotator angle of the input science frame to reconstruct. If the angle lies inbetween two calibrations, the calibration frames will be temporarily interpolated.

The calibration frames XCAL, YCAL and LCAL generated with the recipes kmos_flat and kmos_wave_cal therefore don't just contain 3 extensions each, but rather 3 times the number of rotator angles.

A default value of 6 rotator angles at an increment step of 60° has proved to be sufficient.

4.6 Instrument spectral resolution

The KMOS line spread function has been parametrized as function of wavelength with a Gaussian function for all the gratings, IFUs, and 6 rotator angles. The median over the IFU field of view is computed. It is saved in static calibrations with PRO.CATG=KERNEL_LIBRARY. Each grating has its own kernel library. In each library, the various extensions contain the line spread function for a given IFU and rotator angle combination. The y-axis is the dispersion direction. At each wavelength, the line spread function is parameterixed with a Gaussian function 15 pixels wide and normalized to 1. The size of the pixel in the x direction in units of nanometer is the same as the y-direction.

4.7 RTD Data Formats

Data which will be displayed in the RTD does not have a designated type since it is not archived, nor does it play a role in the pipeline processing of the science OBs. The formats are included here for completeness and to clarify how the data will appear in the RTD.
There will be 2 RTDs for KMOS.
The first will display the raw data, which will appear as a single frame, from which the contributions from the 3 detectors (each 2048×2048 pixels) are spliced together in a row, making a frame of 6144×2048 pixels as shown in Figure 19.

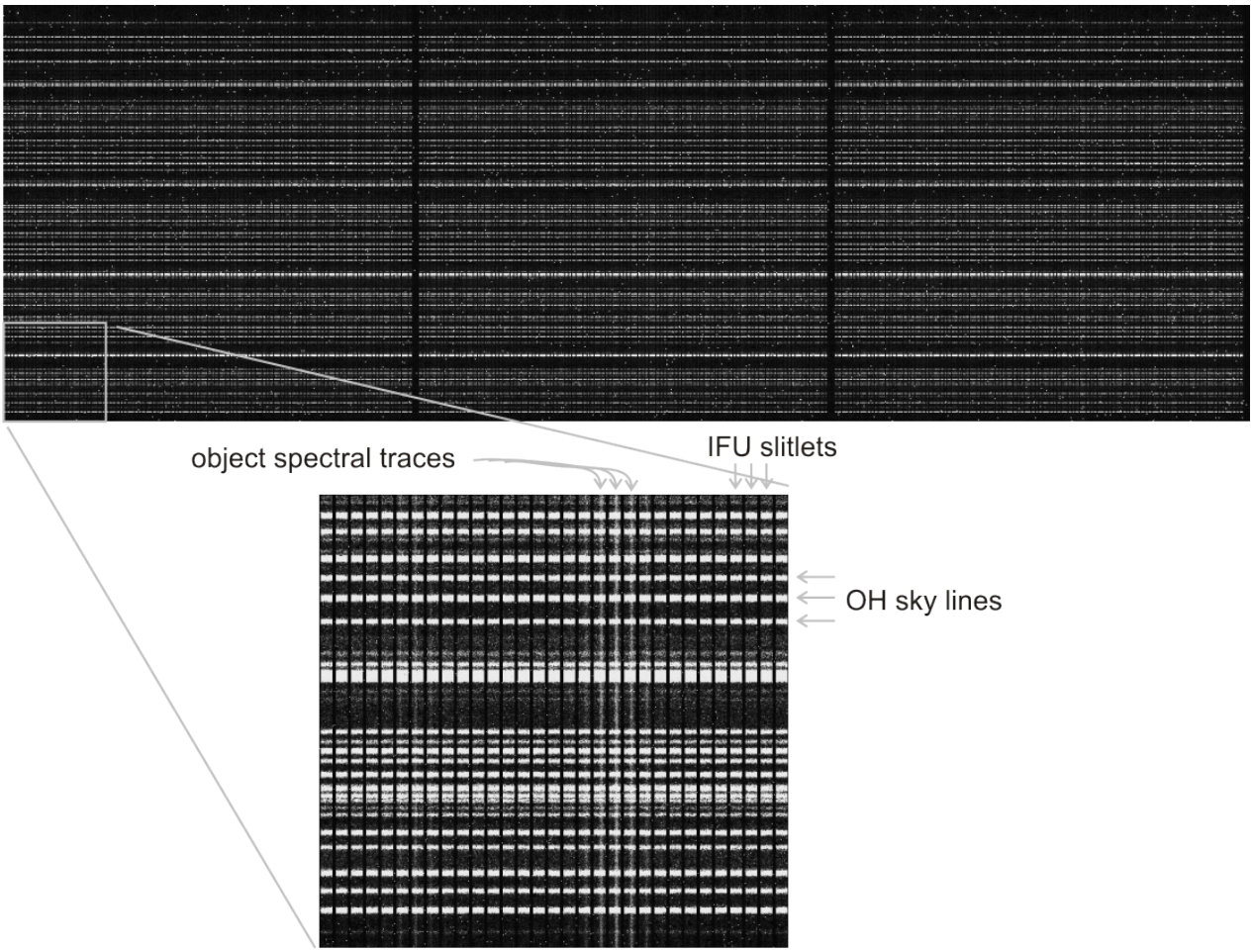

Figure 19: illustrative example of how the raw data appears in the first RTD (top), with the 3 detector frames spliced together. Below is shown a zoom of one part of this, in which it is possible to distinguish individual slitlets from the IFUs, the OH lines, and the spectral traces of 1 or 2 objects. Note that no curvature has been included in this example; the actual curvature will be small.

The second RTD shows the reconstructed images. There is a button so that the user can choose between seeing these images in a grid (Figure 20) or in their actual location within the patrol field (Figure 21).

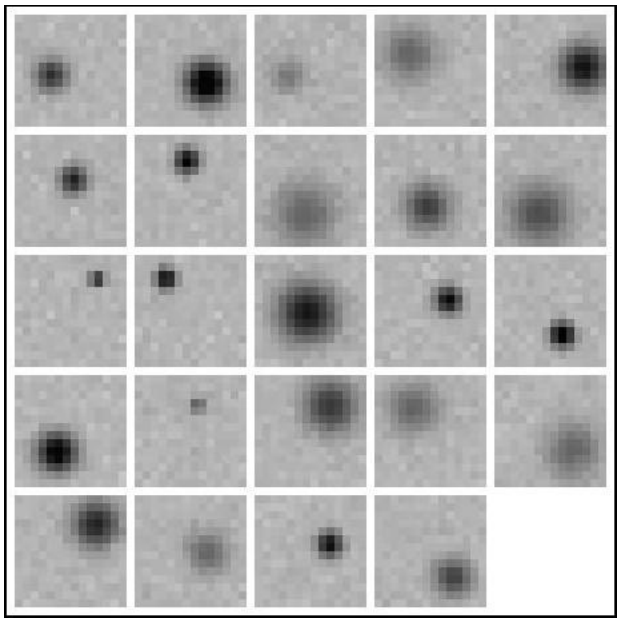

Figure 20: reconstructed images from the 24 IFUs displayed in a 5×5 grid format. This allows one to see immediately and easily what each IFU is looking at.

Each sub-image of the grid-format will be 14×14 pixels. Since a spacing of 1 pixel is included between each sub-image, the whole montage will be 76×76 pixels. Any sub-images which are not reconstructed (e.g. during acquisition, typically only a few IFUs will be used) will be left blank. Thus the position of a sub-image for a particular IFU will *always* be the same, regardless of how many are reconstructed.

The patrol field format will cover 7.2arcmin (plus some extra blank space) at a sampling of 0.2" which matches that of the individual reconstructed images. Thus it will be 2200×2200 pixels. The sub-images will be inserted at the nearest integer position to their actual locations. This is done to avoid the necessity of resampling the reconstructed images, and because this accuracy (i.e. to half a pixel, or 0.1") is sufficient for the purpose of this format.

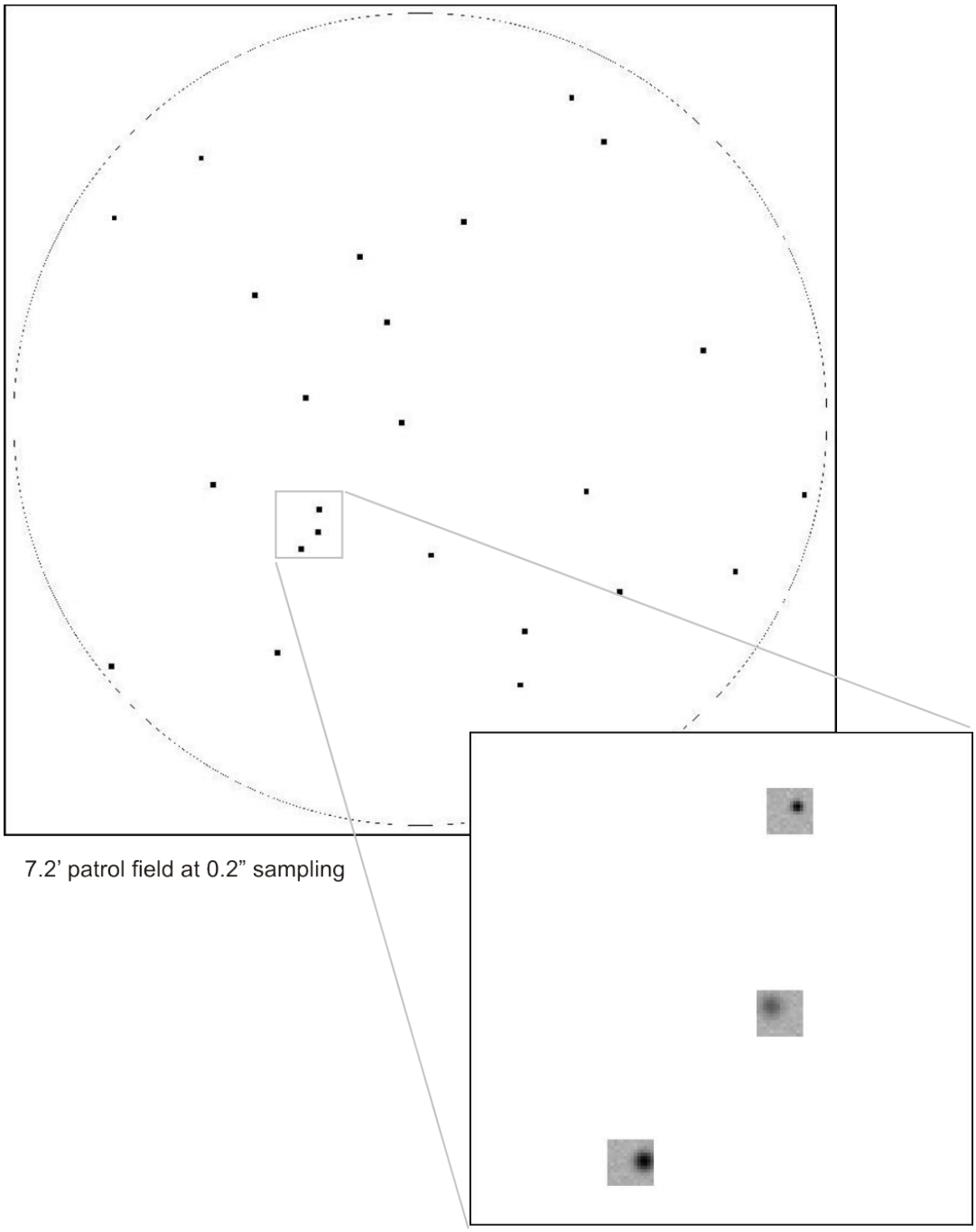

Figure 21: reconstructed images from the 24 IFUs placed in their actual locations within the patrol field. This mode will mostly be used for testing and commissioning KMOS, but may also be useful during certain astronomical acquisitions and observations.

5 Data Reduction Library QC1 Parameters

KMOS has 24 IFUs, each of which has 14 slitlets, giving a total of 336 distinct 2-dimensional spectra. Due to alignment and manufacturing tolerances, the spectral traces and dispersion solutions of these spectra need to be determined independently (e.g. there could be discrete shifts between neighbouring spectra). Furthermore, these parameters will depend on the bandpass used. As a result monitoring all the coefficients of all the fits would yield many thousands of QC1 parameters – which is clearly impractical.

This section concerns the way in which the number of QC1 parameters will be kept to a manageable total. However, it should be realised that in many cases, it is nevertheless necessary to track QC1 parameters separately for

(a) each of the 3 detectors since these correspond, in effect, to optically separate systems.

(b) each of the 5 bandpasses, since many of the optical properties depend on the grating/filter used.

Information about the detector or grating to which each QC1 parameter is associated will be given in the associated PAF.

A concise summary of all the QC1 parameters is given in Appendix B.

5.1 QC1 Parameter descriptions

5.1.1 Dark Frames

`QC DARK`

Direct calculation of the mean value in the Master Dark frame for each detector.
(Stored in each detector header of all created output frames)

`QC DARK MEDIAN`

Direct calculation of the median value in the Master Dark frame for each detector.
(Stored in each detector header of all created output frames)

`QC RON`

Direct calculation of the mean value of the noise of the Master Dark frame for each detector.
(Stored in each detector header of all created output frames)

`QC RON MEDIAN`

Direct calculation of the median value of the noise of the Master Dark frame for each detector.
(Stored in each detector header of all created output frames)

`QC DARKCUR`

Mean value (with iterative rejection) for each detector of a long exposure Master Dark frame, after the Master Dark has been subtracted, divided by the exposure time.
(Stored in each detector header of all created output frames)

`QC BADPIX NCOUNTS`

Total number of pixels in each detector flagged as ‘bad’ in a Master Dark or Master Dark frame. The mimimum number is 32’704, since the four-pixel border around the detector frame (used to monitor detector health) is marked always as bad.
(Stored in each detector headers of all created output frames)

5.1.2 Flat Frames

QC FLAT EFF
The main concern here is whether the brightness of the flatfield lamps has changed, and so a single value suffices for all detectors together. It is defined as the mean normalisation for the Master Flat divided by the exposure time, for each bandpass.
(Stored in the primary headers of all created output frames)

QC FLAT SAT NCOUNTS
This parameter tracks how many of the 12 million pixels (in all three detectors) are saturated in the Master Flat, for each bandpass. It allows one to set the optimal exposure time (DIT). A pixel is flagged as saturated if its value is above some defined limit in at least two of the individual ON frames used to generate the Master Flat.
(Stored in the primary header s of all created output frames)

QC FLAT SN
This parameter tracks the signal-to-noise in the illuminated regions of the Master Flat, for each bandpass. It is defined as the total signal in these regions divided by the total noise (i.e. every illuminated pixel is given equal weighting). This will allow one to monitor whether the signal-to-noise in the flatfield meets the required specification, and adjust the number of co-adds (NDIT) appropriately.
(Stored in the primary headers of all created output frames)

QC GAP MEAN, QC GAP SDV, QC GAP MAXDEV
QC SLIT MEAN, QC SLIT SDV, QC SLIT MAXDEV
For all detected edges the width of gaps and slitlets are determined using the fitted polynomial functions. Deviant values are rejected. Then the mean, standard deviation and maximum deviation (in units of standard deviation) are calculated and Y will be compared to nominal values stored in external files (see Section 4 lower table and Section 4.3 for the data format), which will be determined during testing and updated during commissioning. This will yield two sets of numbers which ideally would have a small scatter about zero.
These 6 parameters are sufficient to monitor changes in spectral curvature solution for each of the detectors and bandpasses.
(Stored in each detector header of all created output frames)

5.1.3 Wavelength Calibration

QC ARC AR EFF,
QC ARC NE EFF
The main concern here is whether the brightness of the argon and neon arc lamps has changed, and so a single value for each lamp suffices for all detectors together. They are defined as the total counts of several specified lines, divided by the exposure time, for each bandpass.
(Stored in the primary header)

QC ARC SAT NCOUNTS
This parameter tracks how many of the 12 million pixels (in all three detectors) are saturated in the arc frame, for each bandpass. It allows one to set the optimal exposure time (DIT). A pixel is flagged as saturated if its value is above some defined limit in at least two of the individual ON frames used to generate the arc frame.
(Stored in the primary header)

QC ARC AR SPECRES, QC ARC AR ERR SPECRES,

QC ARC NE SPECRES, QC ARC NE ERR SPECRES
This monitors the spectral resolution and its errors of each grating for both the argon and neon lamp. The FWHM of a specified arc line is measured for each bandpass and each detector.
(Stored in each detector header)

QC ARC DISP0 MEAN, QC ARC DISP0 SDV, QC ARC DISP0 MAXDEV,
QC ARC DISP1 MEAN, QC ARC DISP1 SDV, QC ARC DISP1 MAXDEV,
QC ARC DISP2 MEAN, QC ARC DISP2 SDV, QC ARC DISP2 MAXDEV
For each slitlet, a set of coefficients relating the pixel position on the detector to its wavelength is determined. The constant (zeroth order), first order, and second order coefficients in Y will be compared to nominal values stored in external files (see Section 4 lower table and Section 4.3 for the data format), which will be determined during testing and updated during commissioning. This will yield three sets of numbers which ideally would have a small scatter about zero.
These 9 parameters are sufficient to monitor changes in dispersion solution for each of the detectors and bandpasses.
(Stored in each detector header)

QC ARC MAX DIFF, QC ARC MAX DIFF ID,
QC ARC MAX SDV, QC ARC MAX SDV ID,
QC ARC MEAN DIFF,
QC ARC MEAN SDV
Once the wavelength calibration look-up table has been generated, the arc frame is reconstructed into a cube. Several prominent arc lines will be used to check the quality of the wavelength calibration. For each IFU the difference between the wavelength of the emission line (across all spaxels) and its true wavelength will be measured. The maximum difference, and the corresponding IFU identity will be written as QC parameters. Similarly, the standard deviation of the wavelengths in each spaxel will be calculated. The maximum value and the identity of the corresponding IFU will be written to a second pair of QC parameters.
These 4 parameters are sufficient to monitor the quality of the dispersion solution for each of the detectors and bandpasses. Similarly, the mean difference and the mean standard deviation are calculated.
(Stored in each detector header)

5.1.4 Illumination Correction

QC SPAT UNIF
This parameter is defined as the RMS of all 14×14 spatial pixels in all the illumination correction images corresponding to the 24 IFUs. It is a simple measure of how uniform the Master Flat is, for each bandpass. It is also sensitive to differences in throughput (e.g. due to vignetting) both between IFUs, and within any individual IFU.
(Stored in the primary header of the created output frame)

QC SPAT MAX DEV,
QC SPAT MAX DEV ID
For these parameters, the mean of the illumination correction is calculated for each of the IFUs in each bandpass. The IFU that deviates most from unity is flagged, as is the amount by which it deviates.
(Stored in the primary header of the created output frame)

QC SPAT MAX NONUNIF,
QC SPAT MAX NONUNIF ID

For these parameters, the standard deviation of the illumination correction is calculated for each of the IFUs in each bandpass. The IFU with the largest standard deviation is flagged, and the standard deviation itself is also recorded.
(Stored in the primary header of the created output frame)

5.1.5 Standard Star Observations

`QC ZPOINT`
This is defined as the mean zeropoint of all standard stars observed in various IFUs for a single pointing, and for which a magnitude is given (although the number of stars may typically be 1). It is different for each bandpass.
(Stored in each detector header of telluric output frame)

`QC THRUPUT, QC THRUPUT MEAN, QC THRUPUT SDV`
This is equivalent to the zeropoint, but in a slightly different form. The throughput will be calculated whenever the zeropoint is calculated. It will be given as the mean (and standard deviation) of the throughput based on all standard stars observed in various IFUs for a single pointing – as long as a magnitude and spectral type is given. The number of photons detected (i.e. counts × gain) will be compared to the number of photons expected from the star, taking into account standard atmospheric extinction. The ratio of these numbers is the throughput from the top of the telescope to the detector, including the detector quantum efficiency.
(`QC THROUGHPUT` is stored in each detector header of telluric output frame,
`QC THROUGHPUT MEAN` and `QC THROUGHPUT SDV` are stored in the primary header of telluric output frame)

`QC SPAT RES`
This is defined as the mean FWHM resolution of all standard stars observed in a single pointing. Although the PSF may be slightly elliptical, the FWHM along the two axes are averaged to yield a single measurement.
(Stored in each detector header of PSF output frame)

`QC STD TRACE`
This QC1 parameter has been introduced to verify the spectral curvature solution by checking whether the trace of a standard star is straight in the reconstructed cube. Note that in the near infrared, differential atmospheric refraction is small and will have little impact on the trace. This parameter measures the standard deviation of the measurements of the positions of the standard star in each spectral slice. This will depend on the bandpass used.
(Stored in each detector header of PSF output frame)

PART II: DRS RECIPE REFERENCE

6 Preliminaries

All calibration and science recipes receive raw or processed frames as input containing data referring to a complete detector array. While processing, this format can change in the way that a frame will refer to a single IFU (see Sec. 4.2.4 onwards). In this case the recipe iterates over all frames of all IFUs in order to process all data supplied by the detectors. Detector frames will be split up into IFU frames, when a cube has to be reconstructed or created. Cubes refer always to IFUs. Reciprocally IFU frames can be combined to a detector frame again.

Data Types

All generated and saved image and cube frames are of type float. Vector frames and scalar values are of type double.

Adressing of IFUs and detectors

When a specific IFU or detector has to be defined in a recipe, an integer has to be supplied to the recipe. Numbering starts always at 1 and ends at 24 for IFUs and at 3 for detectors.

Invalid IFUs

Since not all IFUs need to be active when doing an exposure, some sections of a RAW frame can contain invalid data. The inactive IFUs are marked in the primary header with ESO OCS ARMi NOTUSED (i=1 to 24).
During the reconstruction of detector frame is split up and rearranged into a cube. Invalid IFUs will just contain the extension header and no data (NAXIS=0). The keywords specific to arms are propagated into the respective extension header.

QC Parameters

The QC parameters generated by the recipes are listed in Appendix B.

6.1 Standard workflow

A standard workflow to setup a calibration pipeline would look like:

```
$ esorex kmos_dark dark.sof
$ esorex kmos_flat flat.sof
$ esorex kmos_wave_cal arc.sof
$ esorex kmos_illumination illumination.sof
$ esorex kmos_std_star std_star.sof
$ esorex kmos_sci_red sci_red.sof
$ esorex kmos_combine combine.sof
```

Reconstructing a data cube from a detector image can already be performed after having executed kmos_wave_cal:

```
$ esorex kmos_reconstruct reconstruct_science.sof
```

```
There is also the option to use molecfit / calctrans to correct for
telluric feature. After kmos_std_star, a typical exectution looks like:

$ esorex kmos_molecfit_model molecfit_model.sof
$ esorex kmos_sci_red sci_red.sof
$ esorex kmos_molecfit_calctrans molecfit_calctrans.sof
$ esorex kmos_molecfit_correct molecfit_correct.sof
$ esorex kmos_combine combine.sof
```

6.2 Generating Test Data

Executing the built-in tests of the pipeline generates automatically valid and invalid test data for the various recipes. Valid data has a prefix "v_" and invalid data has a prefix "i_" (stored in the subfolders in kmosp/recipes/tests/test_data).

Test data is also generated for the calibration pipeline (kmosp/recipes/tests/test_data/pipeline). It consists of simulated K-band data. The pipeline will also be executed during the tests and the products are saved to disk (kmosp/recipes/tests).

To run the tests open a terminal and execute `make check` in kmosp/recipes.

6.3 Predefined wavelength ranges

By default the following wavelength ranges are used to reconstruct detector images into cubes:

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

These values can be changed using the parameters *b_end* and *b_start* in the recipes kmos_reconstruct, kmos_illumination, kmos_std_star , kmos_sci_red.

6.4 Lookup table (LUT) for reconstruction

Once the calibration frames XCAL, YCAL and LCAL have been created with kmos_flat and kmos_wave_cal, any detector image can be reconstructed. As long as the calibration frames don't change, every detector image will be reconstructed exactly the same way. To speed up the interpolation during reconstruction, the generated LUT will be saved to disk by default. In each subsequent reconstruction step this LUT can be reused and hasn't to be recalculated therefore. The LUT is saved as binary file and is not editable. It will neither be declared as ESO DFS product since this is an intermediate output.
When a detector image to reconstruct contains only a few valid IFUs, the LUT is only calculated and stored for these IFUs. In a later run the LUT can be updated when other IFUs are active.

A saved LUT can only be reused when following parameters match

- **Filters, gratings and rotation offset**
  Every LUT is specific to filters, gratings and rotation offset. Therefore the LUT gets the same filename extension like other calibration products, e.g. LUT_HHH_HHH_0.fits
- **Reconstruction method, spatial and spectral ranges**
  These parameters are stored in the LUT and are checked before eventually applying it.

- **Timestamp**
  A timestamp is also added to the LUT to assert that the LUT is newer than the above-mentioned calibration frames. If any of the provided calibration frames is newer than the LUT, then the LUT will be recalculated.

The LUT will be erased, recalculated and saved again when any of these parameters don't match.

There are different modes to influence the behaviour of the usage of the LUT.

- **NONE**
  The initial LUT is neither stored to disk nor in memory.
  This method uses CPU resources only and is therefore the slowest method.
  Any possibly existing LUT on disk will be ignored.
- **MEMORY**
  The initial LUT isn't stored to disk but is kept in memory as long a recipe is executed.
  This method uses system memory resources.
  Any possibly existing LUT on disk will be ignored.
- **FILE**
  The initial LUT will be calculated and directly be saved to disk.
  This method uses file system resources.
  Any possibly existing LUT on disk will be examined for usability.
- **BOTH**
  The initial LUT will be kept in memory as long a recipe is executed and saved to disk.
  This method uses system memory and file system resources.
  Any possibly existing LUT on disk will be examined for usability.

The default is LUT_MODE_FILE. The behavior can be changed in defining an environment variable called KMCLIPM_PRIV_RECONSTRUCT_LUT_MODE with any of the the values declared above.

7 Recipes

The KMOS data processing recipes can be divided into following three categories:

- **Calibration Recipes**
  Recipes, which directly produce either calibration frames needed for science reductions or for QC1 parameters.
- **Science Reduction Recipes**
  Recipes which perform science or acquisition reductions (which are largely built from the tools described below)
- **Common Recipes**
  Functionally simple recipes, which can be applied in a straightforward fashion. These recipes are used internally as well for the calibration and science reduction recipes.

The interactions between the calibration and science reduction recipes are displayed as an association map in Figure 22.
It is worth noting that only the high level functions are described here, and the low level such as basic arithmetic and file manipulation functions, are implied.
In Section 7.1 and 7.1.8 the calibration and science processing recipes are ordered following the workflow of the pipeline. In the following section the recipes are ordered alphabetically.

Reference Structure

For each recipe following information is provided:

- **Functional Description**
  A short and more detailed description of the recipe is given.
- **Flow Chart**
  A graphical flow chart and a corresponding description are provided. The stylistic conventions used in the subsequent flowcharts are as follows:
  - Inputs of single values like float, int etc. external to the recipe data flow are indicated in the flowchart by right filled triangles (►).
  - Inputs of cubes, frames or vectors are indicated by arrows (→).
  - Output of quality control parameters is coloured blue.
  - Data-cubes, -frames or –vectors are displayed with bold typeface.
  - The data flow goes from top to bottom. The down arrows can be split by conditionnal statements (diamond) or when one single output triggers several DRL functions.
- **Input Frames**
  The DO categories of the frames needed to run the recipe and the required KMOS Fits Type.
- **Fits Header Keywords**
  Keywords needed in the primary and subsequent headers of the input files.
- **Configuration Parameters**
  Description of all possible parameters with applicable data formats and allowed values. Where appropriate they are divided into basic and advanced parameters.
- **Output Frames**
  Created output files with their DO category and KMOS Fits Type.
- **Examples**
  How one would call the recipe with Esorex. If input is a single fits-file with no category-keyword, it can simply be appended to the recipe-name. If input consists of a file with category-keyword or of multiple files, they have to be written in a so-called sof-file (set of frames). These are pseudo code examples.

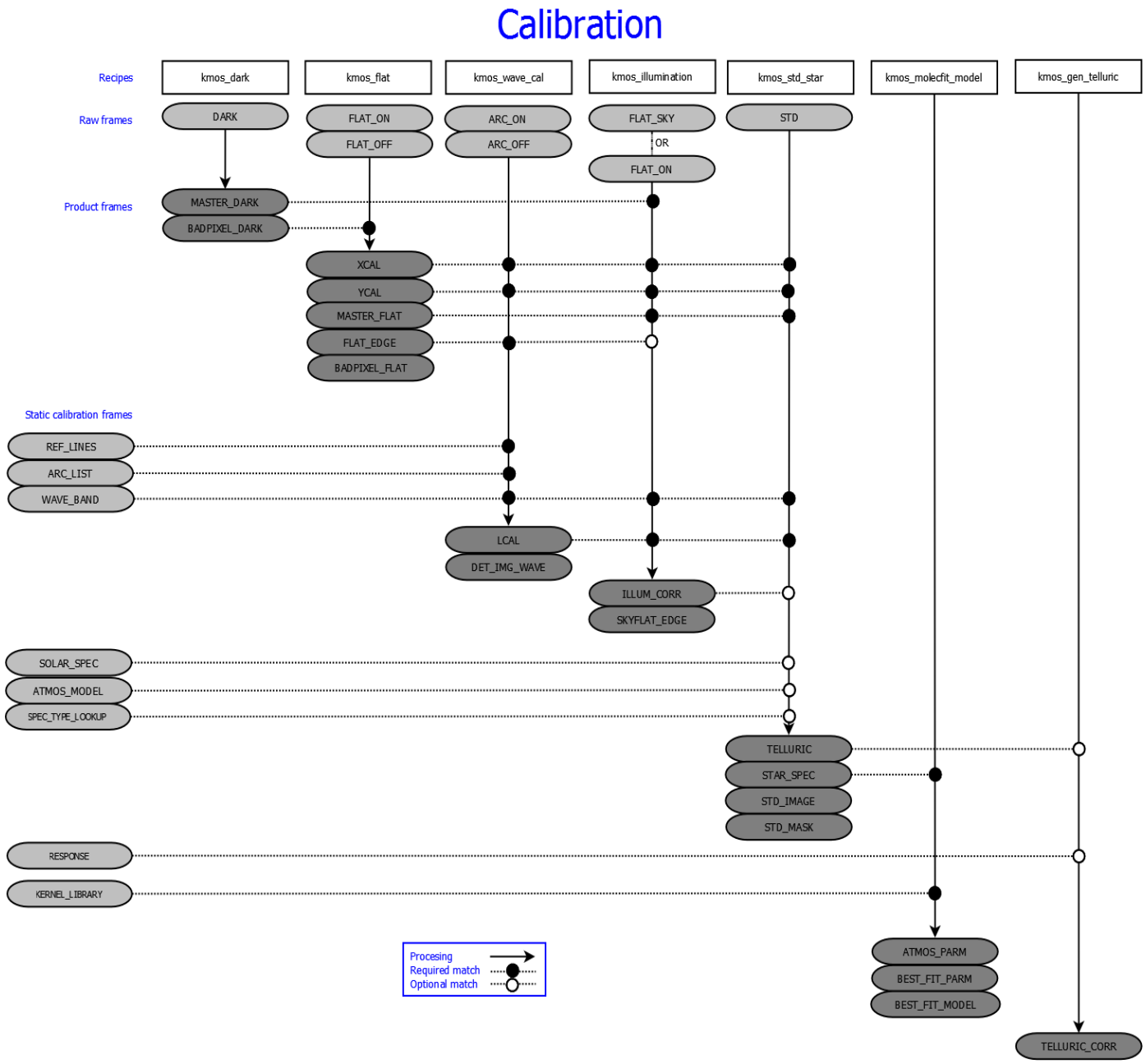

Figure 22**:** Calibration association map for KMOS. The calibration and science recipes are listed, and the interactions between them indicated by the filled circles.

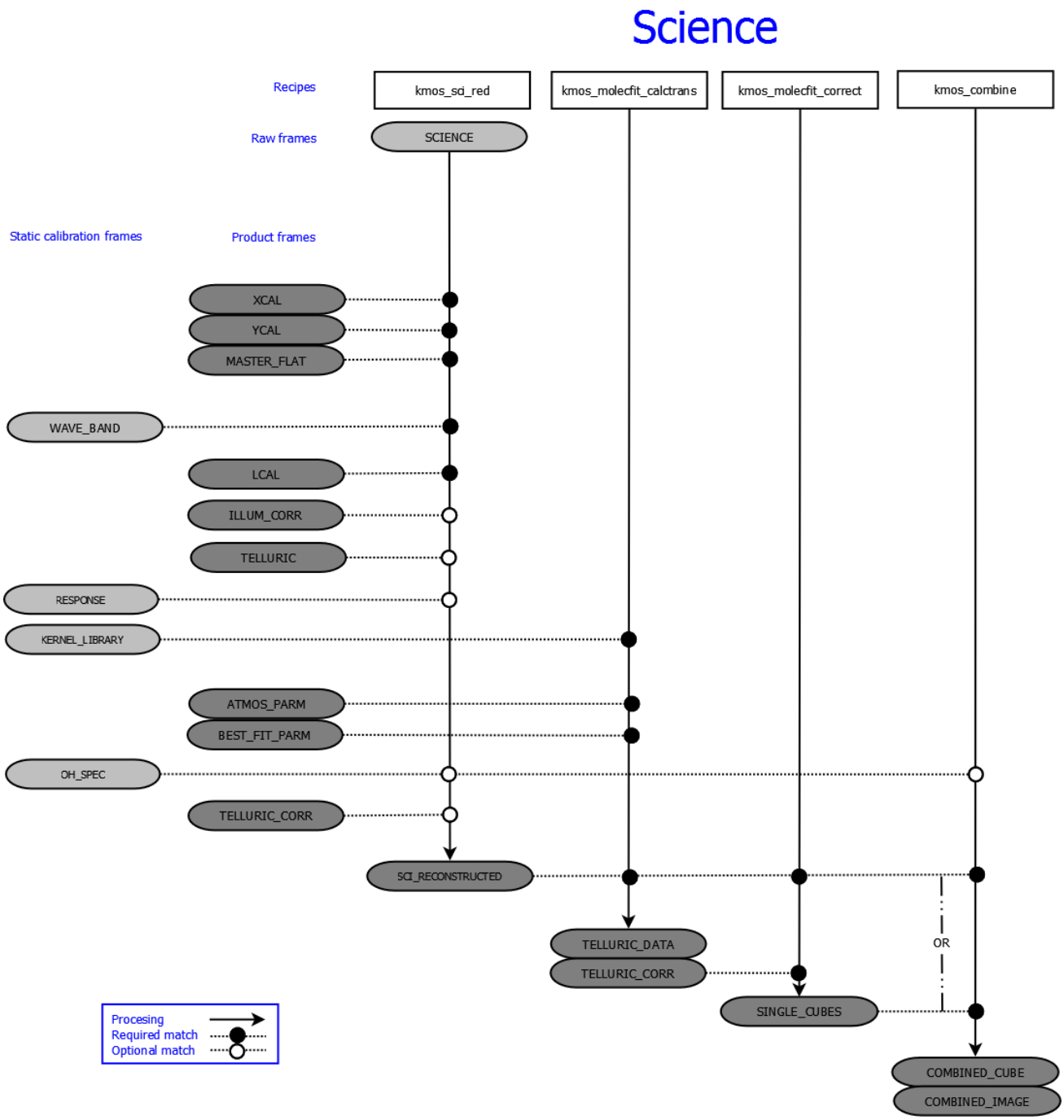

Figure 23**:** Science association map for KMOS. The calibration and science recipes are listed, and the interactions between them indicated by the filled circles.

7.1 Calibration Recipes

7.1.1 kmos_dark: Master Dark Frames

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

7.1.1.1 Description

This recipe calculates the master dark frame.
It is recommended to provide three or more dark exposures to produce a reasonable master with associated noise. See section 8.2 for information on combine less than three frames.

Basic parameters:

`--lcmethod`

The level correction method that needs to be applied to the input frames is specified here. It can be either 'NONE' (for no correction) or 'OSCAN' to apply the overscan correction to the input frames.

`--pos_bad_pix_rej`

`--neg_bad_pix_rej`

Bad pixels above and below defined positive/negative threshold levels will be flagged and output to the BADPIX_DARK frame (which will go into the kmos_flat recipe). The number of bad pixels is returned as a QC1 parameter. The two parameters can be used to change these thresholds.

`--cmethod`

Following methods of frame combination are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:
  `val > mean + stdev * cpos_rej`
  and
  `val < mean - stdev * cneg_rej`
  where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2)
- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.
- **sum**
  At each pixel position the sum is calculated.
- **min_max**
  The specified number of minimum and maximum pixel values will be rejected.
  `--cmax` and `--cmin` apply to this method.

`--file_extension`

Set this parameter to TRUE if the EXPTIME keyword should be appended to the output filenames.

Advanced parameters:

`--cpos_rej`

`--cneg_rej`

`--citer`
see `--cmethod` = ”ksigma”

`--cmax`
`--cmin`
See `--cmethod` = “min_max”

7.1.1.2 Flow Chart

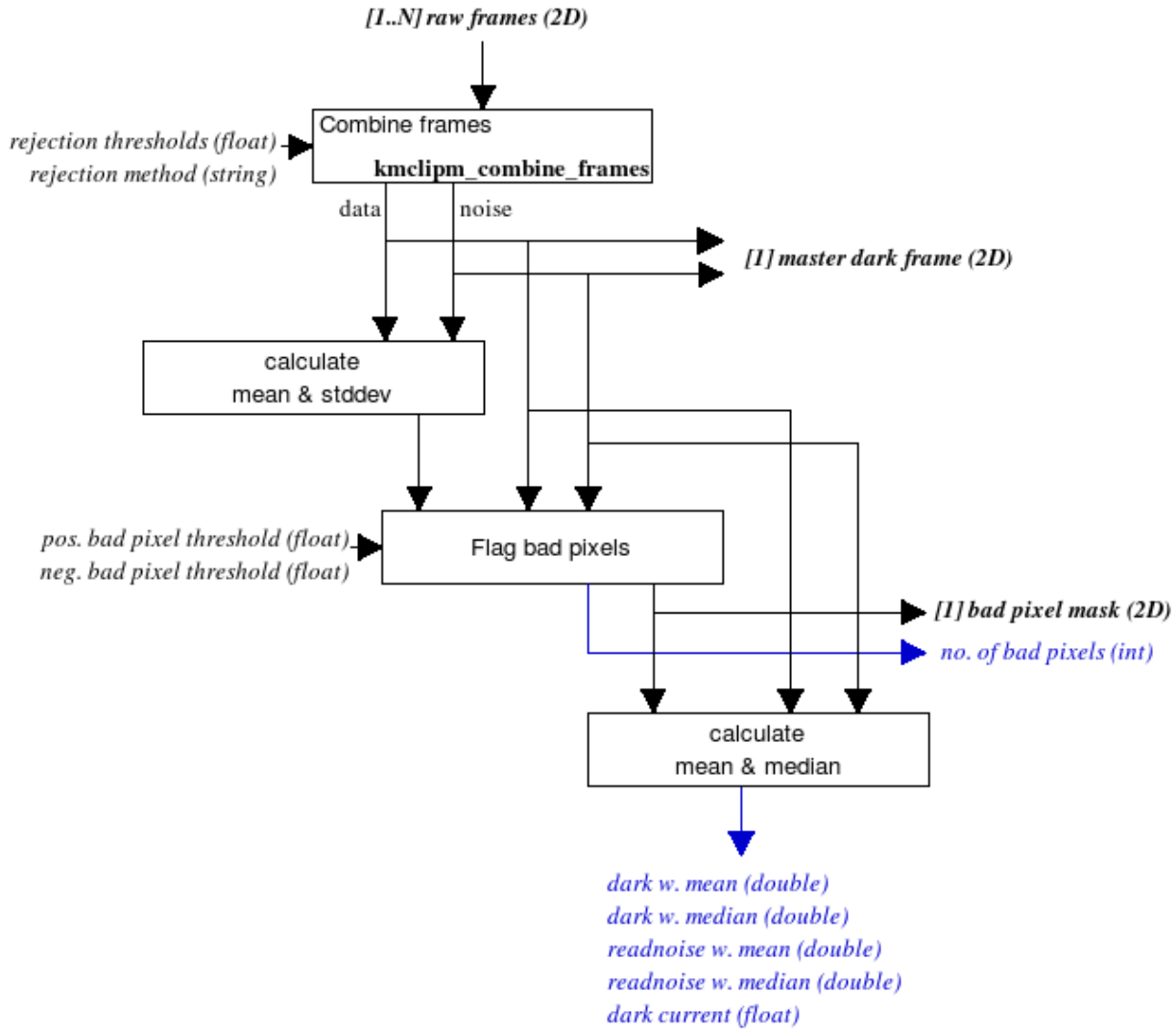

Figure 24: Flow chart of the recipe kmos_dark

The processing steps are:

1. From a series of dark exposures a dark frame (mean) and a noise map (std err) are calculated using pixel rejection.
2. Then bad pixels above and below defined positive/negative threshold levels will be flagged and output to the temporary bad pixel mask (which will go into the kmos_flat recipe). The number of bad pixels is returned as QC1 parameter.
3. Bias, readnoise and the dark current quality parameters will be calculated (see section 5.1.1 for comprehensive explanations on QC1 parameters).

7.1.1.3 Input Frames

KMOS type	DO category	Amount	Comments
RAW	DARK	≥ 1 (≥ 3 recommended)	dark exposures

7.1.1.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
NDIT	int	any	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
EXPTIME	double	any	

7.1.1.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
pos_bad_pix_rej, neg_bad_pix_rej	double	pos_bad_pix_rej ≥ 0, neg_bad_pix_rej ≥ 0	50.0 50.0	The positive and negative rejection threshold for bad pixels. *(optional)*
cmethod	string	"ksigma", "average", "min_max", "sum", "median"	"ksigma"	The averaging method to apply *(optional)*
file_extension	bool	TRUE, FALSE	FALSE	*(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = "ksigma")*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only iwhen --cmethod = "ksigma")*
cmax cmin	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip with min/max-clipping *(optional, applies only when --cmethod = "min_max")*

7.1.1.6 Output Frames

KMOS type	DO Category	Comments
F2D	MASTER_DARK	Calculated master dark frames (with included noise frames)
B2D	BADPIXEL_DARK	Associated badpixel frames

7.1.1.7 Examples

```
$ esorex kmos_dark -pos_bad_pix_rej=2.1 dark.sof
                                                with dark.sof containing:
                                                dark_1.fits          DARK
                                                dark_2.fits          DARK
                                                dark_3.fits          DARK
```

7.1.2 kmos_flat: Master Flat Field

Create master flatfield frame and badpixel map to be used during science reduction.

7.1.2.1 Description

This recipe creates the master flat field and calibration frames needed for spatial calibration for all three detectors. It must be called after the kmos_dark-recipe, which generates a bad pixel mask (badpixel_dark.fits). The bad pixel mask will be updated in this recipe (goes into badpixel_flat.fits). As input at least 3 dark frames, 3 frames with the flat lamp on are recommended. Additionally a badpixel mask from kmos_dark is required.

In order to correct instrument flexure, the flat lamp on frames can be taken at different rotator angles and can be feed to the recipe in one go. For each rotator angle there will be 3 extensions, one for each detector, for every rotator angle. It is recommended to take calibration exposures in 60 degree increments, resulting in a set of 6 rotator angles. It is important, that the same angles are chosen for kmos_flat and kmos_wave_cal.

The badpixel mask contains 0 for bad pixels and 1 for good ones.

The structure of the resulting xcal and ycal frames is quite complex since the arrangement of the IFUs isn't just linear on the detector. Basically the integer part of the calibration data shows the offset of each pixels centre in mas (milli arcsec) from the field centre. The viewing of an IFU is 2800mas (14pix*0.2arcsec/pix). So the values in these two frames will vary between +/-1500 (One would expect 1400, but since the slitlets aren't expected to be exactly vertical, the values can even go up to around 1500). Additionally in the calibration data in y-direction the decimal part of the data designates the IFU to which the slitlet corresponds to (for each detector from 1 to 8). Because of the irregular arrangement of the IFUs not all x-direction calibration data is found in xcal and similarly not all y-direction calibration data is located in ycal. For certain IFUs they are switched and/or flipped in x- or y-direction:

For IFUs 1,2,3,4,13,14,15,16:	x- and y- data is switched
For IFUs 17,18,19,20:	y-data is flipped
For IFUs 21,22,23,24:	x-data is flipped
For IFUs 5,6,7,8,9,10,11,12:	x- and y- data is switched and x- and y- data is flipped

Furthermore frames can be provided for several rotator angles. In this case the resulting calibration frames for each detector are repeatedly saved as extension for every angle.

Advanced features:

To create the badpixel mask the edges of all slitlets are fitted to a polynomial. Since it can happen that some of these fits (3 detectors * 8 IFUs * 14slitlets * 2 edges (left and right edge of slitlet)= 672 edges) fail, the fit parameters are themselves fitted again to detect any outliers. By default the parameters of all left and all right edges are grouped individually and then fitted using chebyshev polynomials. The advantage of a chebyshev polynomial is, that it consists in fact of a series of orthogonal polynomials. This implies that the parameters of the polynomials are independent. This fact predestines the use of chebyshev polynomials for our case. So each individual parameter can be examined independently. The reason why the left and right edges are fitted individually is that there is a systematic pattern specific to these groups. The reason for this pattern is probably to be found in the optical path the light is traversing.
The behaviour of this fitting step can be influenced via environment parameters:

- KF_ALLPARS (default: 1)
  When set to 1 all coefficients of the polynomial of an edge are to be corrected, also when

just one of these coefficients is an outlier. When set to 0 only the outlier is to b e corrected.

- KF_CH (default: 1)
  When set to 1 chebyshev polynomials are used to fit the fitted parameters. When set to 0 normal polynomials are used.
- KF_SIDES (default: 2)
  This variable can either be set to 1 or 2. When set to 2 the left and right edges are examined individually. When set to 1 all edges are examined as one group.
- KF_FACTOR (default: 4)
  This factor defines the threshold factor. All parameters deviating KF_FACTOR*stddev are to be corrected.

Basic parameters:

`--badpix_thresh`

The threshold level to mark pixels as bad on the dark subtracted frames [%].

`--surrounding_pixels`

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

`--cmethod`

Following methods of frame combination are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:
  `val > mean + stdev * cpos_rej`
  and
  `val < mean - stdev * cneg_rej`
  where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2).
- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.
- **sum**
  At each pixel position the sum is calculated.
- **min_max**
  The specified number of minimum and maximum pixel values will be rejected.
  `--cmax` and `--cmin` apply to this method.

Advanced parameters:

`--cpos_rej`
`--cneg_rej`
`--citer`

see `--cmethod` = "ksigma"

`--cmax`
`--cmin`

see `--cmethod` = "min_max"

`--suppress_extension`

If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0

7.1.2.2 Flow Chart

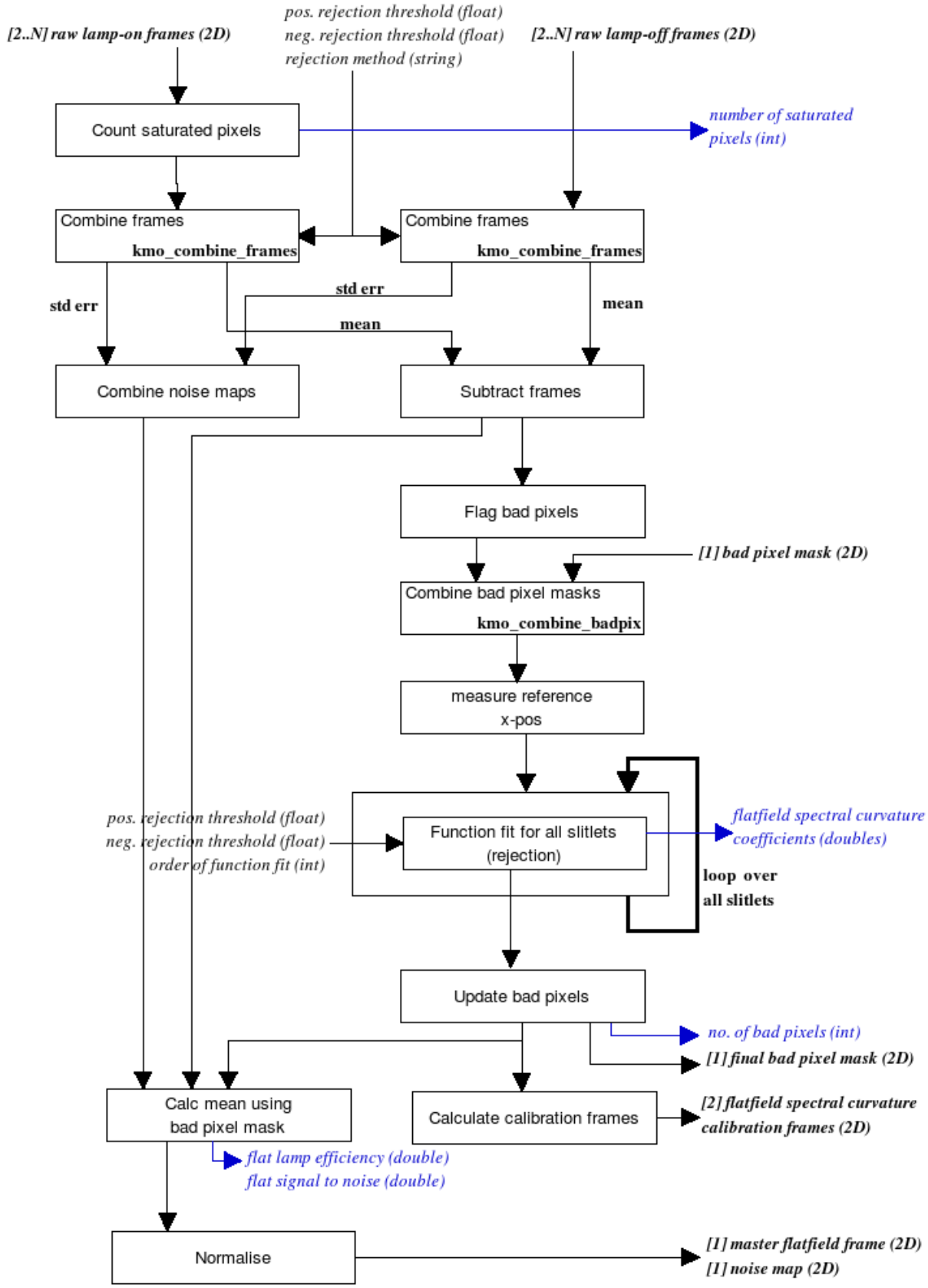

Figure 25**:** Flow chart of the recipe kmos_flat

The processing steps are:

1. The number of saturated pixels (>50'000) in the raw lamp-on frames is counted.
2. The mean frame and associated noise map (std err) from the lamp-on frames are calculated using pixel rejection.
3. Similarly, the mean frame and associated noise map from lamp-off frames will be computed.
4. The two mean frames are subtracted. The noise frames are combined.
5. To flag bad pixels preliminarily, the subtracted data is sorted. The lower 5% and upper 10% are cut off and then the position with the steepest slope is searched. 10% of the value at this position is taken as threshold level. Pixels below will be flagged as bad pixels. Additionally all pixels surrounded by at least 6 bad pixels are also flagged as bad. This bad pixel mask will be combined with the temporary bad pixel mask from kmos_dark recipe resulting into a preliminary bad pixel mask. (Preliminary because the slitlets are to wide at present, but the exact edges are calculated with the fitted edge information afterwards)
6. In the middle of the lower half und upper half of the frame a line profile is taken and analysed for eventually existing rotation, cut or missing slitlets. When the number of slitlets present and their approximate position has been determined, along the y-axis every 9 pixels a gaussfit is done to get a better approximation of the edge. At a last step, a 3rd order polynomial is fitted to the edge. Out of the parameters of the polynomial the QC parameters QC GAP MEAN, QC GAP SDV, QC GAP MAXDEV, QC SLIT MEAN, QC SLIT SDV, QC SLIT MAXDEV are calculated.
7. Now knowing the exact position and shape of the edge, the bad pixel mask is updated and the spectral curvature calibration frames (LUTs), one in x- and one in y-direction, are calculated. Furthermore an eventually existing spectral gradient will be normalised for each slitlet separately. For this all values in the same row of a slitlet are averaged, then a 3rd order polynomial is fitted to the resulting data points. The polynomial is normalised and the slitlet-data will be divided by it.

Now the data and noise frames are normalised as a whole to unity using the mean calculated without bad pixels. Out of these operations we get the master flatfield frame, the noise map and QC1 parameters indicating lamp efficiency and signal to noise.

7.1.2.3 Input Frames

KMOS type	DO category	Amount	Comments
RAW	FLAT_ON	≥ 1 (≥ 3 recommended)	Flatlamp-on frames, optimally at least 3 for every rotator angle
RAW	FLAT_OFF	≥ 1 (≥ 3 recommended)	Flatlamp-off frames (dark exposures)
B2D	BADPIXEL_DARK	1	badpixel frame (from kmos_dark)

7.1.2.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
DIT	double	any	integration time (equals EXPTIME)
NDIT	int	1	
ESO DET READ CURNAME	string	Double, Fowler, Nondest	detector readout mode

ESO INS LAMP1 ST	bool	FALSE	Arc lamp must be off
ESO INS LAMP2 ST	bool	FALSE	Arc lamp must be off
ESO INS LAMP3 ST	bool	TRUE	FLAT_ON: Flat lamp must be on FLAT_OFF: must be off (can be on if ESO INS FILTx ID is 'Block')
ESO INS LAMP4 ST	bool	TRUE	Either LAMP3 or LAMP4 must be on (LAMP4 is a spare)

Sub Headers

None

7.1.2.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
badpix_thresh	int	100 ≥ badpix_thresh ≥ 0	35	in percent
surrounding_pixels	int	8 ≥ surrounding_pixels ≥ 0	5	The amount of bad pixels to surround a specific pixel, to let it be marked bad as well *(optional)*
cmethod	string	"ksigma" "min_max" "average" "median" "sum"	"ksigma"	The averaging method to apply *(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = "ksigma")*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only when --cmethod = "ksigma")*
cmax cmin	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip with min/max-clipping *(optional, applies only when --cmethod = "min_max")*
suppress_extension	bool	TRUE, FALSE	FALSE	

7.1.2.6 Output Frames

KMOS type	DO Category	Comments
F2D	MASTER_FLAT	Normalised flat field (with included noise frames)
B2D	BADPIXEL_FLAT	Updated bad pixel mask
F2D	XCAL	Calibration frame 1 (spatial dimension)
F2D	YCAL	Calibration frame 2 (spatial dimension)
F2L	FLAT_EDGE	Intermediate product needed for kmos_wave_cal and optionally for kmos_illumination. It contains the parameters of the fitted edges of all IFUs of all detectors.

7.1.2.7 Examples

```
$ esorex kmos_flat flat.sof
```

```
with flat.sof containing:
flat_off_1.fits             FLAT_OFF
flat_off_2.fits             FLAT_OFF
flat_off_3.fits             FLAT_OFF
flat_on_1_0deg.fits         FLAT_ON
flat_on_2_0deg.fits         FLAT_ON
flat_on_3_0deg.fits         FLAT_ON
flat_on_1_60deg.fits        FLAT_ON
flat_on_2_60deg.fits        FLAT_ON
flat_on_3_60deg.fits        FLAT_ON
flat_on_1_120deg.fits       FLAT_ON
flat_on_2_120deg.fits       FLAT_ON
flat_on_3_120deg.fits       FLAT_ON
flat_on_1_180deg.fits       FLAT_ON
flat_on_2_180deg.fits       FLAT_ON
flat_on_3_180deg.fits       FLAT_ON
flat_on_1_240deg.fits       FLAT_ON
flat_on_2_240deg.fits       FLAT_ON
flat_on_3_240deg.fits       FLAT_ON
flat_on_1_300deg.fits       FLAT_ON
flat_on_2_300deg.fits       FLAT_ON
flat_on_3_300deg.fits       FLAT_ON
badpixel_dark.fits          BADPIXEL_DARK
```

7.1.3 kmos_wave_cal: Wavelength Calibration

Create a calibration frame encoding the spectral position (i.e. wavelength) of each pixel on the detector.

7.1.3.1 Description

This recipe creates the wavelength calibration frame needed for all three detectors. It must be called after the kmos_flat recipe, which generates the two spatial calibration frames needed in this recipe. As input a lamp-on frame, a lamp-off frame, the flat badpixel frame, the spatial calibration frames and the list with the reference arclines are required.

In order to correct instrument flexure, the flat lamp on frames can be taken at different rotator angles and can be feed to the recipe in one go. For each rotator angle there will be 3 extensions, one for each detector, for every rotator angle. It is recommended to take calibration exposures in 60 degree increments, resulting in a set of 6 rotator angles. It is important, that the same angles are chosen for kmos_flat and kmos_wave_cal.

An additional output frame is the resampled image of the reconstructed arc frame. All slitlets of all IFUs are aligned one next to the other. This frame serves for quality control. One can immediately see if the calibration was successful.
The lists of reference arclines are supposed to contain the lines for both available calibration arc-lamps, i.e. Argon and Neon. The list is supposed to be a F2L KMOS FITS file with three columns:

1. Reference wavelength
2. Relative strength
3. String either containing “Ar” or “Ne”

The recipe extracts, based on the header keywords, either the applying argon and/or neon emission lines. Below are the plots of the emission lines for both argon and neon. The marked lines are the ones used for wavelength calibration.

Furthermore frames can be provided for several rotator angles. In this case the resulting calibration frames for each detector are repeatedly saved as extension for every angle.

Basic parameters:

`--order`

The polynomial order to use for the fit of the wavelength solution. If the special value 0 is used, the appropriate order is chosen automatically depending on the waveband. Otherwise an order of 6 is recommended, except for IZ-band, there order 4 should be used.

Advanced parameters:

`--b_samples`

The number of samples in spectral direction for the resampled image. Ideally this number should be about the same size as the detector.

`--b_start`
`--b_end`

Used to define manually the start and end wavelength for the resampled image. By default the internally defined values are used (see Section 6.3).

`--suppress_extension`

If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0

The lines used to determine the quality of wavelength calibration are as follows:

Band	Argon	Neon
H	1.67446 um	1.71666 um
HK	1.79196 um	1.80882 um
IZ	0.922703 um	0.85676 um
K	2.15401 um	2.25365 um
YJ	1.21430 um	1.17700 um

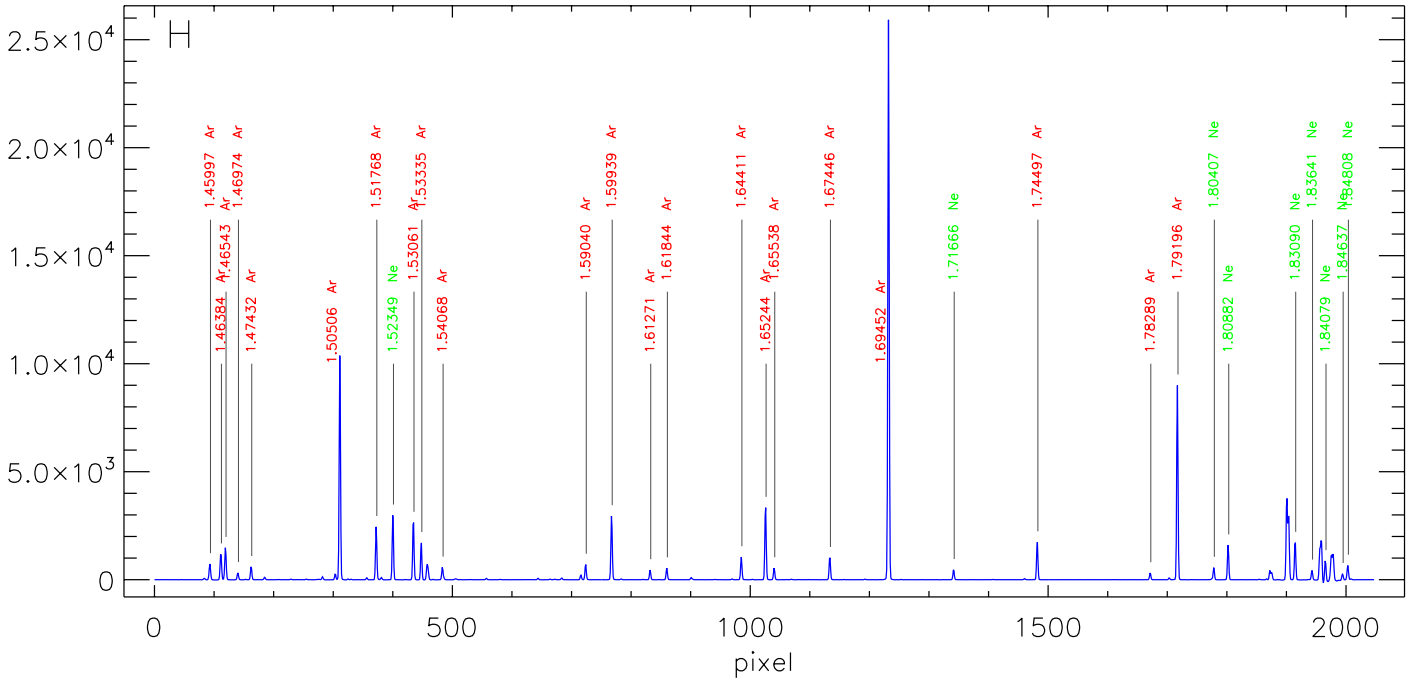

Figure 26: H-band argon and neon emission lines

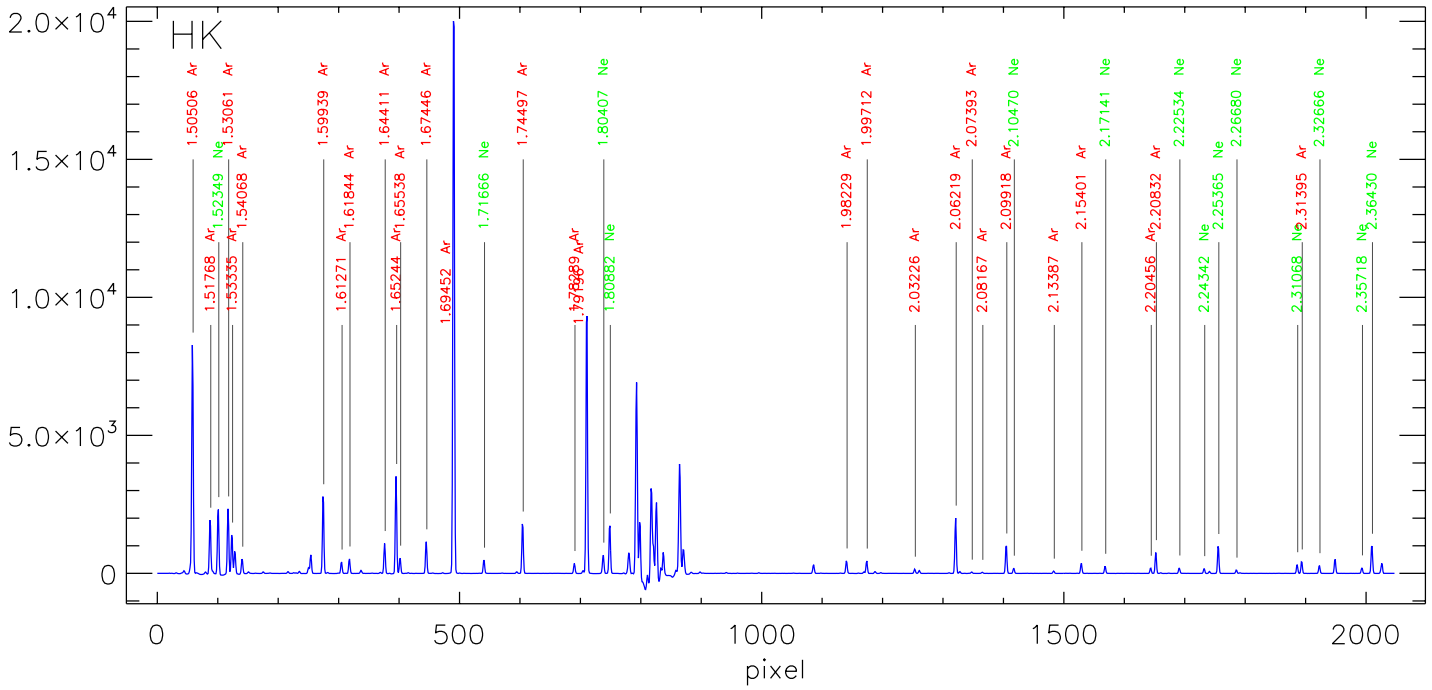

Figure 27 HK-band argon and neon emission lines

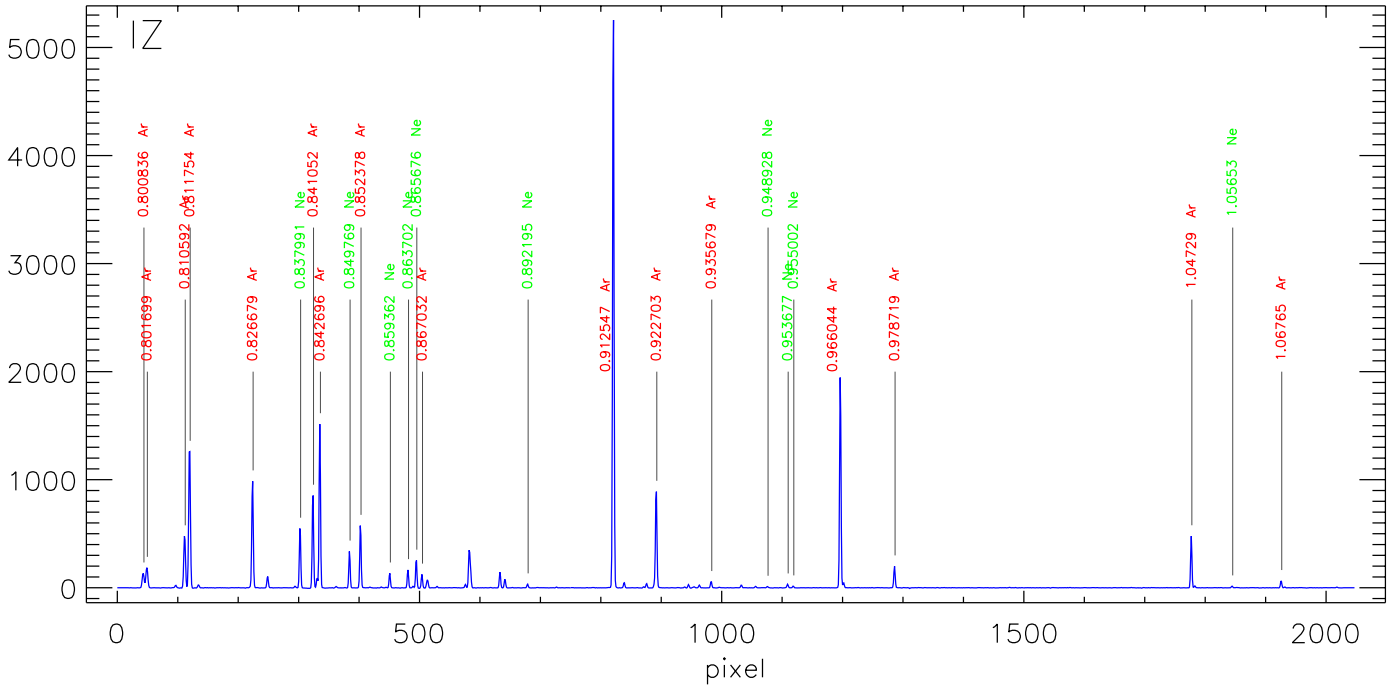

Figure 28 IZ-band argon and neon emission lines

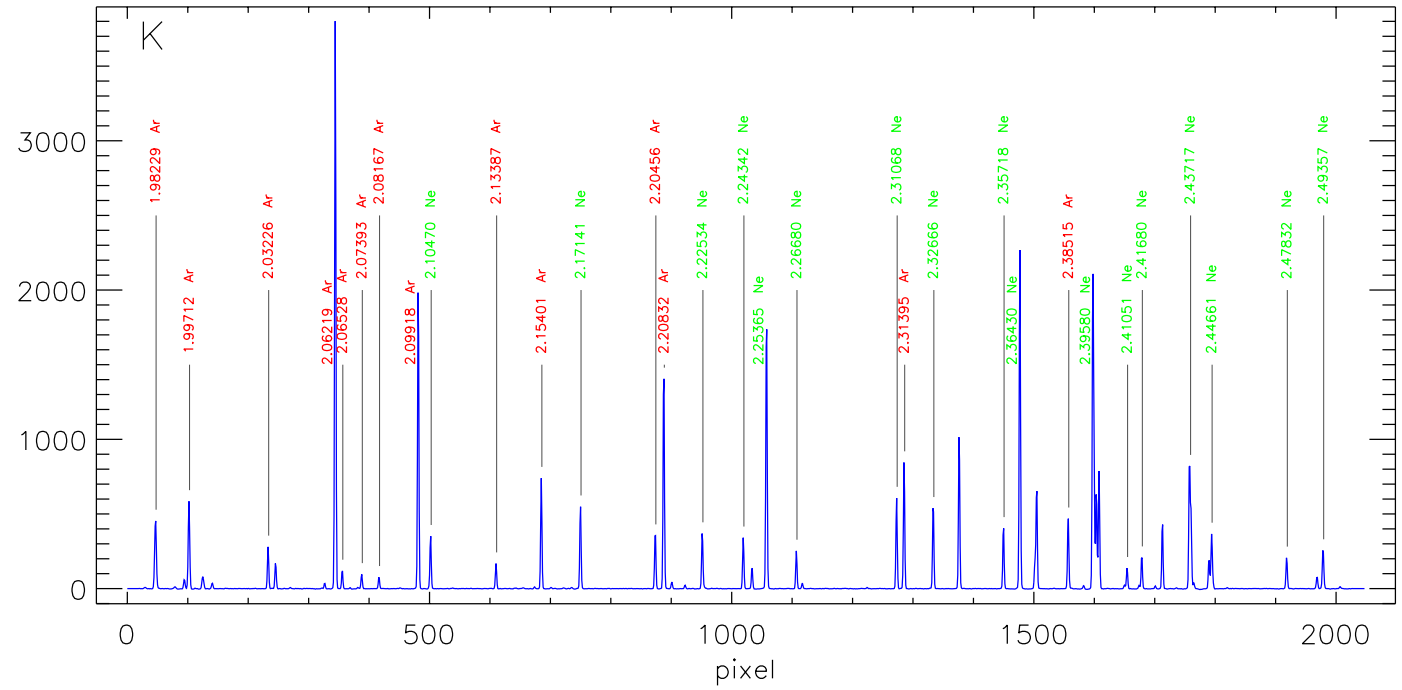

Figure 29 K-band argon and neon emission lines

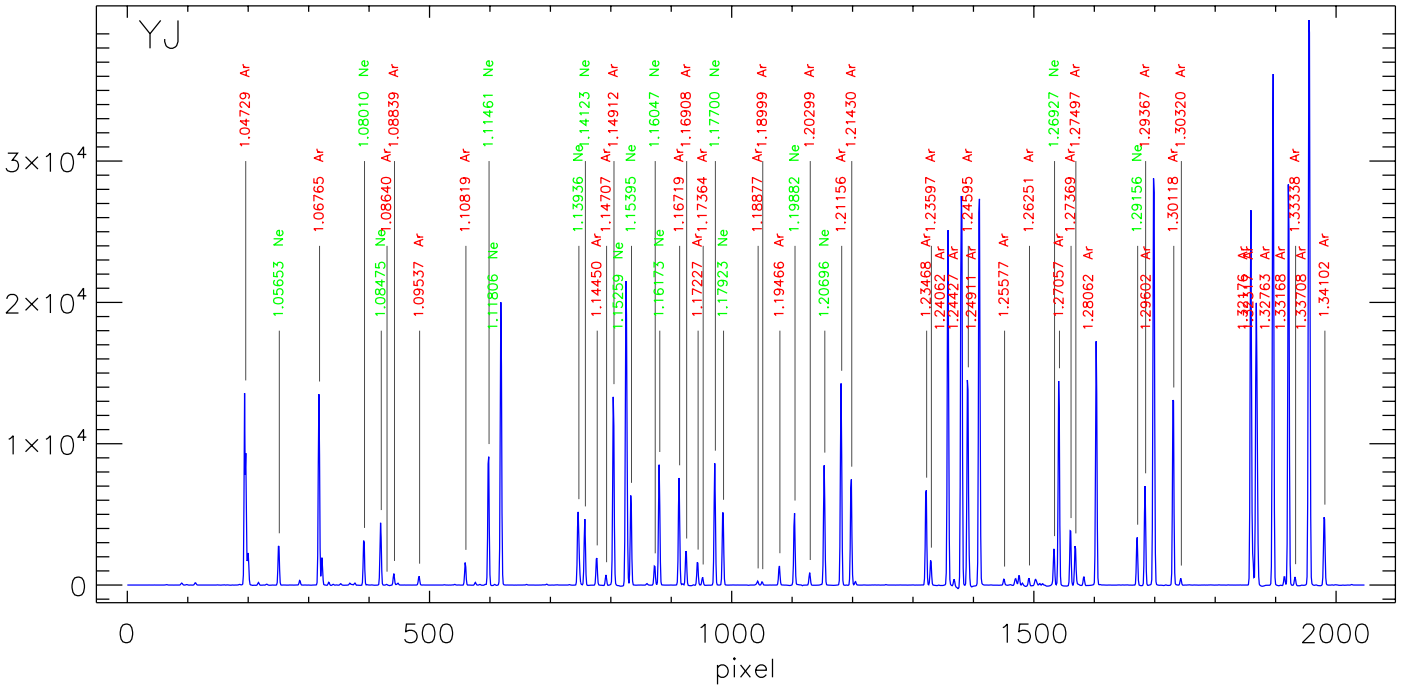

Figure 30 YJ-band argon and neon emission lines

7.1.3.2 Flow Chart

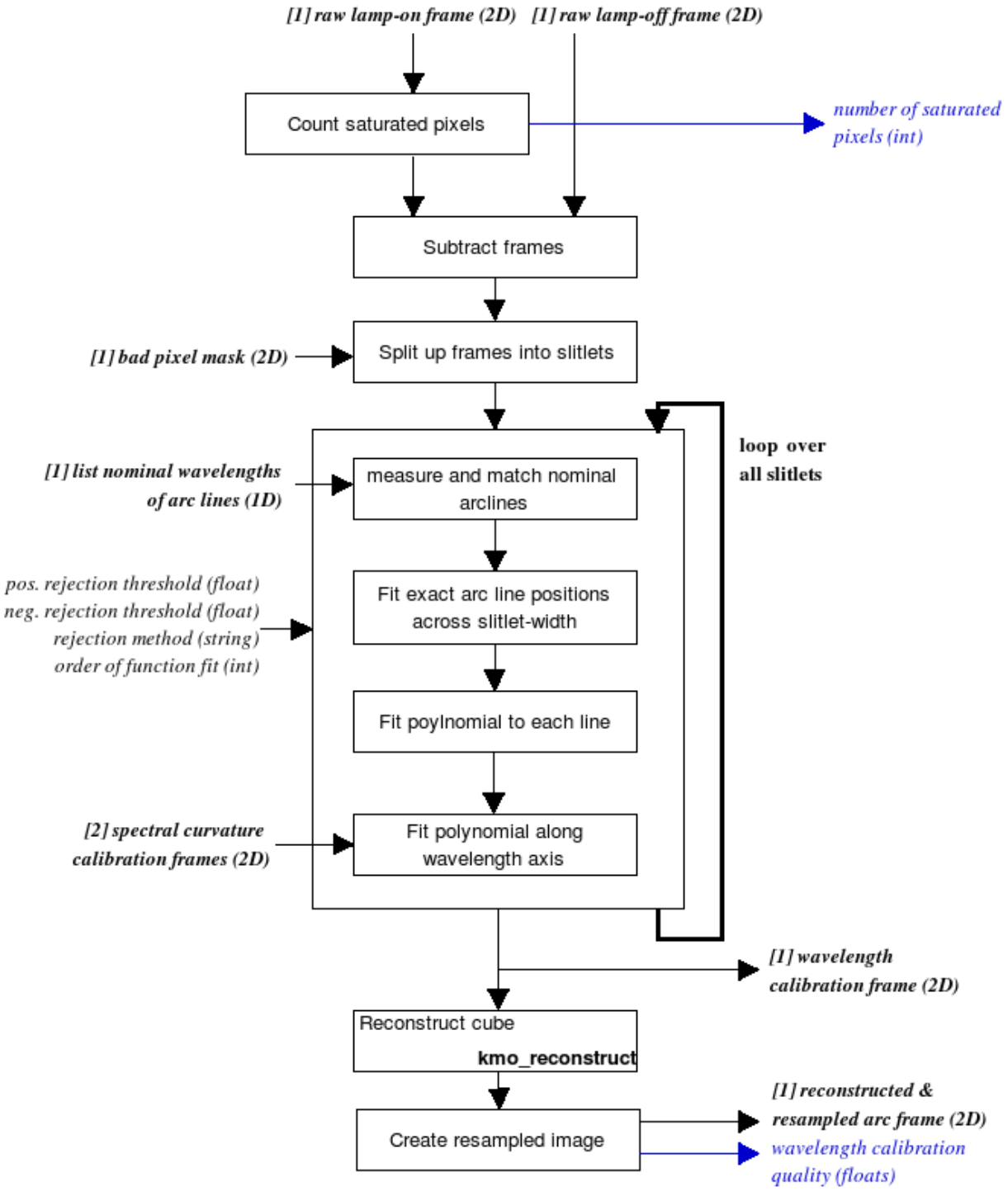

Figure 31**:** Flow chart of the recipe kmos_wave_cal

The processing steps are:

1. A raw lamp-on and a raw lamp-off frame taken with the internal arc lamp are subtracted.
2. The frame is split up into its slitlets (14 per IFU) using the flatfield badpixel mask. The following processing steps are applied to every slitlet. Bad pixels are ignored.
3. The λ positions of the arc lines will be measured and matched to a list of nominal arclines defined in a external file. This results in a first estimate where the line lie in the slitlet.
4. Then the exact positions of all lines across the slitlet width are fitted using a gauss fit.

5. A polynomial is fit to each line across the slitlet in order to extrapolate inexistent values resulting from rotation of the slitlets.
6. A polynomial is fitted along the wavelength direction to get the wavelength calibration data. The product of these operations so far is the 2D wavelength calibration frame (LUT).
7. As last step the provided arc frame will be reconstructed as cube and be decomposed into its slitlets which are saved into a frame with one slitlet beside the other. This way the quality of the wavelength calibration file can be determined quickly svisually.

All fits will be iterated twice, rejecting pixels which deviate by more than a few standard deviations.
The quality of the wavelength calibration is assessed and recorded in several QC1 parameters.

7.1.3.3 Input Frames

KMOS type	DO category	Amount	Comments
RAW	ARC_ON	≥1	Arclamp-on exposure, exactly one for every rotator angle
RAW	ARC_OFF	1	Arclamp-off exposure
F2D	XCAL	1	Calibration frame 1
F2D	YCAL	1	Calibration frame 2
F1L	ARC_LIST	1	List of reference arc lines, either for Argon or Neon or both combined. The first column has to contain the wavelengths and the second one the intensities
F2L	FLAT_EDGE	1	Frame containing the fitted edges of all IFUs.
F2L	REF_LINES	1	Reference line table
F2L	WAVE_BAND	1	Table with start-/end-values of wavelengthrange

7.1.3.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	
EXPTIME	double	any	all frames

7.1.3.5 Configuration Parameters

Basic parameters:

Name	Type	valid values	Default	Comments
order	int	order ≥ 0	0	The polynomial order to use for the fit of the wavelength solution. 0: (default) The appropriate order is choosen automatically depending on the waveband. Otherwise an order of 6 is recommended, except for IZ-band, there

				order 4 should be used.

Options for pipeline developers only:

Name	Type	valid values	Default	Comments
disp	double	disp > 0.0	-1.0	The expected spectral dispersion. By default the correct value is gained via the header keywords regarding filter configuration. This parameter is for testing the recipe with simulated data only.
flip	bool	TRUE, FALSE	TRUE	For some test data sets the wavelength is ascending from bottom to top, so this parameter has to be set to FALSE

Advanced parameters

Name	Type	valid values	Default	Comments
b_samples	int	b_samples > 2	2048	Nr. of samples of reconstructed data for the wavelength
b_start *b_end*	double	b_start > 0.0 b_end > b_start	-1.0	Start and end wavelength. The defaults of -1.0 instruct to use the internally defined range (see Section 6.3)
suppress_extension	bool	TRUE, FALSE	FALSE	

7.1.3.6 Output Frames

KMOS type	DO Category	Comments
F2D	LCAL	Calibration frame 3 (spectral dimension)
F2D	DET_IMG_WAVE	Resampled image of the reconstructed arc frame. All slitlets of all IFUs are aligned one next to the other.

Additional Output

All recipes doing reconstruction of cubes create a LUT which by default is saved to disk. For further information see Sec. 6.4.

7.1.3.7 Examples

```
$ esorex kmos_wave_cal arc.sof
                              with arc.sof containing:
                              arc_on.fits                ARC_ON
                              arc_off.fits               ARC_OFF
                              xcal_HHH.fits              XCAL
                              ycal_HHH.fits              YCAL
                              flat_edge_HHH.fits         FLAT_EDGE
                              kmos_ar_ne_list_h.fits     ARC_LIST
                              kmos_wave_ref_table.fits   REF_LINES
                              kmos_wave_band.fits        WAVE_BAND
```

7.1.4 kmos_illumination: Illumination Correction

Creates a calibration file to correct spatial non-uniformity of flatfield.

7.1.4.1 Description

This recipe creates the spatial non-uniformity calibration frame needed for all 24 IFUs. It can reduce sky flats (FLAT_SKY) or lamp flatfields (FLAT_ON) data. As calibration, it needs a master dark (only for FLAT_SKY), a master flat (only for FLAT_SKY) and the spatial and spectral calibration frames are required. The created product, the illumination correction, can be used as input for kmos_std_star and kmos_sci_red.

Basic parameters:

`--imethod`
The interpolation method used for reconstruction.

`--used_flat_type`
Type of input RAW files the recipe needs to use in case both are available in the input sof (sky/lamp)

`--range`
The spectral range [um] to combine when collapsing the reconstructed cubes.

Advanced parameters:
`--flux`
Specify if flux conservation should be applied.

`--add_all`
By default the first FLAT_SKY frame is omitted, since in the KMOS_spec_cal_skyflat template this is an acquisition frame to estimate the needed exposure time for the subsequent FLAT_SKY frames. If anyway all frames should be considered, set this parameter to TRUE.

`--neighborhoodRange`
Defines the range to search for neighbors during reconstruction

`--b_samples`
The number of samples in spectral direction for the reconstructed cube. Ideally this number should be greater than 2048, the detector size.

`--b_start`
`--b_end`
Used to define manually the start and end wavelength for the reconstructed cube. By default the internally defined values are used (see Section 6.3).

`--cmethod`
Following methods of frame combination are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:
  ```
  val > mean + stdev * cpos_rej
  ```
  and
  ```
  val < mean - stdev * cneg_rej
  ```

where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2).

- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.
- **sum**
  At each pixel position the sum is calculated.
- **min_max**
  The specified number of minimum and maximum pixel values will be rejected. `--cmax` and `--cmin` apply to this method.

`--cpos_rej`
`--cneg_rej`
`--citer`
see `--cmethod` = "ksigma"

`--cmax`
`--cmin`
see `--cmethod` = "min_max"

`--pix_scale`
Change the pixel scale [arcsec]. Default of 0.2\" results into cubes of 14x14pix, a scale of 0.1 results into cubes of 28x28pix, etc.

`--suppress_extension`
If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0.

7.1.4.2 Flow Chart

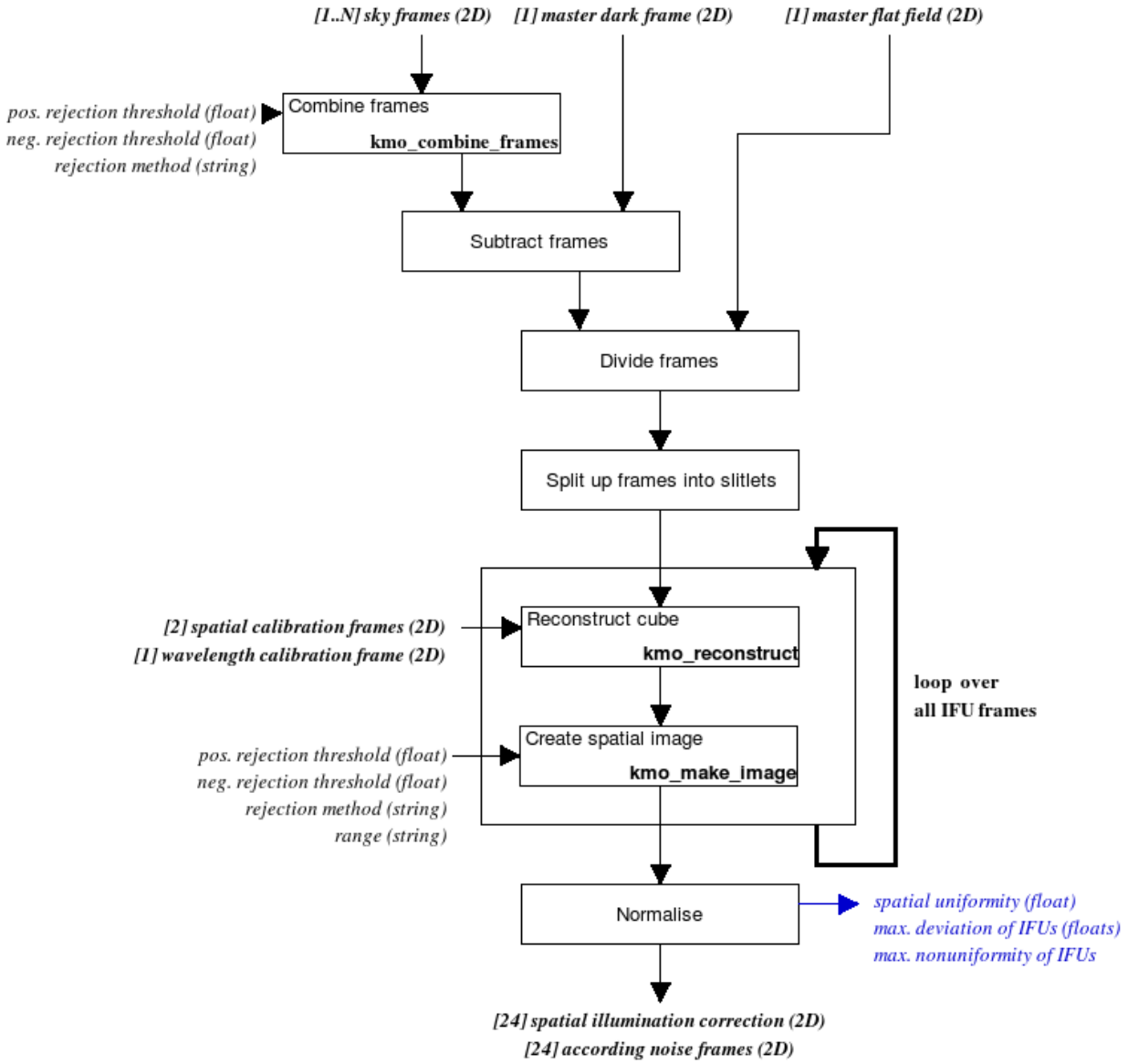

Figure 32**:** Flow chart of the recipe kmos_illumination

The processing steps are:

1. The RAW frames are separated by rotation angle. The reduction occurs separately for all angles. The product will contain as many extensions as there are angles.
2. The RAW frames are averaged using pixel rejection with a large sigma for clipping
3. An appropriate dark frame will be subtracted. The result is divided by the master flat field (only for FLAT_SKY inputs)
4. The frame is split up into frames referring to single IFUs.
5. Now the cubes are reconstructed (one for each IFU) using a bad pixel mask (from kmos_flat), a spectral curvature calibration frame (from kmos_flat) and a wavelength calibration frame (from kmos_wave_cal) and subsequently collapsed to spatial images.
6. The images will be normalized as a group. (i.e. so that the mean of all IFUs on the same detector is unity).

Furthermore several QC1 parameters are calculated, see section 5.1.4 for details.

7.1.4.3 Input Frames

KMOS type	DO category	Amount	Comments
F2D	FLAT_SKY	≥ 1	Flat sky exposure
F2D	FLAT_ON	≥ 1	Flat lamp exposure
F2D	MASTER_DARK	1 or 0	Master dark frame
F2D	MASTER_FLAT	1 or 0	Master flat frame
F2D	XCAL	1	Spatial calibration file
F2D	YCAL	1	Spatial calibration file
F2D	LCAL	1	Spectralcalibration file
F2L	WAVE_BAND	1	Table with start-/end-values of wavelengthrange
F2L	FLAT_EDGE	0 or 1	Table with the fitted slitlet edges from kmos_flat. MASTER_FLAT will be shifted to match FLAT_SKY frames.

7.1.4.4 Fits Header Keywords

Primary Header

None

Sub Headers

None

7.1.4.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
imethod	string	"NN" "lwNN" "swNN" "MS", "CS"	"CS"	Interpolation method for reconstruction: NN: Nearest Neighbor lwNN: linear weighted NN swNN: square weighted NN MS: Modified Shepard's method CS: Cubic spline *(optional)*
used_flat_type	string	"sky" "lamp"	"sky"	Only in case both RAW files types FLAT_SKY and FLAT_ON are passed to the recipe, this parameter is used to choose which type of RAW file has to be reduced.
range	string	"x1_start,x1_end; x2_start,x2_end"	""	The spectral ranges in microns to combine when collapsing the reconstructed cubes spectrally

Advanced parameters

Name	Type	valid values	Default	Comments

flux	bool	TRUE, FALSE	FALSE	Apply flux conservation
add_all	bool	TRUE, FALSE	FALSE	Considering 1st FLAT_SKY or not
neighborhoodRange	double	≥ 1	1.001	Defines the range to search for neighbors during reconstruction
b_samples	int	b_samples > 2	2048	Nr. of samples of reconstructed data for the wavelength
b_start *b_end*	double	b_start > 0.0 b_end > b_start	-1.0	Start and end wavelength. The defaults of -1.0 instruct to use the internally defined range (see Section 6.3)
cmethod	string	"ksigma" "min_max" "average" "median" "sum"	"ksigma"	The averaging method to apply *(optional)*
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = "ksigma")*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only when --cmethod = "ksigma")*
cmax cmin	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip with min/max-clipping *(optional, applies only when --cmethod = "min_max")*
pix_scale	double	TRUE, FALSE	0.2	The pixel scale: 0.2 arcsec results in cubes of 14x14 pixels. 0.1 arcsec result in cubes of 28x28 pixels
suppress_extension	bool	TRUE, FALSE	FALSE	

7.1.4.6 Output Frames

KMOS type	DO Category	Comments
F2I	ILLUM_CORR	The spatial non-uniformity calibration frame
F2L	SKYFLAT_EDGE	The parameters of the fitted edges of all IFUs of all detectors. From the FLAT_SKY frames

Additional Output

All recipes doing reconstruction of cubes create a LUT which by default is saved to disk. For further information see Sec. 6.4.

7.1.4.7 Examples

```
$ esorex kmos_illumination illum_sky.sof
                                  with illum.sof containing:
                                  sky1.fits              FLAT_SKY
                                  sky2.fits              FLAT_SKY
                                  sky3.fits              FLAT_SKY
                                  master_dark.fits       MASTER_DARK
                                  master_flat_HHH.fits   MASTER_FLAT
                                  xcal_HHH.fits          XCAL
                                  ycal_HHH.fits          YCAL
                                  lcal_HHH.fits          LCAL
                                  kmos_wave_band.fits    WAVE_BAND
                                  flat_edge_HHH.fits     FLAT_EDGE

or

$ esorex kmos_illumination illum_lamp.sof
                                  with illum.sof containing:
                                  lamp1.fits             FLAT_ON
                                  lamp2.fits             FLAT_ON
                                  lamp3.fits             FLAT_ON
                                  xcal_HHH.fits          XCAL
                                  ycal_HHH.fits          YCAL
                                  lcal_HHH.fits          LCAL
                                  kmos_wave_band.fits    WAVE_BAND
                                  flat_edge_HHH.fits     FLAT_EDGE
```

7.1.5 kmos_std_star: Telluric Standard Star

Creates a spectrum for telluric correction and derives zeropoint for flux calibration. In addition, this will estimate the spatial resolution (PSF).

7.1.5.1 Description

This recipe creates a telluric calibration frame and a PSF frame. It accepts an optional illumination correction frame as input created with the kmos_illumination-recipe.
Since there won't be enough standard stars to observe for all IFUs in one exposure, one has to do several exposures in a way that there is at least one standard star and one sky exposure in each IFU. A internal data organiser will analyse the provided exposures and select the appropriate frames as follows:

1. For each IFU the first standard star in the list of provided exposures is taken. All subsequent standard star exposures for this IFU will be ignored
2. A corresponding sky exposure will be chosen which will be as close in time to the standard star exposure as possible.
3. For any IFUs not containing a standard star and a sky exposure an empty frame will be returned.

NOISE_SPEC contains in any case the shot noise [sqrt(counts*gain)/gain]. If the exposures have been taken with template KMOS_spec_cal_stdstarscipatt, then an additional noise component is added in: All existing sky exposures for an IFU are subtracted pairwise, spectra are extracted and the std deviation is calculated.

Basic parameters:

`--startype`

If this parameter is specified, the stored star types of the observed obejcts in the FITS headers are overridden. This value applies to all objects examined in the input frames. Examples would be "A3I", "G3IV" or "K0I". The first letter defines the star type, the second letter the spectral class and the last letters the luminosity class.

`--magnitude`

If this parameter is specified, the stored magnitudes in the FITS headers are overridden. For HK two magnitudes for each H and K have to be specified. All other gratings just use a single magnitude. If two values are provided, they have to be separated with a comma.

`--fmethod`

The type of function that should be fitted spatially to the collapsed image. This fit is used to create a mask to extract the spectrum of the object. Valid values are "gauss" and "moffat".

`--imethod`

The interpolation method used for reconstruction. As default 'CS' is selected. Note that no error spectra will be generated for this interpolation method. Select a nearest neighbour method otherwise.

`--range`

The spectral range [um] to combine when collapsing the reconstructed cubes.

`--save_cubes`

Set this parameter to TRUE in order to save the reconstructed cubes.

`--no_noise`

Applies only for data taken with template KMOS_spec_cal_stdstarscipatt:
Skip lengthy calculation of noise-spectra on all sky exposures (no NOISE_SPEC will be produced).

Advanced parameters:

`--flux`
Specify if flux conservation should be applied.

`--neighborhoodRange`
Defines the range to search for neighbors during reconstruction

`--b_samples`
The number of samples in spectral direction for the reconstructed cube. Ideally this number should be greater than 2048, the detector size.

`--b_start`
`--b_end`
Used to define manually the start and end wavelength for the reconstructed cube. By default the internally defined values are used (see Section 6.3).

`--cmethod`
Following methods of frame combination are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:
  ```
  val > mean + stdev * cpos_rej
  ```
  and
  ```
  val < mean - stdev * cneg_rej
  ```
  where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2).
- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.
- **sum**
  At each pixel position the sum is calculated.
- **min_max**
  The specified number of minimum and maximum pixel values will be rejected.
  `--cmax` and `--cmin` apply to this method.

`--cpos_rej`
`--cneg_rej`
`--citer`
see `--cmethod` = "ksigma"

`--cmax`
`--cmin`
see `--cmethod` = "min_max"

`--xcal_interpolation`

If TRUE interpolate the pixel position in the slitlet (xcal) using the two closest rotator angles in the calibration file. Otherwise take the values of the closest rotator angle.

`--suppress_extension`
If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0

7.1.5.2 Flow Chart

The flowchart for this recipe is split up in two diagrams. To simplify the flowchart the internal data organising workflow isn't depicted. All steps apply to each active IFU individually. The resulting PSF frames, telluric & error spectra of all processed IFUs are merged into the defined output frames.

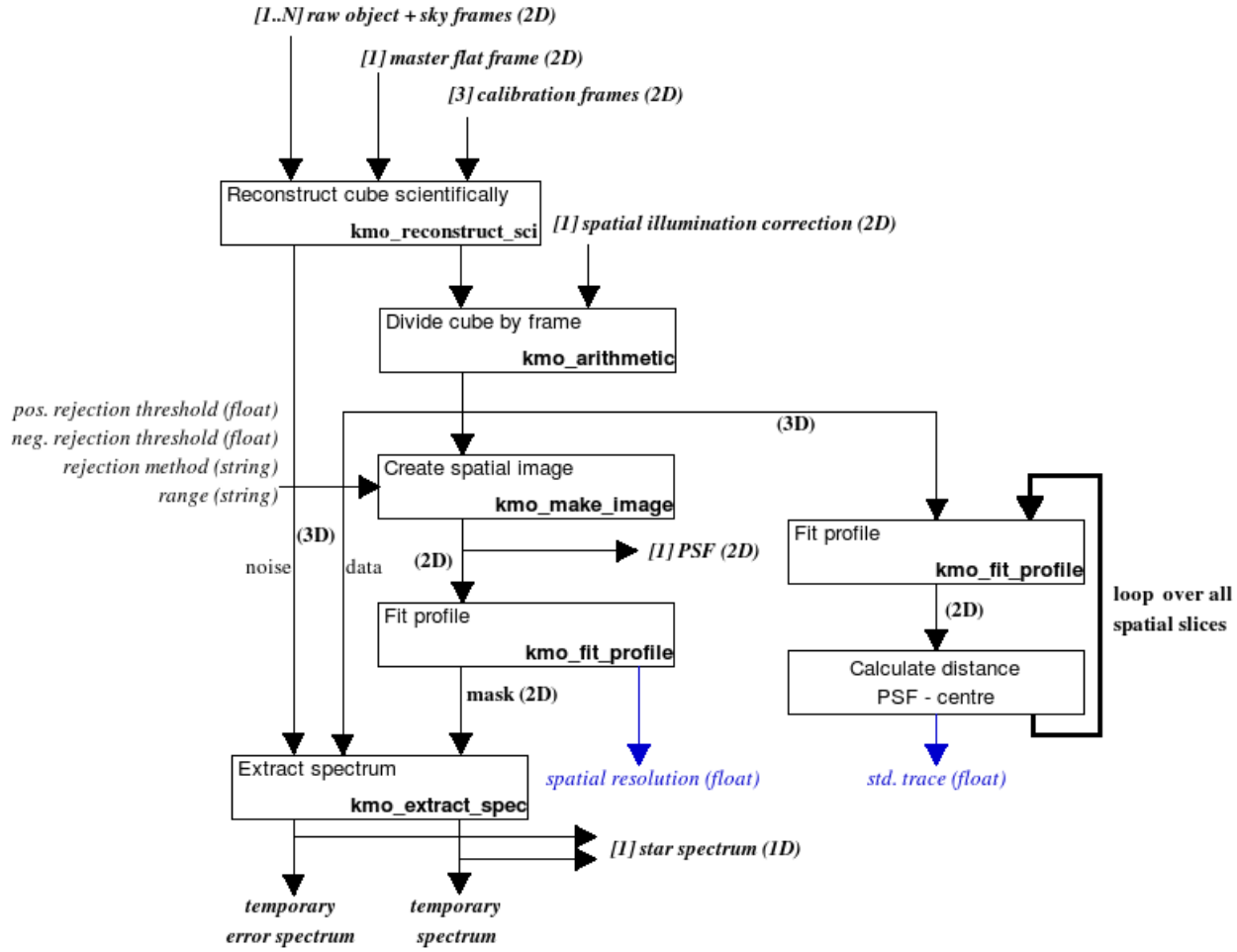

Figure 33**:** Flow chart of the recipe kmos_std_star (Part 1)

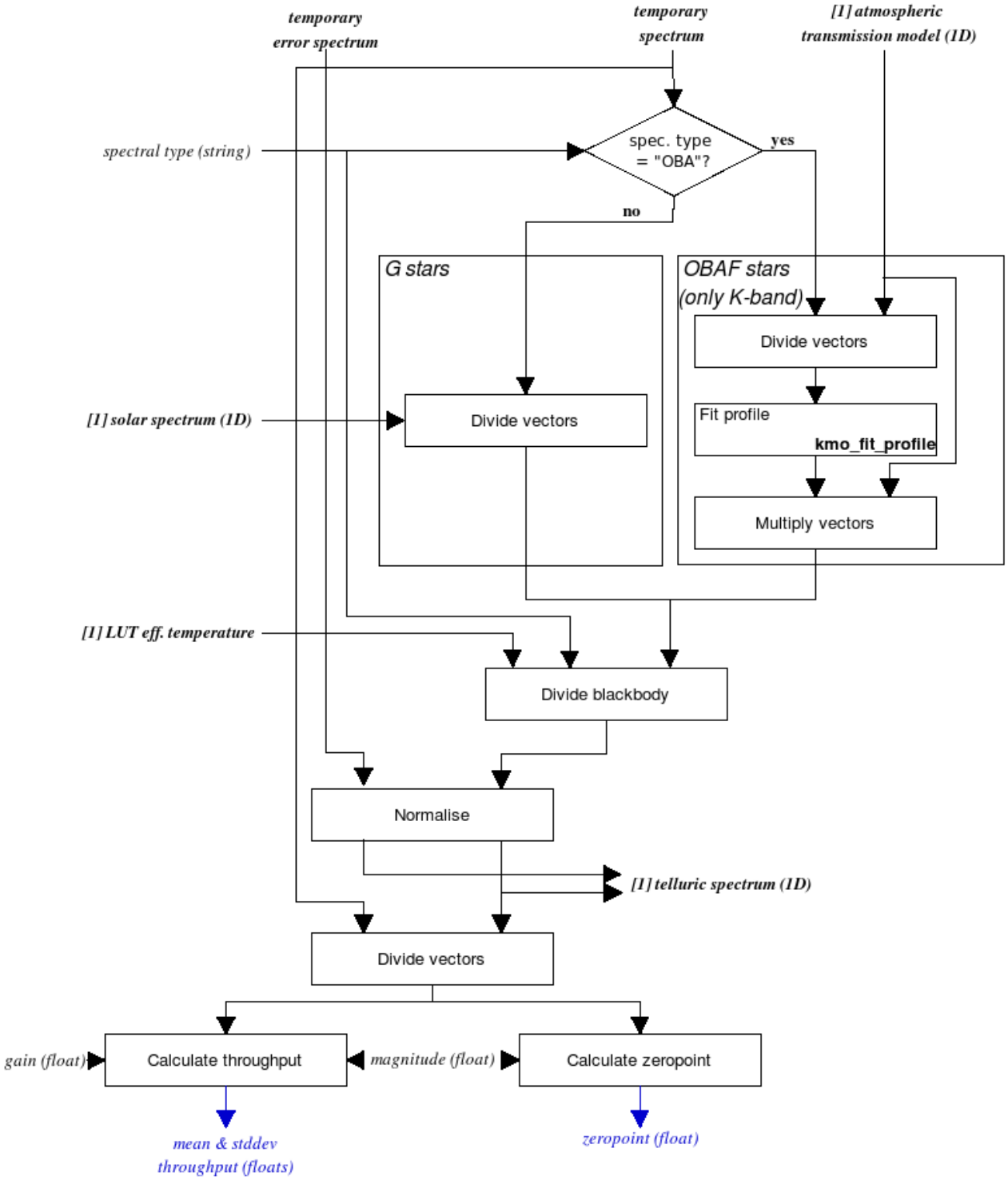

Figure 34**:** Flow chart of the recipe kmos_std_star (Part 2)

The processing steps are:

1. From one or more raw object and sky frames the IFUs containing observed standard stars are extracted.
2. The signal frame and the noise frame are reconstructed as cubes using a bad pixel mask (from kmos_flat), two spectral curvature calibration frames (from kmos_flat) and a wavelength calibration frame (from kmos_wave_cal). The corresponding IFU frames are also extracted from these auxiliary inputs.
3. The reconstructed cube is divided spatially by the spatial illumination correction frame.

4. To the data cube for each spatial slice a 2D-profile is fitted to obtain the position of the object. The RMS of these values is saved as header keyword QC STD TRACE.
5. The signal cube is collapsed to a spatial image. This results into an image of the PSF of the IFU.
6. From the signal and the noise cubes the signal and error spectra are extracted. As a mask, the profile fit of the PSF image is used. This intermediate spectrum is saved as STAR_SPEC.
7. Two cases are distinguished in the further processing in function of the spectral type of the standard star observed:
    a. OBAF stars
        I. The temporary signal spectrum is divided by a model atmospheric transmission.
        II. Fit a Lorentzian function to stellar absorption line(s) and subtract.
        III. Multiply the model atmospheric transmission back in.

       This applies only in K-band. For other bands a warning is emitted.
    b. G stars
        I. Convolve the solar spectrum to the correct spectral resolution and divide it out of the temporary signal spectrum.
8. Divide the result by a curve corresponding to the effective temperature of the star.
9. Normalising the spectrum (and also the error spectrum) yield the telluric correction and the final error spectrum.
10. By dividing the temporary spectrum by the telluric correction and by providing the magnitude of the star and the gain of the detector (in fits header) two QC1 parameters can be calculated: the zeropoint and the throughput (mean and standard deviation).

Above steps are repeated for all IFUs containing a standard star and a sky frame in the input data.

7.1.5.3 Input Frames

KMOS type	DO category	Amount	Comments
RAW	STD	≥ 1	Flat sky exposure
F2D	XCAL	1	Spatial calibration file
F2D	YCAL	1	Spatial calibration file
F2D	LCAL	1	Spectralcalibration file
F2D	MASTER_FLAT	1	Master flat frame
F2L	WAVE_BAND	1	Table with start-/end-values of wavelengthrange
F2D	ILLUM_CORR	0,1	Illumination correction
F1S	SOLAR_SPEC	0,1	Solar spectrum (only for G stars)
F1S	ATMOS_MODEL	0,1	Atmospheric transmission model (only for OBAF stars in K-band)
F2L	SPEC_TYPE_LOOKUP	0,1	Look up table of effective stellar temperatures

7.1.5.4 Fits Header Keywords

Primary Header

None

Sub Headers

None

7.1.5.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
startype	string	Star type: O, B, A, F, G, K Spectral class: 1 to 9 (K: only 0) Luminosity class: I to V (e.G. "G4VI")	""	The spectral type of the star *(optional)*
magnitude	string	A single or two comma separated float values	""	The magnitude of the star *(optional)*
fmethod	string	"gauss" or "moffat"	"gauss"	The 2D function to fit to the collapsed cube *(optional)*
imethod	string	"NN" "lwNN" "swNN" "MS" "CS"	"CS"	Interpolation method for reconstruction: NN: Nearest Neighbor lwNN: linear weighted NN swNN: square weighted NN MS: Modified Shepard's method CS: Cubic spline *(optional)*
range	string	"x1_start,x1_end; x2_start,x2_end"	""	The spectral ranges in microns to combine when collapsing the reconstructed cubes spectrally *(optional)*
save_cubes	bool	TRUE, FALSE	FALSE	Save intermediate reconstructed cubes *(optional)*
no_noise	bool	TRUE, FALSE	FALSE	Skip noise-calculation on sky-frames *(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
flux	bool	TRUE, FALSE	FALSE	Apply flux conservation *(optional)*
neighborhoodRange	double	≥ 1	1.001	Defines the range to search for neighbors during reconstruction *(optional)*
b_samples	int	b_samples > 2	2048	Nr. of samples of reconstructed data for the wavelength

b_start *b_end*	double	b_start > 0.0 b_end > b_start	-1.0	Start and end wavelength. The defaults of -1.0 instruct to use the internally defined range (see Section 6.3)
cmethod	string	"ksigma" "min_max" "average" "median" "sum"	"ksigma"	The averaging method to apply *(optional)*
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = "ksigma")*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only when --cmethod = "ksigma")*
cmax cmin	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip with min/max-clipping *(optional, applies only when --cmethod = "min_max")*
xcal_interpolation	bool	TRUE, FALSE	TRUE	*(optional)*
suppress_extension	bool	TRUE, FALSE	FALSE	*(optional)*

7.1.5.6 Output Frames

KMOS type	DO Category	Comments
F1I	TELLURIC	The normalised telluric spectrum [ADU/DIT]
F1I	STAR_SPEC	The extracted star spectrum [ADU/DIT]
F2I	STD_IMAGE	The standard star PSF
F2I	STD_MASK	The generated mask used to extract the star spectrum
F1I	NOISE_SPEC	The shot noise: sqrt(counts*gain)/gain *Only for data taken with template KMOS_spec_cal_stdstarscipatt:* Add in noise estimate based on the sky exposures present in all exposures: Skies are subtracted pairwise and reconstructed for every IFU over all exposures. Then the spectra are extracted and for every wavelength point the stddev is calculated and saved)

Additional Output

All recipes doing reconstruction of cubes create a LUT which by default is saved to disk. For further information see Sec. 6.4.

7.1.5.7 Examples

```
$ esorex kmos_std_star std.sof
                              with std.sof containing:
                              obj1.fits                 STD
                              obj2.fits                 STD
                              obj13fits                 STD
                              sky1.fits                 STD
                              sky2.fits                 STD
                              xcal_HHH.fits             XCAL
                              ycal_HHH.fits             YCAL
                              lcal_HHH.fits             LCAL
                              master_flat_HHH.fits      MASTER_FLAT
                              illum_corr_HHH.fits       ILLUM_CORR
                              kmos_wave_band.fits       WAVE_BAND
                              kmos_solar_h_2400.fits    SOLAR_SPEC
                              kmos_atmos_k.fits         ATMOS_MODEL
                              kmos_spec_type.fits       SPEC_TYPE_LOOKUP
```

7.1.6 kmos_gen_telluric: Generate a telluric frame

7.1.6.1 Description

This recipe creates the TELLURIC_GEN frame needed by kmos_sci_red by merging the TELLURIC (produced by kmos_std_star), the static RESPONSE frame and the TELLURIC_CORR (provided by the user). The way the frames are combined is controlled by the –method parameter.

Parameters:

```
--method
How the TELLURIC_GEN is generated [Default: 0]
```

The way the frames are combined is controlled by this parameter:

1. The recipe output accounts for telluric absorption, instrument response, and flux calibration as included in the TELLURIC file produced by kmos_std_star. If TELLURIC is missing, the recipe output accounts only for instrument response and flux calibration as provided in the RESPONSE calibration file.
2. The user-providad file TELLURIC_CORR will be used as output. It is responsibility of the user to provide a TELLURIC_CORR that accounts for the desired calibrations (instrument response, telluric features, and flux calibration).
3. The recipe outputs accounts only for instrument response and flux calibration. The instrument response is taken from the RESPONSE calibration file. The zeropoint is taken from the TELLURIC file produced by kmos_std_star. If TELLURIC is not present, average zeropoints stored in the RESPONSE calibration file are used. Use this option if the telluric correction is not needed or if it is done independently via external tools.
4. The recipe outputs accounts telluric absorption, instrument response, and flux calibration. It is generated by multiplying the user-input TELLURIC_CORR by the RESPONSE calibration file. The zeropoint are taken from the TELLURIC_CORR. It is respoinsibility of the user to provide a TELLURIC_CORR that accounts for telluric absorptions and that contains the correct zeropoint.
5. The recipe output accounts for telluric absorption and flux calibration (taken from the TELLURIC file produced by kmos_std_star), and for instrument response (taken from the RESPONSE calibration file) (PIPE-7945).

7.1.6.2 Flow Chart

The processing steps are:

1. Take the correct way and check the inputs depending of the parameter --method.
2. Load DATA extensions and apply the method for generation the output telluric correction.

7.1.6.3 Input Frames

KMOS type	DO category	Amount	Comments
F1I	TELLURIC	0,1	Produced by kmos_std_star
F1I	TELLURIC_CORR	0,1	Provided by the user
F3I	RESPONSE	0,1	Static calibration

It is mandatory provide, at least, input.

7.1.6.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
ESO INS GRAT1 ID	string	"IZ"/"YJ"/"H"/"K"/"HK"	Grating used in the file
ESO OCS ROT NAANGLE	float	[-360:360]	Rotation angle

Sub Headers

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.DATA" / "IFU.XX.NOISE"	Name of extension
ESO QC ZPOINT	double	If data → value	Zpoint

7.1.6.5 Configuration Parameters

Name	Type	valid values	Default	Comments
method	int	0,1,2,3	"0"	How the TELLURIC is generated

7.1.6.6 Output Frames

KMOS type	DO Category	Comments
F1I	TELLURIC_GEN	Telluric correction, used by kmos_sci_red

7.1.6.7 Examples

```
$ esorex kmos_gen_telluric --method=3 gen_telluric.sof
                           with std.sof containing:
                           RESPONSE.fits             RESPONSE
                           TELLURIC_CORR.fits        TELLURIC_CORR
```

7.1.7 kmos_molecfit_model: Compute an atmospheric model

It runs molecfit on KMOS standard star file to compute an atmospheric model.

7.1.7.1 Description

This recipe compute an atmospheric model using as entry a 1D spectrum in KMOS format STAR_SPEC. Alternatively, it can accepts an optional input KMOS format 1D EXTRAC_SPEC or a 3D CUBE (SCIENCE or SCI_RECONSTRUCTED) from which it is extracted a 1D spectrum.
The type STAR_SPEC contains 48 extensions (24 data and 24 noise) and the others can be contain 24 or 48 extensions. The noise extensions are ignored.

Parameters:

`--process_ifus`
A list IFUs to process. If set to -1, all the IFUs that have data will be process. [Default value: "-1"]

`--wave_range`
A list of numbers defining the wavelength ranges to fit in the grating. If set to -1, grating dependent default values are used. [Default value: "-1"]

The grating dependent values of –wave_rage are:

IZ: '0.815,0.830,0.894,0.899,0.914,0.919,0.929,0.940,0.972,0.986'

YJ: '1.106,1.116,1.075,1.083,1.131,1.137,1.139,1.149,1.155,1.166,1.177,1.189,1.201,1.209, 1.263,1.276,1.294,1.303,1.312,1.336'

H: '1.482,1.491,1.500,1.512,1.559,1.566,1.598,1.605,1.575,1.583,1.622,1.629,1.646,1.671, 1.699, 1.711,1.721,1.727,1.746,1.758,1.764,1.767,1.773,1.780,1.789,1.794'

K: '1.975,1.987,1.993,2.010,2.041,2.060,2.269,2.291,2.308,2.335,2.360,2.379,2.416,2.440, 2.445,2.475'

HK: '1.575,1.584,1.594,1.606,1.646,1.671,1.756,1.771,1.781,1.811,1.945,1.969,1.975,1.987, 1.993,2.030,2.043,2.089,2.242,2.294,2.308,2.335,2.360,2.379'

`--list_molec`
A list of molecules to fit. If set to -1, grating dependent default values are used. [Default value: "-1"]

`--fit_molec`
Flags to fit the column density of the corresponding list_molec in grating. If set to -1, grating dependent default values are used. [Default value: "-1"]

`--relcol`
Column density relative to atmospheric profile of the corresponding list_molec in grating. If set to -1, grating dependent default values are used. [Default value: "-1"]

The default grating dependent values for --list_molec, --fit_molec and --relcol are defined as follows. The number of entries of each parameter has to be the same:

IZ:

- list_molec: 'H2O'
- fit_molec: '1'
- relcol: '1.'

YJ:

- list_molec: 'H2O,CO2,CH4,O2'
- fit_molec: '1,0,0,1'
- relcol: '1.,1.06,1.,1.'

H:

- list_molec: 'H2O,CO2,CO,CH4'
- fit_molec: '1,1,0,1'
- relcol: = '1.,1.06,1.,1.'

K:

- list_molec: 'H2O,CO2,CH4'
- fit_molec: '1,1,1'
- relcol: '1.,1.06,1.'

HK:

- list_molec: 'H2O,CO2,CH4'
- fit_molec: '1,1,1'
- relcol: '1.,1.06,1.'

`--ftol`
Relative chi-square convergence criterion. [Default value: 0.01]

`--xtol`
Relative parameter convergence criterion. [Default value: 0.001]

`--fit_cont`
Flag to enable/disable the polynomial fit of the continuum. [Default value “TRUE”]

`--cont_n`
Degree of polynomial continuum fit. [Default value: 1]

`--fit_wlc`
Flag to enable/disable the refinement of the wavelength solution. [Default value: “TRUE”]

`--wlc_n`
Polynomial degree of the refinen wavelength solution. [Default value: 2]

`--wlc_const`
Initial constant term for wavelength adjustment (shift relative to half wavelength range. [Default value: 0.0]

`--use_input_kernel`
The parameters below are ignored if use_input_kernel. [Default value: “TRUE”]

This is used to call Molecfit and specified if need to fit with the inside kernels (FALSE) or to use one the library provided via the .sof file (TRUE).

`--fit_res_box`
Fit resolution by Boxcar LSF. [Default value: “FALSE”]

`--relres_box`
Initial value for FWHM of Boxcar rel. to slit width (Range between 0 and 2). [Default value: 0.0]

`--fit_res_gauss`
Fit resolution by Gaussian. [Default value: “TRUE”]

`--res_gauss`
Initial value for FWHM of the Gaussian in pixels. If set to -1, grating dependent default values are used. [Default value: “-1”]

The default values are the next: IZ=1.84, YJ=1.82, H=1.76, K=1.73, HK=2.06 .

`--fit_res_lorentz`
Fit resolution by Lorentz. [Default value: “FALSE”]

`--res_lorentz`
Initial value for FWHM of the Lorentz in pixels. [Default value: 0.5]

`--kernmode`
Voigt profile approximate or independent Gauss and Lorentz. [Default value: “FALSE”]

`--kernfac`
Size of Gaussian/Lorentz/Voigt kernel in FWHM. [Default value: 5.0]

`--varkern`
Does the kernel size increase linearly with wavelength?. [Default value: “FALSE”]

`--suppress_extension`
If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0

7.1.7.2 Flow Chart

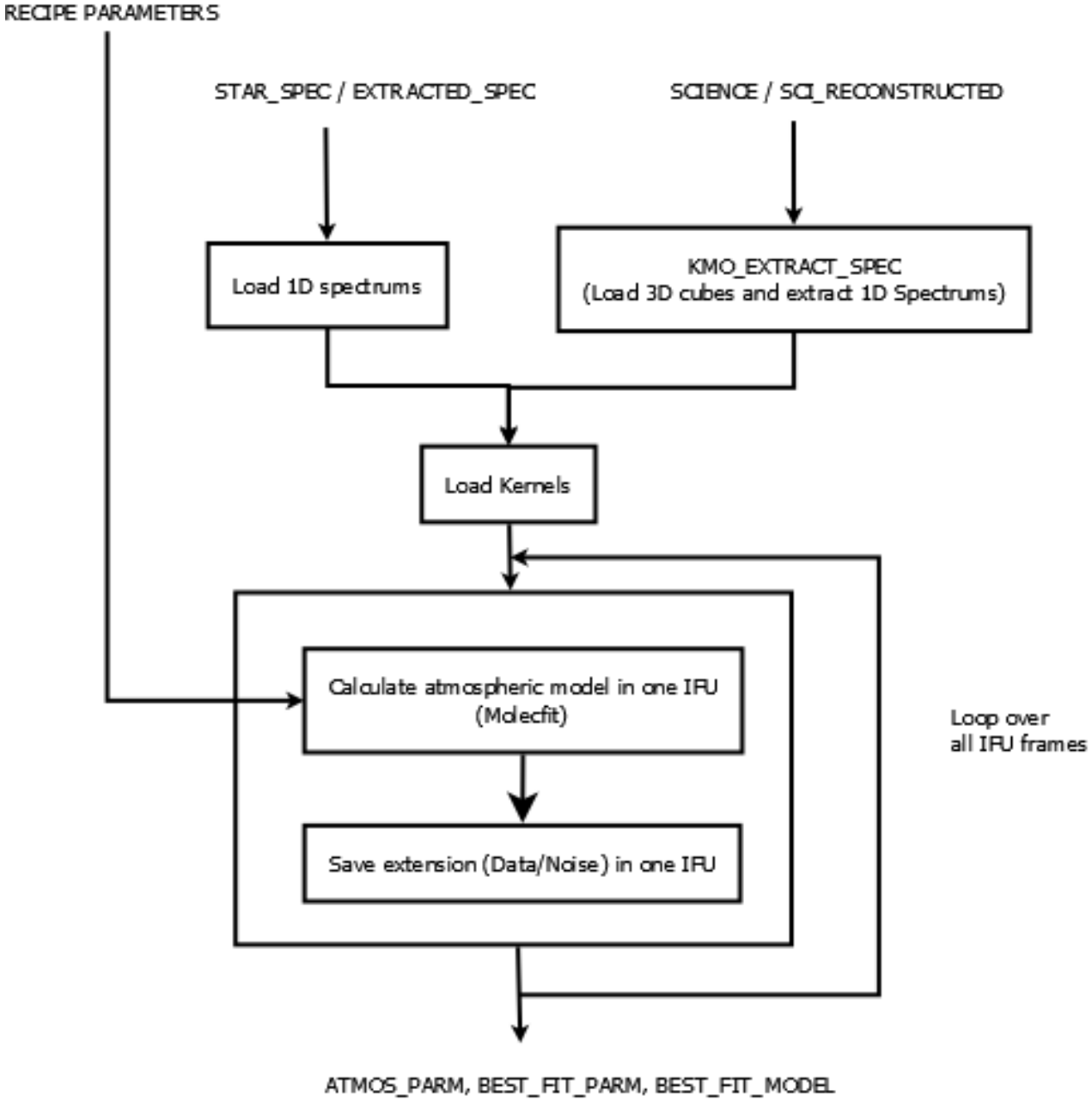

Figure 35**:** Flow chart of the recipe kmos_molecfit_model

The processing steps are:

1. The IFUs containing data are extracted from the input raw object and sky frames.
2. If kernel library is provided, the recipe selects the instrumental line spread function that matches. The IFU and that is close in rotator angle.
3. The atmospheric model is calculated in each IFU with data through Molecfit
4. The obtained results of Molecfit are storage in the output files.

7.1.7.3 Input Frames

KMOS type	DO category	Amount	Comments
F1I	STAR_SPEC	0,1	Spectrum 1D
F1I	EXTRACT_SPEC	0,1	Spectrum 1D
F3I	SCIENCE	0,1	Spectrum 1D (in 3D cubes)
F3I	SCI_RECONSTRUCTED	0,1	Spectrum 1D (in 3D cubes)
F2I	KERNEL_LIBRARY	0,1	Kernel library with grating dependency (equal to spec)

It is mandatory provide one input spectrum.

7.1.7.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
ESO INS GRAT1 ID	string	"IZ"/"YJ"/"H"/"K"/"HK"	Grating used in the file
ESO OCS ROT NAANGLE	float	[-360:360]	Rotation angle

Sub Headers

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.DATA" / "IFU.XX.NOISE"	Name of extension
CRVAL1	double	[0:2.5]	Initial wavelength (1D)
CDELT1	double	>0	Wavelength step (1D)
CRVAL3	double	[0:2.5]	Initial wavelength (3D)
CDELT3	double	>0	Wavelength step (3D)

Primary Headers: KERNEL_LIBRARY

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.ANGLE.YYY"	Name of extension
CRVAL1	double	[0:2.5]	Initial wavelength
CD2_2	double	>0	Wavelength step

7.1.7.5 Configuration Parameters

Name	Type	valid values	Default	Comments
process_ifus	string	Comma separated number of ifus	"-1"	A list of IFUS to process. If -1, all IFUs with data will be process
wave_range	string	"x1_start,x1_end; x2_start,x2_end"	"-1"	A list of wavelength ranges to fit. If -1, default grating dependent.
list_molec	string	Comma separated string with valid values	"-1"	A list of molecules to fit. If -1, default grating dependent
fit_molec	string	Comma separated int (bool) values	"-1"	Flags (same number of values that in list_molec) to fit the column density. If -1, default grating dependent
relcol	string	Comma separated double values	"-1"	Column density relative to atmospheric profile (same number of parameter that in list_molec). If -1, default grating dependent

ftol	double	>=1.e-10	0.01	Relative chi-square convergence criterion
xtol	double	>=1.e-10	0.001	Relative parameter convergence criterion
fit_cont	bool	TRUE, FALSE	FALSE	Flag to enable/disable the polynomial fit of the continuum
cont_n	int	>=0 && <= 8	1	Polynomial degree of the continuum fit.
fit_wlc	bool	TRUE, FALSE	TRUE	Flag to enable/disable the refinement of the wavelength.
wlc_n	int	>=1 && <= 8	2	Polynomial degree of the refined wavelength solutions
wlc_const	double	>= 0.0	0.0	Initial constant term for wavelength adjustement (shift relative to half wavelength range)
use_input_kernel	bool	TRUE, FALSE	TRUE	The parameters below are ignored if use_input_kernel
fit_res_box	bool	TRUE, FALSE	FALSE	Fit resolution by Boxar LSF.
relres_box	double	>=0 && <=2.0	0.0	Initial value for FWHM of Boxcar rel. to slit width.
fit_res_gauss	bool	TRUE, FALSE	TRUE	Fit resolution by Gaussian
res_gauss	double	>= 0.01	-1.0	Initial value for FWHM of the Gaussian in pixels. If -1.0, default grating dependent
fit_res_lorentz	bool	TRUE, FALSE	FALSE	Fit resolution by Lorentz.
res_lorentz	double	>=0.0 && <=100.	0.5	Initial value for FWHM of the Lorentz in pixels
kernmode	bool	TRUE, FALSE	FALSE	Voigt profile approximate or independent Gauss and Lorentz.
kernfac	double	>=3.0 && <= 30.	5.0	Size of Gaussian/Lorentz/Voigt kernel in FWHM.
varkern	bool	TRUE, FALSE	FALSE	Does the kernel size increase linearly with wavelength?

7.1.7.6 Output Frames

KMOS type	DO Category	Comments
F1L	ATMOS_PARM	Atmospheric parameters
F1L	BEST_FIT_PARM	Best fitting parameters
F1L	BEST_FIT_MODEL	Best fit model and intermediate products

7.1.7.7 Examples

```
$ esorex kmos_molecfit_model mf_model.sof
                             with std.sof containing:
                             STAR_SPEC_IZIZIZ.fits     STAR_SPEC
                             kernel_iz.fits            KERNEL_LIBRARY
```

7.1.8 kmos_molecfit_calctrans: Compute the telluric correction

Read the results from kmos_molecfit_model and computes the telluric correction for a scientific input data.

7.1.8.1 Description

This recipe uses the results from kmos_molecfit_model and runs calctrans to calculate the telluric correction for the scientific input data. The scientific input data can have category:

- STAR_SPEC (24 DATA plus 24 NOISE extensions).
- EXTRAC_SPEC (24 DATA extensions, additional 24 NOISE extensions are optional).
- SCIENCE (24 DATA extensions, additional 24 NOISE extensions are optional).
- SCI_RECONSTRUCTED (24 DATA extensions, additional 24 NOISE extensions are optional).

It is not mandatory that all the DATA extensions contains data.
The recipe accounts for the difference in airmass between the input model and the input scientific data, and for different spectral resolutions between the various KMOS IFU's.

Parameters:

`--use_input_kernel`
If true, the input kernel will be used, if present. If false (or if the input kernel is not present) the values related to the instrumental resolution that are stored in BEST_FIT_PARM will be used.
[Default value: "TRUE"]

`--IFU_1:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 1. [Default value: "-1"]
`--IFU_2:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 2 [Default value: "-1"]
`--IFU_3:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 3 [Default value: "-1"]
`--IFU_4:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 4 [Default value: "-1"]
`--IFU_5:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 5 [Default value: "-1"]
`--IFU_6:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 6 [Default value: "-1"]
`--IFU_7:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 7 [Default value: "-1"]
`--IFU_8:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 8 [Default value: "-1"]
`--IFU_9:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 9 [Default value: "-1"]
`--IFU_10:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 10 [Default value: "-1"]
`--IFU_11:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 11 [Default value: "-1"]
`--IFU_12:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 12 [Default value: "-1"]
`--IFU_13:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 13 [Default value: "-1"]

`--IFU_14:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 14 [Default value: “-1”]
`--IFU_15:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 15 [Default value: “-1”]
`--IFU_16:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 16 [Default value: “-1”]
`--IFU_17:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 17 [Default value: “-1”]
`--IFU_18:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 18 [Default value: “-1”]
`--IFU_19:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 19 [Default value: “-1”]
`--IFU_20:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 20 [Default value: “-1”]
`--IFU_21:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 21 [Default value: “-1”]
`--IFU_22:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 22 [Default value: “-1”]
`--IFU_23:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 23 [Default value: “-1”]
`--IFU_24:` IFU number in ATMOS_PARM and BEST_FIT_PARM to be used to compute the telluric correction for IFU 24 [Default value: “-1”]
`--suppress_extension`
If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0

If IFU_Y (value) = -1, the following algorithm is used to determine the IFU_X (source) number in ATMOS_PARM and BEST_FIT_PARAM to compute the telluric correction in IFU_Y.

1. If IFU_Y in ATMOS_PARM and BEST_FIT_PARM contains data, the IFU_X=IFU_Y.

2. Otherwise the first IFU_first_chip_Y with valid data among those that belong to the same detector as IFU_Y will be used: IFU_X = IFU_first_chip_Y.

3. If there are no available IFU for that detector, the first IFU_first among the input data that contains data will be used: IFU_X=IFU_first.

7.1.8.2 Flow Chart

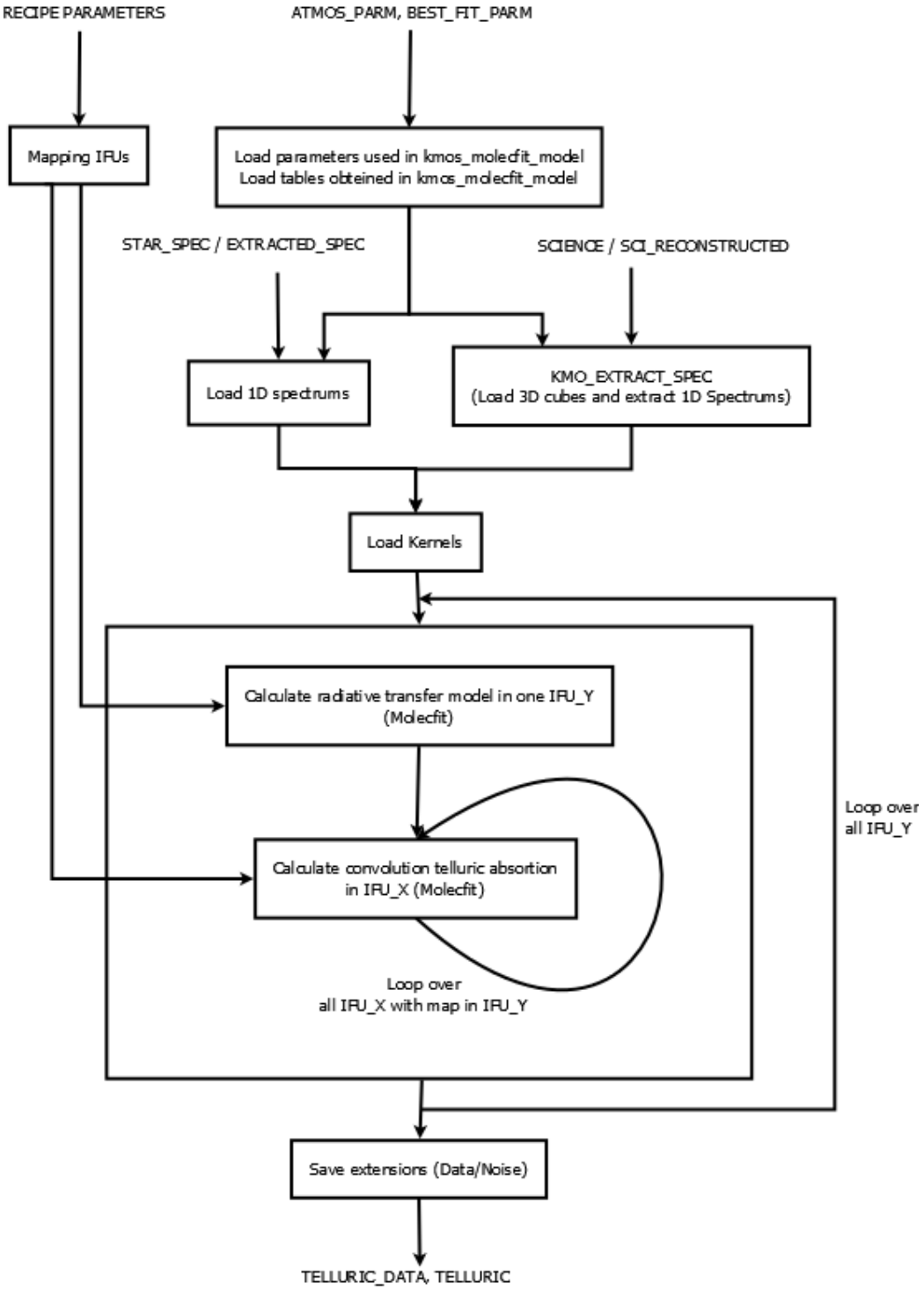

Figure 36**:** Flow chart of the recipe kmos_molecfit_calctrans

The processing steps are:

1. With the recipe parameters the IFU mapping is done.
2. The ATMOS_PARM and BEST_FIT_PARM are load.
3. Following the mapping IFUs the input raw data are extracted. They need to match with the extensions with data in the step 2.
4. If kernel library is provided, according to the rotation angle is charge the corresponding kernel for each input extension data.
5. Using molecfit, the radiative transfer model is calculated for each IFU_Y

6. For each IFU_X that have as reference the IFU_Y, using molecfit, the convolution telluric absortion is calculated.
7. Save the data of all of extension in the files TELLURIC_DATA and TELLURIC_CORR.

7.1.8.3 Input Frames

KMOS type	DO category	Amount	Comments
F1I	STAR_SPEC	0,1	Spectrum 1D
F1I	EXTRACT_SPEC	0,1	Spectrum 1D
F3I	SCIENCE	0,1	Spectrum 1D (in 3D cubes)
F3I	SCI_RECONSTRUCTED	0,1	Spectrum 1D (in 3D cubes)
F1L	ATMOS_PARM	1	Atmospheric model
F1L	BEST_FIT_PARM	1	Best fitting params
F2I	KERNEL_LIBRARY	0,1	Kernel library with grating dependency (equal to spec)

It is mandatory provide one input spectrum.

7.1.8.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
ESO INS GRAT1 ID	string	"IZ"/"YJ"/"H"/"K"/"HK"	Grating used in the file
ESO OCS ROT NAANGLE	float	[-360:360]	Rotation angle

Sub Headers

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.DATA" / "IFU.XX.NOISE"	Name of extension
CRVAL1	double	[0:2.5]	Initial wavelength (1D)
CDELT1	double	>0	Wavelength step (1D)
CRVAL3	double	[0:2.5]	Initial wavelength (3D)
CDELT3	double	>0	Wavelength step (3D)

Primary Headers: ATMOS_PARM, BEST_FIT_PARM

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.DATA" / "IFU.XX.NOISE"	Name of extension
KMOS MOLECFIT MODEL PARAM wave_range	string	"x1_start,x1_end; x2_start,x2_end"	A list of wavelength ranges to fit.
KMOS MOLECFIT MODEL PARAM list_molec	string	"H20[,CH4][,CO][, CO2][,O2]"	A list of molecules to fit.
KMOS MOLECFIT MODEL PARAM fit_molec	string	"1|0[*(,1|0)]"	Flags to fit the column density.
KMOS MOLECFIT MODEL PARAM relcol	string	"1|0[*(,1|0)]"	Column density relative to atmospheric profile.
KMOS MOLECFIT MODEL PARAM kernmode	bool	TRUE, FALSE	Voigt profile approximate or independent Gauss and Lorentz.
KMOS MOLECFIT MODEL	double	>=3.0 && <= 30.	Size of Gaussian/Lorentz/Voigt

PARAM kernfac			kernel in FWHM.
KMOS MOLECFIT MODEL PARAM varkern	bool	TRUE, FALSE	Does the kernel size increase linearly with wavelength?
KMOS MOLECFIT MODEL PARAM fit_cont	bool	TRUE, FALSE	Flag to enable/disable the polynomial fit of the continuum
KMOS MOLECFIT MODEL PARAM cont_n	int	>=0 && <= 8	Polynomial degree of the continuum fit.
KMOS MOLECFIT MODEL PARAM fit_wlc	bool	TRUE, FALSE	Flag to enable/disable the refinement of the wavelength.
KMOS MOLECFIT MODEL PARAM wlc_n	int	>=1 && <= 8	Polynomial degree of the refined wavelength solutions
KMOS MOLECFIT MODEL PARAM wlc_const	double	>= 0.0	Initial constant term for wavelength adjustement (shift relative to half wavelength range)

Primary Headers: KERNEL_LIBRARY

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.ANGLE.YYY"	Name of extension
CRVAL1	double	[0:2.5]	Initial wavelength
CD2_2	double	>0	Wavelength step

7.1.8.5 Configuration Parameters

Name	Type	valid values	Default	Comments
use_input_kernel	bool	TRUE, FALSE	TRUE	Flag to use the provided kernel of the values specified in BEST_FIT_PARM
IFU.1	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.2	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.3	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.4	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.5	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.6	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.7	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.8	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.9	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.10	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.11	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.12	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.13	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.14	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.15	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.16	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.17	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.18	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.19	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.20	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.21	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.22	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.23	int	-1 | [1:24]	-1	IFU input frames to be used
IFU.24	int	-1 | [1:24]	-1	IFU input frames to be used

7.1.8.6 Output Frames

KMOS type	DO Category	Comments
F1L	TELLURIC_DATA	Telluric correction and intermediate products
F1I	TELLURIC_CORR	Telluric correction

7.1.8.7 Examples

```
$ esorex kmos_molecfit_calctrans mf_calctrans.sof
                              with std.sof containing:
                              science_A.fits            SCIENCE
                              ATMOS_PARM_IZIZIZ.fits    ATMOS_PARM
                              BEST_FIT_PARM_IZIZIZ.fits BEST_FIT_PARM
                              kernel_iz.fits            KERNEL_LIBRARY
```

7.1.9 kmos_molecfit_correct: Apply the telluric correction

Read the results from kmos_molecfit_calctrans and apply the telluric correction for a scientific input data.

7.1.9.1 Description

The recipe divides the input scientific data either by the instrument response (if RESPONSE input is provided) or by the telluric correction (if TELLURIC input is provided). The scientific input data can have the following categories:

- STAR_SPEC (24 DATA plus 24 NOISE extensions).
- EXTRAC_SPEC (24 DATA extensions, additional 24 NOISE extensions are optional).
- SCIENCE (24 DATA extensions, additional 24 NOISE extensions are optional).
- SCI_RECONSTRUCTED (24 DATA extensions, additional 24 NOISE extensions are optional).

It is not mandatory that all the DATA extensions contains data.
The recipe will be run on all the extensions that contain data, including noise extensions.

The input corrections (TELLURIC_CORR or RESPONSE) need to have 24 DATA extensions. Additional 24 noise extensions are optional.

The output products can have the following categories, depending on the format of the input scientific data:

- SINGLE_SPECTRA (1D scientific input).
- SINGLE_CUBES (3D scientific input).

7.1.9.2 Flow Chart

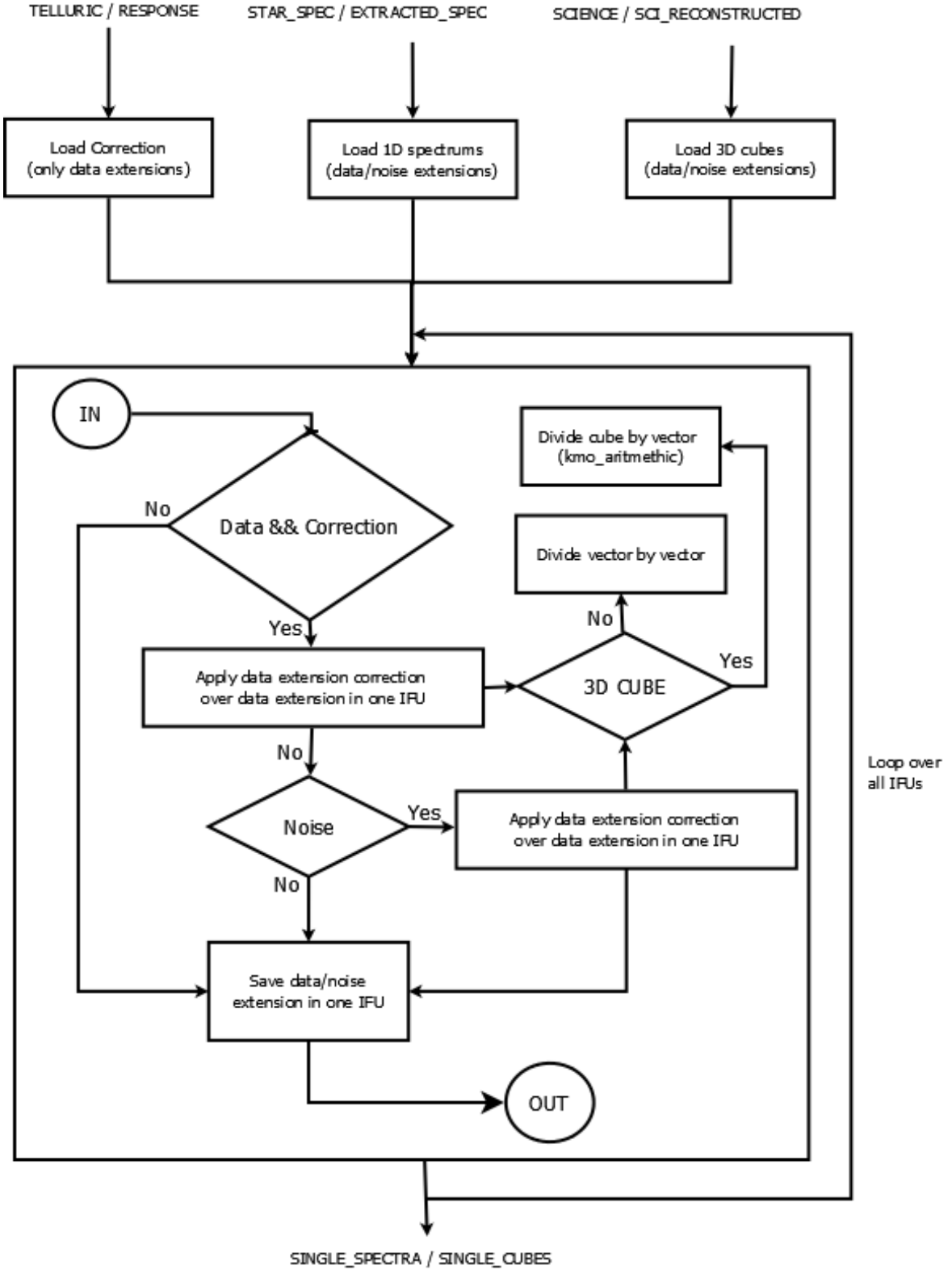

Figure 37**:** Flow chart of the recipe kmos_molecfit_correct

The processing steps are:

1. The input raw and the correction data are extracted.
2. For each IFU with data, if exist correction apply it to the extensions data and noise.
3. Save the data of all of extension in the files SINGLE_SPECTRA or SINGLE_CUBES, depending of the input raw dimensions.

7.1.9.3 Input Frames

KMOS type	DO category	Amount	Comments
F1I	STAR_SPEC	0,1	Scientific spectra 1D
F1I	EXTRACT_SPEC	0,1	Scientific spectra 1D
F3I	SCIENCE	0,1	Scientific cubes 3D
F3I	SCI_RECONSTRUCTED	0,1	Scientific cubes 3D
F1L	TELLURIC_CORR	0,1	Telluric correction (24/48ext)
F1S	RESPONSE	0,1	Response correction (24/48 ext)

It is mandatory provide one input spectrum and Telluric or Response correction.

7.1.9.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
ARCFILE	string	"Name_file"	Archive filename

Sub Headers

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.DATA" / "IFU.XX.NOISE"	Name of extension

Primary Headers: TELLURIC, RESPONSE

Keyword	Type	Value	Comments
EXTNAME	string	"IFU.XX.DATA" / "IFU.XX.NOISE"	Name of extension

7.1.9.5 Output Frames

KMOS type	DO Category	Comments
F1I	SINGLE_SPECTRA	1D input data corrected file.
F3I	SINGLE_CUBES	3D input data corrected file.

7.1.9.6 Examples

```
$ esorex kmos_molecfit_correct mf_correct.sof
                              with std.sof containing:
                              SCI_RECONSTRUCTED.fits    SCI_RECONSTRUCTED
                              TELLURIC_CORR_IZIZIZ.fits TELLURIC_CORR
```

7.2 Science Reduction Recipes

In this section two recipes are described which do in fact the same thing: reconstructing the data and combining the corresponding cubes. The way they do it differs significantly.

kmos_sci_red implements the straight approach, which is as well the standard one. First all the data frames are reconstructed and saved to disk. Then all data cubes are shifted and combined in a second step. In this recipe the data will be interpolated twice, during reconstruction and shifting.

Automatic object-sky association

In a series of science exposures the IFUs can be allocated to objects and skies arbitrarily. The recipes therefore analize for every IFU its contents and which sky will be subtracted from which object. If several skies are available, the one closest in time to the object will be picked. If there is no sky available, there won't be any sky subtraction applied. The association table is displayed in the recipe output (see as well the SPARK instructional guide) and stored to disk by default.
The generated obj_sky_table.txt can be edited to choose for example the sky from another exposure. In this case the ID for the sky frame can be altered.
If e.g. an object should be subtracted from another object, the second object could be marked as 'S' and the ID be set accordingly.
If for any reason no sky at all is available for a specific IFU one can as well indicate to use the sky from another IFU. Therefore the IFU ID is appended to the frame ID, separated by a slash. In this case residues are expected due to the different signatures of the IFUs.
For a more detailed examples see the SPARK instructional guide.

7.2.1 kmos_sci_red: Processing for Science Data

Reconstruct obj/sky-pairs individually and combine them afterwards.

7.2.1.1 Description

Ideally at least two data frames have to be provided since we need for each IFU pointing to an object as well a sky frame for the same IFU.
If an OH spectrum is given in the SOF file the lambda axis will be corrected using the OH lines as reference.
Every IFU containing an object will be reconstructed and divided by telluric and illumination correction, if provided. By default these intermediate cubes are saved to disk. Frames just containing skies won't produce an output here, so the number of output frames can be smaller than the number of input frames.
Then the reconstructed objects with the same object name are combined. These outputs are also saved to disk, the number of created files depends on the number of reconstructed objects of different name. If the user just wants to combine a certain object, the parameters `--name` or `--ifus` can be used. When the ZPOINT is available in the provided TELLURIC frame header, the reconstructed and combined cubes will be converted in physical units.
The reconstructed cubes and the combined cubes can be collapsed and the collapsed image stored as an additional product if requested with –collapse_combined and –collapse_reconstructed.
For exposures taken with the templates KMOS_spec_obs_mapping8 and KMOS_spec_obs_mapping24 the recipe behaves a bit different: All active IFUs will be combined, regardless of the object names.

Basic parameters:
`--imethod`
The interpolation method used for reconstruction.

`--smethod`
The interpolation method used for shifting.

`--lcmethod`
The level correction method that needs to be applied to the input frames is specified here. It can be either 'NONE' (for no correction), 'OSCAN' (overscan correction), 'SLICES_MEAN' or 'SLICES_MEDIAN' (for the LCAL inter-slices corrections).

`--name`
`--ifus`
Since an object can be present only once per exposure and since it can be located in different IFUs for the existing exposures, there are two modes to identify the objects:

- Combine by object names (default)
  In this case the object name must be provided via the `--name` parameter. The object name will be searched for in all primary headers of all provided frames in the keyword ESO OCS ARMx NAME.
- Combine by index (advanced)
  In this case the `--ifus` parameter must be provided. The parameter must have the same number of entries as frames are provided, e.g. \"3;1;24\" for 3 exposures. The index doesn't reference the extension in the frame but the real index of the IFU as defined in the EXTNAME keyword (e.g. 'IFU.3.DATA').

--collapse_reconstructed
--collapse_combined
The cubes are collapsed in a single image.

Advanced parameters:

--flux
Specify if flux conservation should be applied.

--background
Specify if background subtraction should be applied during the cube reconstruction.

--suppress_extension
If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0

--sky_tweak
If set to TRUE sky subtraction is not done by subtracting the corresponding detector images but subtracting a modified sky cube from the object cube. It is not allowed that --sky_tweak and --no_subtract both are TRUE.

--discard_subband
If set to TRUE, the last sub-band in the sky tweaking will be ignored.

--stretch
If set to TRUE, the SKY cube is stretched (polynomial of degree stretch_degree, resampling method specified by stretch_resampling) before the sky tweaking computation. The stretching polynomial is computed by fitting the sky lines found in the OBJ cube with those found in the SKY cube.
--stretch_degree
The degree of the stretching polynomial.

--stretch_resampling
The stretching resampling method.

--skip_sky_oh_align
If TRUE, the OH lines correction using the OH_SPEC input calibration file is not applied on the sky cube. This task is then left to the stretching algorithm. Skipping the OH lines correction is only possible if the stretching is applied.

--save_interims
Save interim object and sky cubes. Can only be used together with --sky_tweak

--tbsub
If set to TRUE subtract the thermal background from the cube resulting from sky tweaking. Default value is TRUE.

--obj_sky_table
The automatic obj-sky-associations can be modified by indicating a file with the desired associations. Therefore the file written to disk by default (without setting this option) can be edited ma-

nually. The formatting must absolutely be retained, just the type codes ('O' and 'S') and the associated frame indices should be altered.

`--velocity_offset`
Specify velocity offset correction in km/s for lambda scale.

Advanced reconstruction parameters:
`--neighborhoodRange`
Defines the range to search for neighbors during reconstruction

`--b_samples`
The number of samples in spectral direction for the reconstructed cube. Ideally this number should be greater than 2048, the detector size.

`--b_start`
`--b_end`
Used to define manually the start and end wavelength for the reconstructed cube. By default the internally defined values are used (see Section 6.3).

`--fast_mode`
If set to TRUE, the reconstructed cubes will be collapsed (using median) and only then be shifted and combined.

`--pix_scale`
Change the pixel scale [arcsec]. Default of 0.2\" results into cubes of 14x14pix, a scale of 0.1 results into cubes of 28x28pix, etc.

`--no_subtract`
If set to TRUE, the found objects and references won't be sky subtracted. Additionally all IFUs will be reconstructed, even the ones containing skies. This option sets the parameter `no_combine` to TRUE automatically.

`--xcal_interpolation`
If TRUE interpolate the pixel position in the slitlet (xcal) using the two closest rotator angles in the calibration file. Otherwise take the values of the closest rotator angle.

`--extrapolate`
By default no extrapolation is applied. This means that the intermediate reconstructed cubes will shrink at most one pixel, which is ok for templates like KMOS_spec_obs_nodtosky or KMOS_spec_obs_freedither. When the cubes will be arranged as a map, a grid is likely to occur between the IFUs. Therefore extrapolation during the shifting process can be switched on in order to get IFUs of original size. For frames taken with mapping templates, extrapolation is switched on automatically.

Advanced combining parameters:
`--edge_nan`
Set borders of two sides of the cubes to NaN before combining them. This minimises unwanted border effects when dithering.

`--no_combine`

If set to TRUE, the reconstructed cubes will not be combined.

`--method`
There are following sources to get the shift parameters from:

- header (default)
  The shifts are calculated according to the WCS information stored in the header of every IFU. The output frame will get larger, except the object is at the exact same position for all exposures. The size of the exposures can differ, but the orientation must be the same for all exposures.
- none
  The cubes are directly recombined, not shifting at all. The ouput frame will have the same dimensions as the input cubes.
  If the size differs a warning will be emitted and the cubes will be aligned to the lower left corner. If the orientation differs a warning will be emitted, but the cubes are combined anyway.
- center
  The shifts are calculated using a centering algorithm. The cube will be collapsed and a 2D profile will be fitted to it to identify the centre. With the parameter `--fmethod` the function to fit can be provided. The size of the exposures can differ, but the orientation must be the same for all exposures.
- user
  Read the shifts from a user specified file. The path of the file must be provided using the `--filename` parameter. For every exposure (except the first one) two shift values are expected per line, they have to be separated with simple spaces. The values indicate pixel shifts and are referenced to the first frame. The 1st value is the shift in x-direction to the left, the 2nd the shift in y-direction upwards. The size of the exposures can differ, but the orientation must be the same for all exposures.

`--fmethod`
see `--method` = "center"
The type of function that should be fitted spatially to the collapsed image. This fit is used to create a mask to extract the spectrum of the object. Valid values are "gauss" and "moffat".

`--filename`
see `--method` = "user"

`--cmethod`
Following methods of frame combination are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:
  ```
  val > mean + stdev * cpos_rej
  ```
  and
  ```
  val < mean - stdev * cneg_rej
  ```
  where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2).
- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.

- **sum**
  At each pixel position the sum is calculated.

min_max
The specified number of minimum and maximum pixel values will be rejected.
`--cmax` and `--cmin` apply to this method.

`--cpos_rej`
`--cneg_rej`
`--citer`
see `--cmethod` = ”ksigma”

`--cmax`
`--cmin`
see `--cmethod` = “min_max”

7.2.1.2 Flow Chart

To simplify the flowchart the internal data organising workflow isn’t depicted. All steps apply individually to each active IFU containing an object and a sky exposure.
The reduced data and noise cube is stored in a similar manner as the input frames.

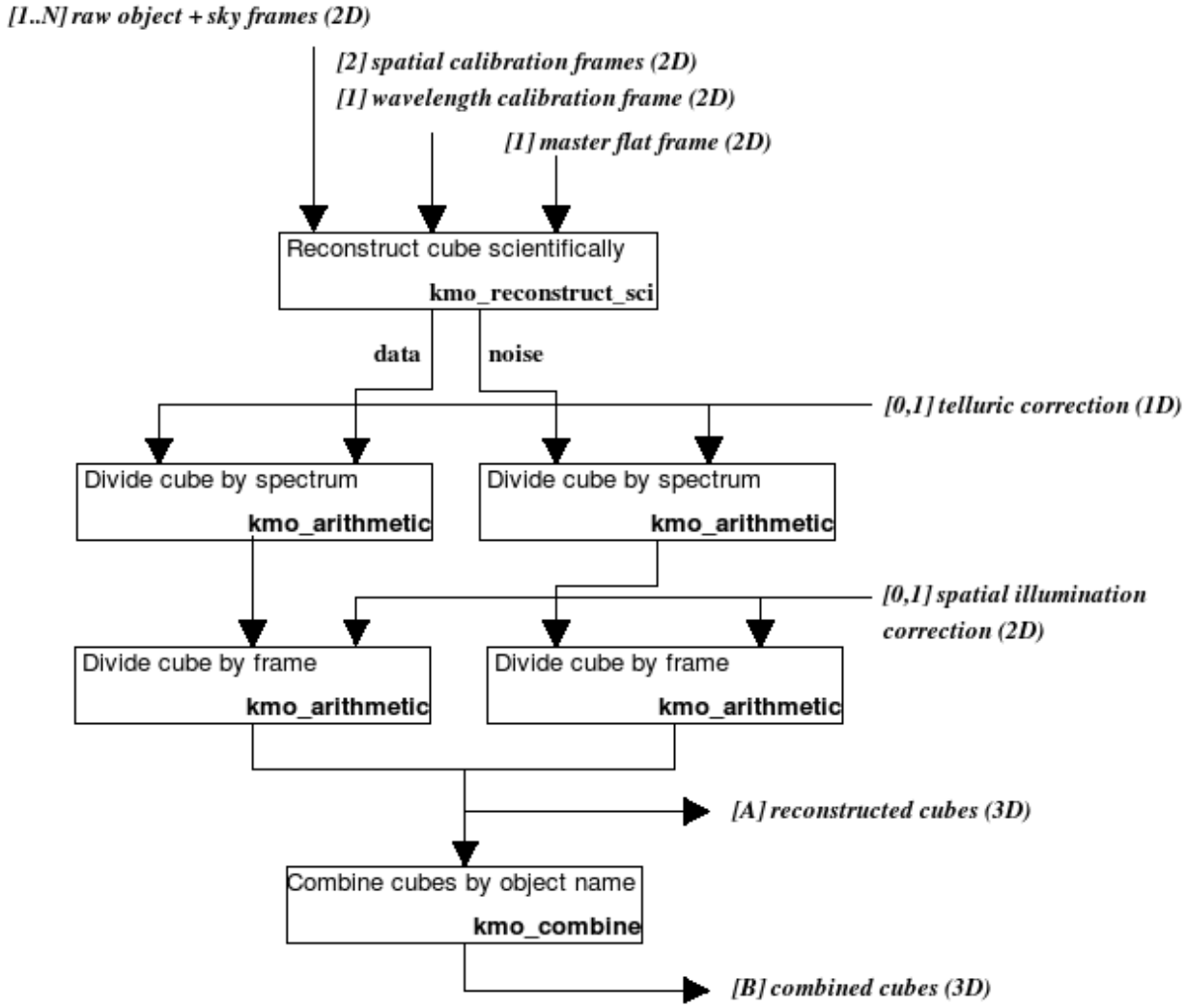

Figure 38**:** Flow chart of the recipe kmos_sci_red

The processing steps are:

1. The raw object is sky subtracted and reconstructed into a cube
2. The resulting data and noise cubes are divided by the telluric spectrum each.
3. Both data and noise cubes are divided spatially by the illumination correction.
4. The cubes are converted in physical units when possible.
5. Above steps are repeated for each IFU containing an object.

The zeropoint is written as the QC parameter `QC.ZPOINT` and is defined so that

`mag = qc.zpoint – 2.5log10(ADU/sec)`

where `mag` is the magnitude of a source that has a mean count rate of `ADU/sec` per spectral pixel. You can then convert the magnitude to a flux density in `ERG/sec/cm2/A`. Putting these steps together you have

`flux density = cts/sec × F0 × 10^[-0.4 × qc.zpoint] / 10.`

where F_0 is the zero magnitude flux density taken from the table below in whichever units are preferred.
The QC CUBE_UNIT contains the unit of the output cube, indicating whether the conversion has been made or not.

KMOS band	*2MASS band*	*Band pass for calibration*	*Zero magnitude flux density*	
K	K	2.028 –2.290 μm	4.283×10^{-10} W/m^2/μm	4.65×10^{9} ph/s/m^2/μm
HK	H & K	1.5365 – 1.7875 μm + 2.028 - 2.290 μm	1.133×10^{-9} W/m^2/μm & 4.283×10^{-10} W/m^2/μm	9.47×10^{9} ph/s/m^2/μm & 4.65×10^{9} ph/s/m^2/μm
H	H	1.5365 – 1.7875 μm	1.133×10^{-9} W/m^2/μm	9.47×10^{9} ph/s/m^2/μm
YJ	J	1.154 – 1.316 μm	3.129×10^{-9} W/m^2/μm	1.944×10^{9} ph/s/m^2/μm
IZ	—	0.985 – 1.000 μm	7.63×10^{-9} W/m^2/μm	3.81×10^{10} ph/s/m^2/μm

When applying advanced OH handling techniques, the reconstruction step in **Figure** 38 gets a bit more complicated (see Figure 39). There are three corrections available:

The first takes a reference OH lines catalog (OH_SPEC) as input and matches the reconstructed OH lines with the catalog. A 2nd degree polynomial fit is then applied to the Object cube and Sky cubes.

The second stretches the SKY cube in order to align the sky lines visible in it to those that appear in the OBJ cube. The stretching is a relatively high degree polynomial, and allows to remove the residuals that remain from the first correction.

The third removes thermal background and compensates vibrational variations (the same functionality can be used stand-alone with recipe kmos_sky_tweak, see section 7.2.3.8).

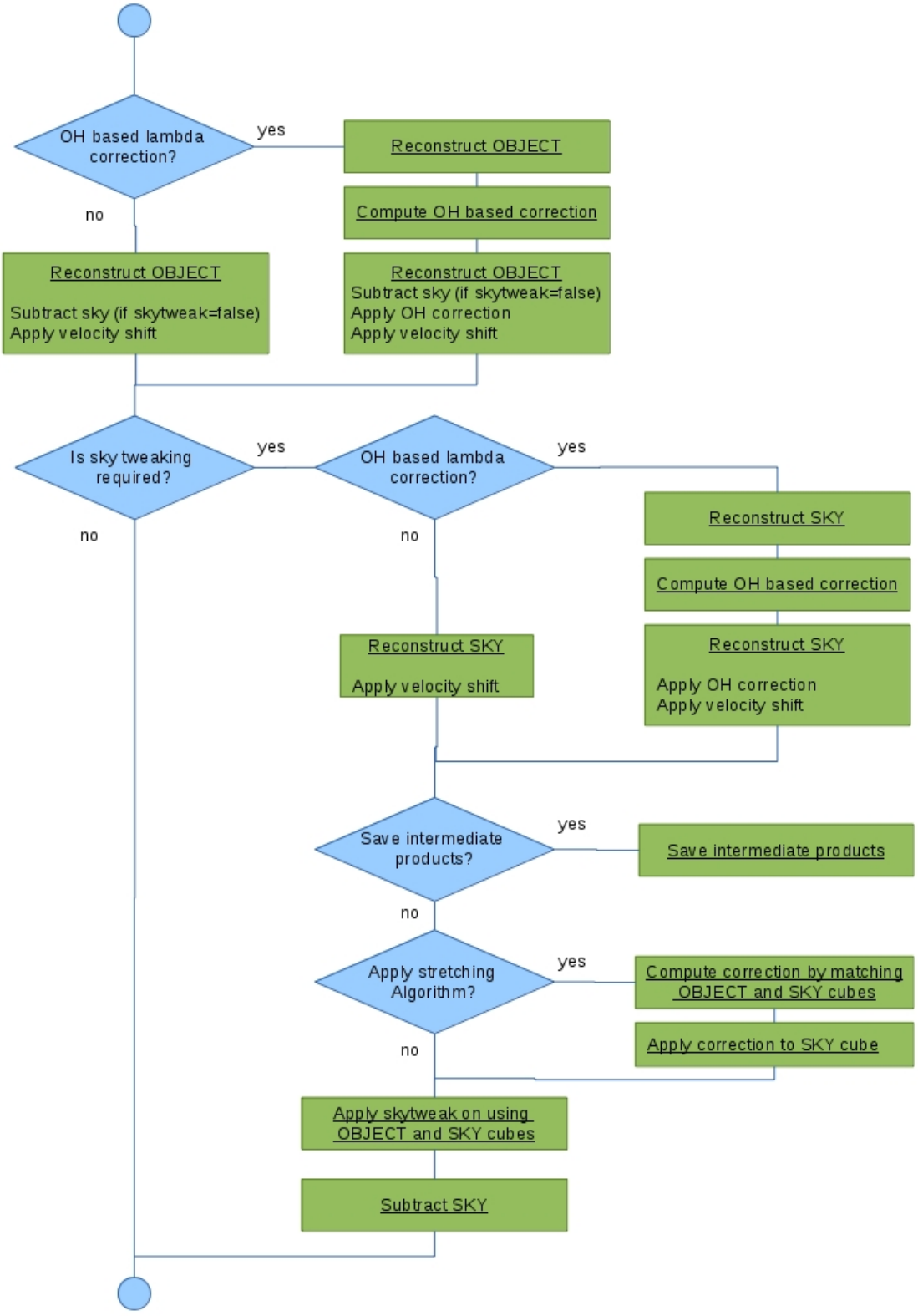

Figure 39 Flowchart of the two advanced OH handling techniques

7.2.1.3 Input Frames

KMOS type	DO category	Amount	Comments

RAW	SCIENCE	≥1	The science frames
F2D	XCAL	1	Calibration frame 1 *(from* kmos_flat*)*
F2D	YCAL	1	Calibration frame 2 *(from* kmos_flat*)*
F2D	LCAL	1	Calibration frame *(from* kmos_wave_cal*)*
F2L	WAVE_BAND	1	Table with start-/end-values of wavelengthrange
F2D	MASTER_FLAT	0,1	*(from* kmos_flat*)*
F2I	ILLUM_CORR	0,1	*(from* kmos_illumination*)*
F1I	TELLURIC	0,1	*(from* kmos_std_star*)*
F1S	OH_SPEC	0,1	OH reference spectrum

7.2.1.4 Fits Header Keywords

Primary Header

None

Sub Headers

None

7.2.1.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
imethod	string	"NN" "lwNN" "swNN" "MS" "CS"	"CS"	Interpolation method: NN: Nearest Neighbor lwNN: linear weighted NN swNN: square weighted NN CM: Modified Shephard CS: cubic spline *(optional)*
smethod	string	"NN" "CS"	"CS"	Interpolation method: NN: Nearest Neighbor CS: cubic spline *(optional)*
name	string	any	""	Name of the object to combine as defined in the keyword ESO OCS ARMi NAME *(if this parameter is set, the --ifus parameter can't be set)*
ifus	string	"ifu1;ifu2;..."	""	The indices of the IFUs to combine. The number of entries has to match the number of input frames *(if this parameter is set, the --name parameter can't be set*

collapse_reconstructed	bool	TRUE, FALSE	FALSE	Create the collapsed reconstructed cubes
collapse_combined	bool	TRUE, FALSE	FALSE	Create the collapsed combined cubes

Advanced parameters

Name	Type	valid values	Default	Comments
flux	bool	TRUE, FALSE	FALSE	Apply flux conservation
background	bool	TRUE, FALSE	FALSE	
suppress_extension	bool	TRUE, FALSE	FALSE	
sky_tweak	bool	TRUE, FALSE	FALSE	Use modified sky cube for sky subtraction
diiscard_subband	bool	TRUE, FALSE	FALSE	Ignore the last sub-band in the sky tweaking
stretch	bool	TRUE, FALSE	FALSE	
stretch_degree	int	>=0	8	
stretch_resampling	string	linear / spline	spline	
skip_sky_oh_align	bool	TRUE, FALSE	FALSE	Only if stretch is on
save_interims	bool	TRUE, FALSE	FALSE	Save intermediate reconstructed data
tbsub	bool	TRUE, FALSE	TRUE	Subtract thermal background from input cube
obj_sky_table	string	any	“”	*(optional)* If the obj-sky-association should be altered
velocity_offset	double	any	0.0	Add a velocity offset to the lambda scale. A value of 0.0 does nothing. Values are expected to be small.
neighborhoodRange	double	≥ 1	1.001	Defines the range to search for neighbors
b_samples	int	b_samples > 2	2048	Nr. of samples of reconstructed data for the wavelength
b_start *b_end*	double	b_start > 0.0 b_end > b_start	-1.0	Start and end wavelength. The defaults of -1.0 instruct to use the internally defined range (see Section 6.3)
fast_mode	bool	TRUE, FALSE	FALSE	TRUE if cubes should be collapsed before combining
pix_scale	double	TRUE, FALSE	0.2	The pixel scale: 0.2 arcsec results in cubes of 14x14 pixels. 0.1 arcsec result in cubes of 28x28 pixels
no_subtract	bool	TRUE, FALSE	FALSE	Don’t subtract cubes
xcal_interpolation	bool	TRUE, FALSE	TRUE	*(optional)*
extrapolate	bool	TRUE, FALSE	FALSE	FALSE: shifted IFU will be filled with NaNs at the borders TRUE: shifted IFU will be extrapolated at the borders *(optional, applies only when*

				smethod=CS and doing sub pixel shifts)
edge_nan	bool	TRUE, FALSE	FALSE	Set two sides of the cubes to NaN
no_combine	bool	TRUE, FALSE	FALSE	Don't combine cubes
method	string	"none" "header" "center" "user"	"header"	The shifting method
fmethod	string	"gauss" or "moffat"	"gauss"	The 2D function to fit to the collapsed cube
filename	string	any	""	The path to the file with the shift vectors. *(applies only to --method = "user")*
cmethod	string	"ksigma" "min_max" "average" "median" "sum"	"ksigma"	The averaging method to apply *(optional)*
cpos_rej *cneg_rej*	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = "ksigma")*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only when --cmethod = "ksigma")*
cmax *cmin*	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip with min/max-clipping *(optional, applies only when --cmethod = "min_max")*

7.2.1.6 Output Frames

KMOS type	DO Category	Comments
F3I	SCI_COMBINED	Combined cubes with noise
F3I	SCI_RECONSTRUCTED	Reconstructed cubes
F2I	EXP_MASK	Exposure time frame, every spaxel indicates how many input frames are taken into account when combining
F3I	SCI_INTERIM_OBJECT	*(optional)* Intermediate reconstructed object cubes used for sky tweaking, no noise (set `--sky_tweak` and `--save_interims`)
F3I	SCI_INTERIM_SKY	*(optional)* Intermediate reconstructed sky cubes used for sky tweaking, no noise (set `--sky_tweak` and `--save_interims`)

	SCI_COMBINED_COLL	*(optional)* Collapsed combined cubes (set –collapse_combined)
	SCI_RECONSTRUCTED_COLL	*(optional)* Collapsed reconstructed cubes (set –collapse_reconstructed)

7.2.1.7 Examples

```
$ esorex kmos_sci_red reduce.sof
```

with reduce.sof containing:

```
science_1.fits            SCIENCE
science_2.fits            SCIENCE
science_3.fits            SCIENCE
xcal_HHH.fits             XCAL
ycal_HHH.fits             YCAL
lcal_HHH.fits             LCAL
kmos_wave_band.fits       WAVE_BAND
master_flat_HHH.fits      MASTER_FLAT
illum_corr_HHH.fits       ILLUM_CORR
telluric_HHH.fits         TELLURIC
kmos_oh_spec_h.fits       OH_SPEC
```

7.2.2 kmos_reconstruct: Reconstructing a Cube

Performs the cube reconstruction using different interpolation methods.

7.2.2.1 Description

Data with or without noise is reconstructed into a cube using the calibration frames XCAL, YCAL and LCAL. XCAL and YCAL are generated using recipe kmos_flat, LCAL is generated using recipe kmos_wave_cal.
The input data can contain noise extensions and will be reconstructed into additional extensions.

Basic parameters:

`--imethod`
The interpolation method used for reconstruction.

`--lcmethod`
The level correction method that needs to be applied to the input frames is specified here. It can be either 'NONE' (for no correction) or 'OSCAN' (overscan correctio).

`--detectorimage`
Specify if a resampled image of the input frame should be generated. Therefore all slitlets of all IFUs are aligned one next to the other. This frame serves for quality control. One can immediately see if the reconstruction was successful.

`--file_extension`
Set to TRUE if OBS_ID (from input frame header) should be appended to the output frame.

Advanced parameters:

`--flux`
Specify if flux conservation should be applied.

`--neighborhoodRange`
Defines the range to search for neighbors during reconstruction

`--b_samples`
The number of samples in spectral direction for the reconstructed cube. Ideally this number should be greater than 2048, the detector size.

`--b_start`
`--b_end`
Used to define manually the start and end wavelength for the reconstructed cube. By default the internally defined values are used (see Section 6.3).

`--pix_scale`
Change the pixel scale [arcsec]. Default of 0.2\" results into cubes of 14x14pix, a scale of 0.1\" results into cubes of 28x28pix, etc.

`--xcal_interpolation`
If TRUE interpolate the pixel position in the slitlet (xcal) using the two closest rotator angles in the calibration file. Otherwise take the values of the closest rotator angle.

7.2.2.2 Flow Chart

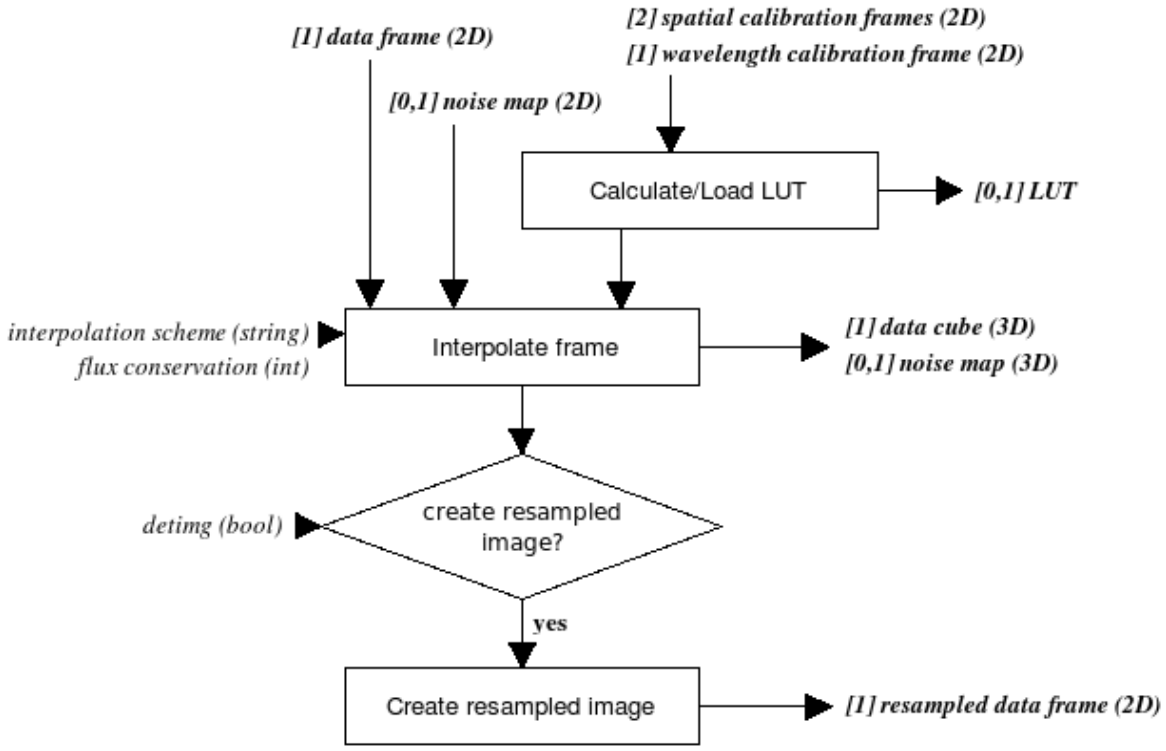

Figure 40**:** Flow chart of the recipe kmos_reconstruct

The processing steps are:

1. First the LUT for correcting spectral curvature and wavelength position is calculated and saved to disk or just loaded from disk (see Sec. 6.4)
2. Then the data cube and the optional noise map are interpolated according the LUT. Additionally the interpolation scheme can be chosen and if flux conservation should be applied.
3. If desired the reconstructed cube can also be saved as resampled image, meaning that the reconstructed cube is decomposed into its slitlets which are saved into a frame with one slitlet beside the other. This way the quality of reconstruction can be determined quicklys visually.

7.2.2.3 Input Frames

KMOS type	DO category	Amount	Comments
RAW or F2D	DARK or FLAT_ON or ARC_ON or OBJECT or STD or SCIENCE	1	data frame, with or without noise
F2D	XCAL	1	Calibration frame 1 *(from kmos_flat)*
F2D	YCAL	1	Calibration frame 2 *(from kmos_flat)*
F2D	LCAL	1	Calibration frame 3 *(from kmos_wave_cal)*
F2L	WAVE_BAND	1	Table with start-/end-values of wavelengthrange

F1S	OH_SPEC	0,1	OH reference spectrum

7.2.2.4 Fits Header Keywords

Primary Header

None

Sub Headers

None

7.2.2.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
imethod	string	"NN" "lwNN" "swNN" "MS" "CS"	"CS"	Interpolation method: NN: Nearest Neighbor lwNN: linear weighted NN swNN: square weighted NN MS: Modified Shepard's method CS: Cubic spline *(optional)*
detectorimage	bool	TRUE, FALSE	FALSE	TRUE if resampled detector image should be created, FALSE otherise
file_extension	bool	TRUE, FALSE	FALSE	TRUE if OBS_ID keyword should be appended to output frames, FALSE otherwise

Advanced parameters

Name	Type	valid values	Default	Comments
flux	bool	TRUE, FALSE	FALSE	Apply flux conservation
neighborhoodRange	double	≥ 1	1.001	Defines the range to search for neighbors
b_samples	int	b_samples > 2	2048	Nr. of samples of reconstructed data for the wavelength
b_start *b_end*	double	b_start > 0.0 b_end > b_start	-1.0	Start and end wavelength. The defaults of -1.0 instruct to use the internally defined range (see Section 6.3)
pix_scale	double	TRUE, FALSE	0.2	The pixel scale: 0.2 arcsec results in cubes of 14x14 pixels. 0.1 arcsec result in cubes of 28x28 pixels
xcal_interpolation	bool	TRUE, FALSE	TRUE	

7.2.2.6 Output Frames

KMOS type	DO Category	Comments
F3I	CUBE_DARK or	Reconstructed cube with or without

	CUBE_FLAT or CUBE_ARC or CUBE_OBJECT_SCIENCE or CUBE_SKY_SCIENCE	noise
F2D	DET_IMG_REC	if parameter `–detimg` has been set to TRUE

Additional Output

All recipes doing reconstruction of cubes create a LUT which by default is saved to disk. For further information see Sec. 6.4.

7.2.2.7 Examples

```
$ esorex kmos_reconstruct reconstruct.sof

                              with reconstruct.sof containing:
                              object_science.fits   OBJECT_SCIENCE
                              xcal_YJYJYJ.fits      XCAL
                              ycal_YJYJYJ.fits      YCAL
                              lcal_YJYJYJ.fits      LCAL
                              kmos_wave_band.fits   WAVE_BAND
```

7.2.3 kmos_combine: Combining Cubes

Combine cubes spatially.

7.2.3.1 Description

This recipe shifts several exposures of an object and combines them. The different methods to match the exposures are described below (`--method` parameter). The output cube is larger than the input cubes, according to the shifts to be applied. Additionally a border of NaN values is added. The WCS is the same as for the first exposure.
For each spatial/spectral pixel a new value will be calculated (according the `--cmethod` parameter) and written into the output cube.
Only exposures with equal orientation regarding the WCS can be combined (except `--method="none"`), north must point to the same direction. It is recommended to apply any rotation possibly after combining.

The behavior of the selection of IFUs to combine differs for some templates and can be controlled with the parameters `--name` and `--ifus`.
If the input data cubes stem from templates `KMOS_spec_obs_mapping8` or `KMOS_spec_obs_mapping24` all extensions from all input frames are combined into a single map by default (like in recipe `kmos_sci_red`). If just the area of a specific IFU should be combined, the parameter `--ifus` can be specified, or more easily `--name`.
If the input data cubes stem from other templates like e.g. `KMOS_spec_obs_freedither` all extensions of all input frames are combined into several output frames by default. The input IFUs are grouped according their targeted object name stored in the keywords `ESO OCS ARMx NAME`. If just a specific object should be combined, its name can be specified with parameter `--name`. If arbitrary IFUs should be combined, one can specify these with the parameter `--ifus`.

The default mapping mode is done via the `--name` parameter, where the name of the object has to be provided. The recipe searches in all input data cubes IFUs pointing to that object.

Basic parameters:
`--name`
`--ifus`
Since an object can be present only once per exposure and since it can be located in different IFUs for the existing exposures, there are two modes to identify the objects:

- Combine by object names (default)
  In this case the object name must be provided via the `--name` parameter. The object name will be searched for in all primary headers of all provided frames in the keyword ESO OCS ARMx NAME.
- Combine by index (advanced)
  In this case the `--ifus` parameter must be provided. The parameter must have the same number of entries as frames are provided, e.g. \"3;1;24\" for 3 exposures. The index doesn't reference the extension in the frame but the real index of the IFU as defined in the EXTNAME keyword (e.g. 'IFU.3.DATA').

`--method`
There are following sources to get the shift parameters from:

- none (default)
  The cubes are directly recombined, not shifting at all. The ouput frame will have the same dimensions as the input cubes.
  If the size differs a warning will be emitted and the cubes will be aligned to the lower left

corner. If the orientation differs a warning will be emitted, but the cubes are combined anyway.

- header
  The shifts are calculated according to the WCS information stored in the header of every IFU. The output frame will get larger, except the object is at the exact same position for all exposures. The size of the exposures can differ, but the orientation must be the same for all exposures. This method is needed for MAPPING data.
- center
  The shifts are calculated using a centering algorithm. The cube will be collapsed and a 2D profile will be fitted to it to identify the centre. With the parameter `--fmethod` the function to fit can be provided. The size of the exposures can differ, but the orientation must be the same for all exposures.
- user
  Read the shifts from a user specified file. The path of the file must be provided using the `--filename` parameter. For every exposure (except the first one) two shift values are expected per line, they have to be separated with simple spaces. The values indicate pixel shifts and are referenced to the first frame. The 1st value is the shift in x-direction to the left, the 2nd the shift in y-direction upwards. The size of the exposures can differ, but the orientation must be the same for all exposures.

  Example: Combine 3 frames in order to produce the final cube for target1. The input file "data.sof" looks like:

```
sci_reconstructed_1.fits SCI_RECONSTRUCTED
sci_reconstructed_2.fits SCI_RECONSTRUCTED
sci_reconstructed_3.fits SCI_RECONSTRUCTED
```

  the user-input shift file "shifts.dat" looks like:

```
 1 1

-1 -1
```

  1 1 are the shifts to be applied to sci_reconstructed_2.fits
  -1 -1 are the shifts to be applied to sci_reconstructed_3.fits
  shifts of sci_reconstructed_1.fits are assumed to be 0.

  The recipe call looks like:

```
   esorex kmos_combine --method=user --filename=shifts.dat
--name=target_1 data.sof
```

  Note1: kmos combine expects to find data in 3 arms that targets the object target_1 and will produce one final combined cube for that object.
  Note2: data.sof must contain only the exposures that have target_1

`--cmethod`
Following methods of frame combination are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:
  `val > mean + stdev * cpos_rej`
  and
  `val < mean - stdev * cneg_rej`
  where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2).
- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.
- **sum**
  At each pixel position the sum is calculated.
- **min_max**
  The specified number of minimum and maximum pixel values will be rejected.
  `--cmax` and `--cmin` apply to this method.

Advanced parameters:

`--edge_nan`
Set borders of two sides of the cubes to NaN before combining them. This minimises unwanted border effects when dithering

`--fmethod`
see `--method` = "center"
The type of function that should be fitted spatially to the collapsed image. This fit is used to create a mask to extract the spectrum of the object. Valid values are "gauss" and "moffat".

`--filename`
see `--method` = "user"

`--cpos_rej`
`--cneg_rej`
`--citer`
see `--cmethod` = "ksigma"

`--cmax`
`--cmin`
see `--cmethod` = "min_max"

`--flux`
Specify if flux conservation should be applied

`--suppress_extension`
If set to TRUE, the arbitrary filename extensions are suppressed. If multiple products with the same category are produced, they will be numered consecutively starting from 0

7.2.3.2 Flow Chart

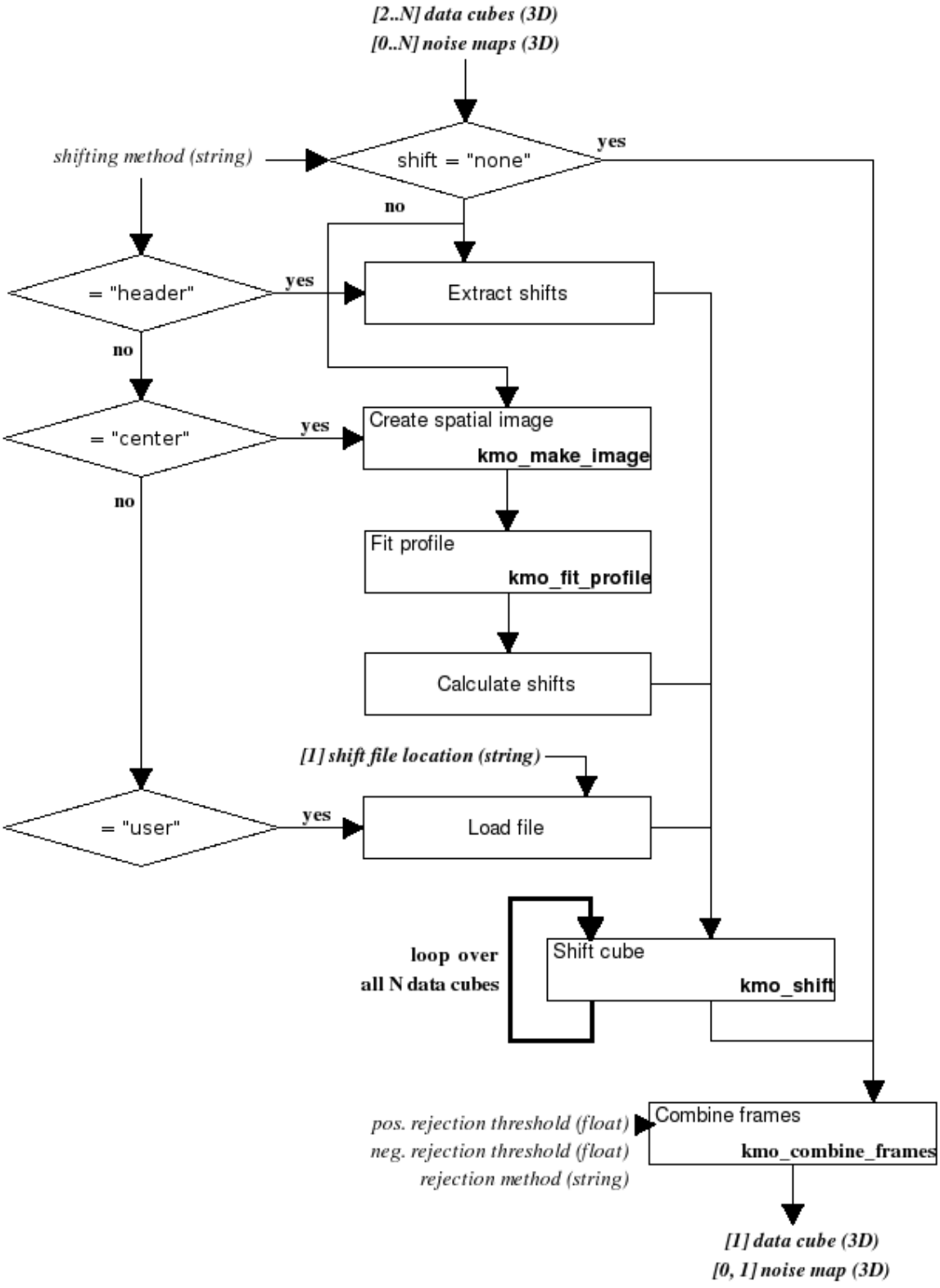

Figure 41**:** Flow chart of the recipe kmos_combine

The processing steps are:

1. The actions taken depend on the shifting method:
    a. “none”: Since no shifting is wanted the data and noise is directly propagated.
    b. “header”: The shift information is extracted from the fits file headers of the data cubes. All shifts are relative to the first cube in the list.
    c. “center”: The shifts are calculated using a centering algorithm. First the cubes are collapsed spatially, then a profile will be fit to find the centre of the object.
    d. “user”: The user provides a file with stored shift information, relative to the first cube in the list.
2. The actual shift is executed now.

The data cubes and corresponding noise maps are combined using rejection.

7.2.3.3 Input Frames

KMOS type	DO category	Amount	Comments
F31	Any	≥ 1	any F3I data frames, the DO category is propagated to the output

7.2.3.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	Int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
CRPIX1, CRPIX2, CRPIX3	double	any	all frames
CRVAL1, CRVAL2, CRVAL3	double	any	all frames
CDELT1, CDELT2, CDELT3	double	any	all frames
CD1_1, CD1_2, CD2_1 CD2_2	double	any	all frames

7.2.3.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
name	string	Any	“”	Name of the object to combine as defined in the keyword ESO OCS ARMi NAME *(if this parameter is set, the --ifus parameter can’t be set)*

ifus	string	"ifu1;ifu2;..."	“”	The indices of the IFUs to combine. The number of entries has to match the number of input frames *(if this parameter is set, the --name parameter can’t be set*
method	string	“none” “header” “center” “user”	“none”	The shifting method
cmethod	string	“ksigma” “min_max” “average” “median” “sum”	“ksigma”	The averaging method to apply *(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
edge_nan	bool	TRUE, FALSE	FALSE	
fmethod	string	“gauss” or ”moffat”	“gauss”	The 2D function to fit to the collapsed cube
filename	string	any	“”	The path to the file with the shift vectors. *(applies only to --method = "user")*
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = “ksigma”)*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only when --cmethod = “ksigma”)*
cmax cmin	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip with min/max-clipping *(optional, applies only when --cmethod = “min_max”)*
flux	bool	TRUE, FALSE	FALSE	
suppress_extension	bool	TRUE, FALSE	FALSE	

7.2.3.6 Output Frames

KMOS type	DO Category	Comments
F3I	COMBINED_CUBE	
F2I	COMBINED_IMAGE	
F2I	EXP_MASK	Exposure time frame, every spaxel indicates how many input frames are taken into account when combining

7.2.3.7 Examples

```
$ esorex kmos_combine –name=”NGC_150” combine.sof
                      with combine.sof containing:
                                 fits1_NGC_150_in_ifu_2.fits CUBE_OBJECT
                                fits2_NGC_150_in_ifu_17.fits CUBE_OBJECT
                                 fits3_NGC_150_in_ifu_9.fits CUBE_OBJECT
```

7.2.3.8 Combining exposures with different rotations

kmos_combine combines only exposures that have the same orientation on the sky. In order to combine 2 exposures (e.g., SINGLE_CUBES) that have different orientation, one has first to derotate one to the same reference of the other, using the value of the header keyword OCS ROT OFFSET = 120 as reference. The recipe kmo_rotate (Section 7.3.7) can be used for this purpose.

For example:

Combine the following reconstructed cubes
SINGLE_CUBES_1.fits with OCS ROT OFFSET = 0, and SINGLE_CUBES_2.fits with OCS ROT OFFSET = 120.

1. Type:
esorex kmo_rotate --rotations=120. SINGLE_CUBES_2.fits

It will generate the de-rotated file named ROTATE.fits

2. Create a combine.sof with the following entries:
SINGLE_CUBES_1.fits SINGLE_CUBES
ROTATE.fits SINGLE_CUBES

3. Combine the 2 exposures with:

esorex kmos_combine combine.sof

For more information about the parameters of kmo_rotate, please consult Section 7.3.7

7.2.4 kmos_level_correct: Level correction

Level correction method.

7.2.4.1 Description

The level correction method using SLICES method corrects separately boxes of 64x128 pixels spread on the full detector. In each box, the correction value is the MEAN or MEDIAN value of the intra slices pixels found in the LCAL image.

7.2.4.2 Input Frames

KMOS type	DO category	Amount	Comments
F2I	SCIENCE	≥1	Raw images
F2I	LCAL	1	LCAL
F2I	BADPIXEL_DARK	1	BADPIXEL_DARK

7.2.4.3 Configuration Parameters

`--lcmethod`

The level correction method that needs to be applied to the input frames is specified here. It can be either 'NONE' (for no correction), 'OSCAN' (overscan correction), 'SLICES_MEAN' or 'SLICES_MEDIAN' (for the LCAL inter-slices corrections).

7.2.4.4 Output Frames

KMOS type	DO Category	Comments
F2I	LEVEL_CORRECTED	Level corrected images

7.2.4.5 Examples

```
$ esorex kmos_level_correct sof
                                with reduce.sof containing:
                                raw1.fits                SCIENCE
                                raw2.fits                SCIENCE
                                LCAL.fits                LCAL
                                bpm.fits                 BADPIXEL_DARK
```

7.2.5 kmos_sky_tweak: Second Order Sky Subtraction

Removal of OH sky lines.

7.2.5.1 Description

The recipe, as implemented, is divided into 2 main processing steps: removal of thermal background (**Figure** 42) and compensation of vibrational variations (**Figure** 43).

7.2.5.2 Flow Chart

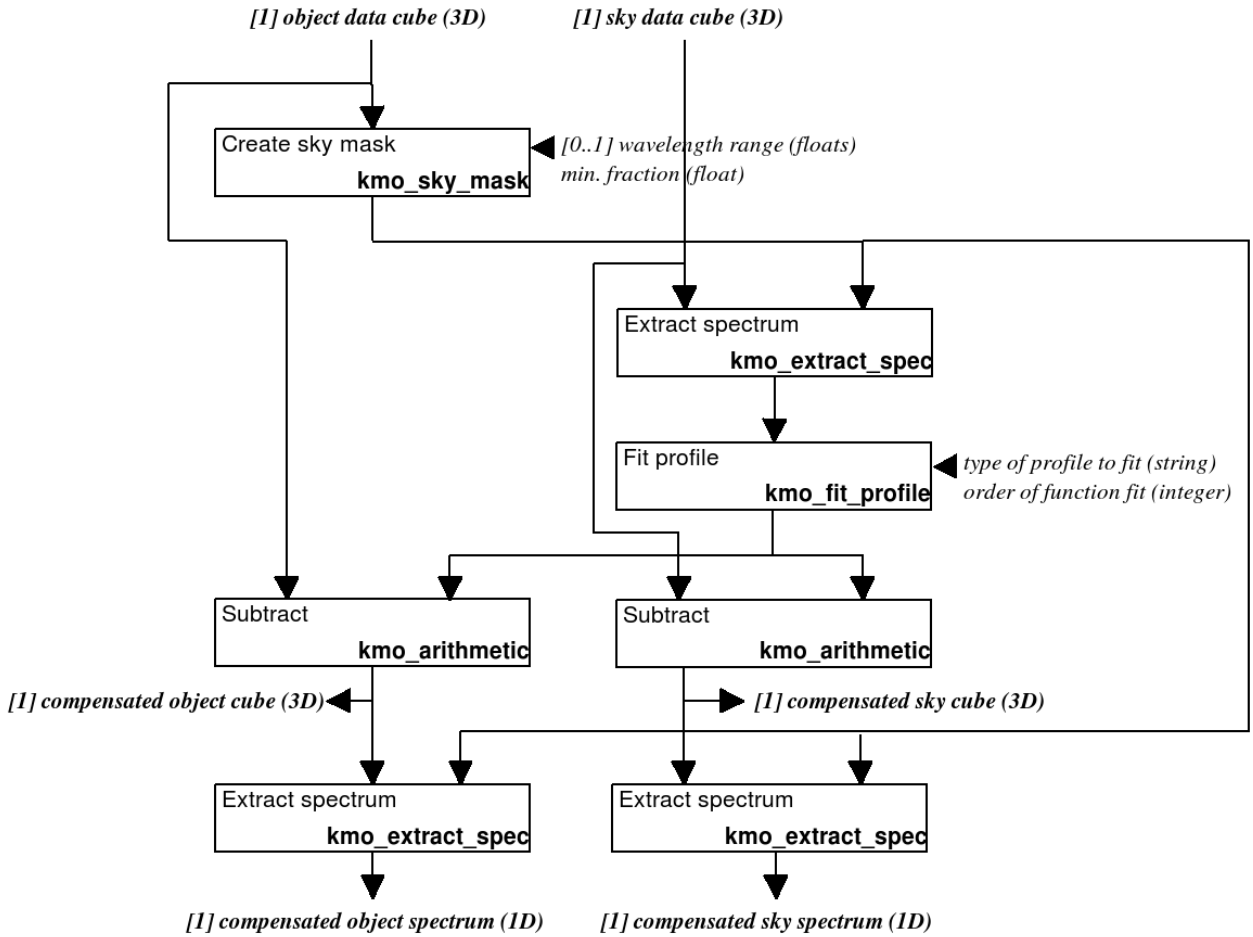

Figure 42**:** Flow chart of the recipe kmos_sky_tweak (Part 1)

The processing steps of **Figure** 42 are:

1. Identify spaxels with least flux in object cube (kmos_sky_mask).
2. Sum spectra from these spaxels in both object and sky cubes separately.
3. Fit a blackbody function to the underlying continuum in the sky spectrum (the thermal background).
4. The fitted function is subtracted from both the original object and sky cubes and from the extracted object and sky spectra.
5. The spectra with removed thermal background are compared with regard to offsets in bright OH lines. The sky cube (with removed thermal background) is shifted accordingly. Note that for KMOS, the default is for spectral flexure to already be corrected. However there may be some situations where this is not so, in which case this step is carried out here.
6. Again the spectrum of the processed object and sky cubes are extracted using the same mask as in step 1.

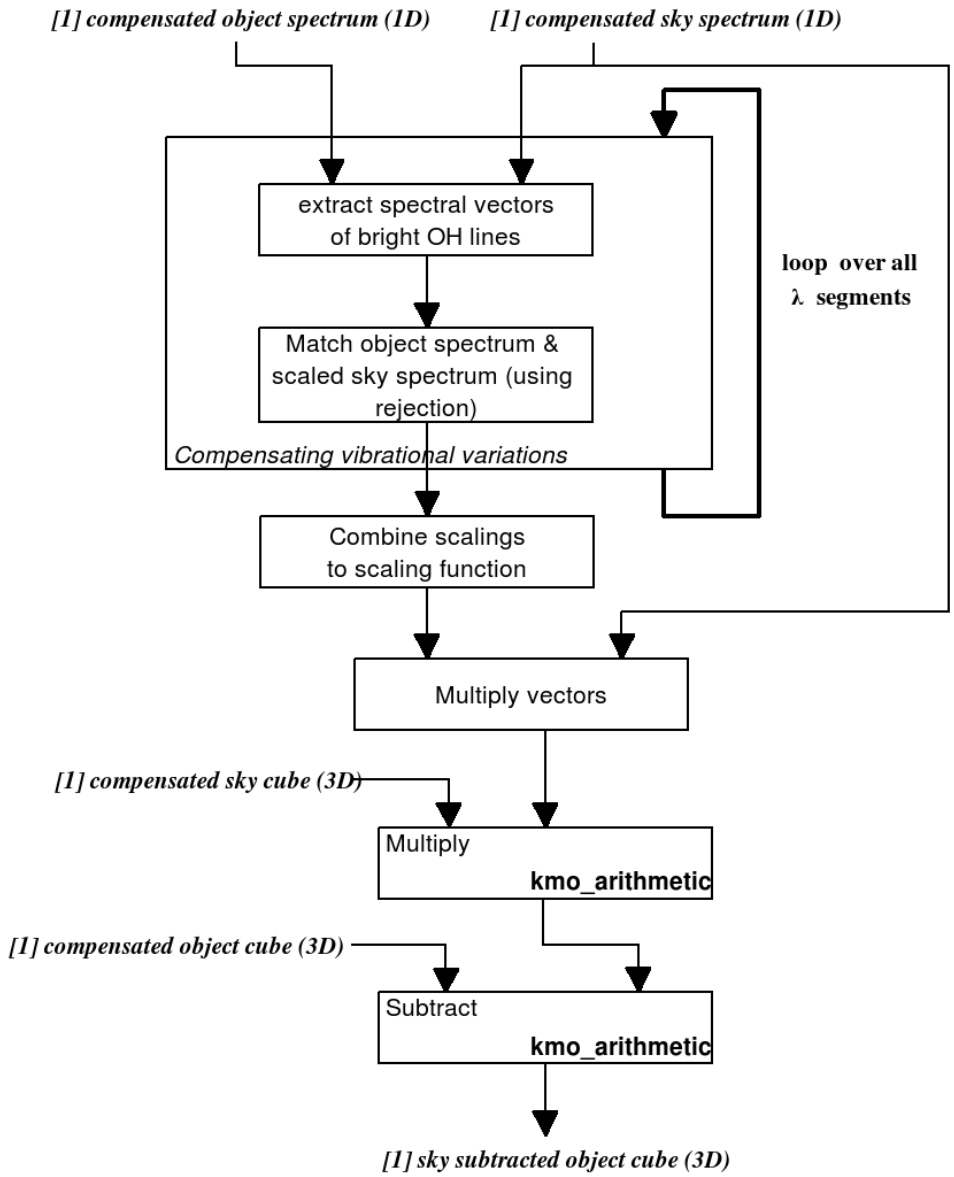

Figure 43**:** Flow chart of the recipe kmos_sky_tweak (Part 2)

The processing steps of **Figure** 43 are:

1. To correct vibrational variations, the spectra are divided into segments along the wavelength axis. For each segment the spectral vectors of bright OH lines are extracted.
2. The sky spectrum is scaled to match the object spectrum in each spectral segment.
3. The scalings of each spectral segment are combined to a single scaling function which is applied to the sky spectrum.
4. To correct rotational variations, steps 7 to 9 are repeated.
5. The two scaling functions are multiplied.
6. The resulting scaling function is multiplied with the compensated sky cube which in turn is subtracted from the compensated object cube.

7.2.5.3 Input Frames

KMOS type	DO category	Amount	Comments
F3I	CUBE_OBJECT	≥1	object cubes
F3I	CUBE_SKY	1	sky cubes

7.2.5.4 Fits Header Keywords

Primary Header

None

Sub Headers

None

7.2.5.5 Configuration Parameters

`--discard_subband`
If set to TRUE, the last sub-band in the sky tweaking will be ignored.

`--stretch`
If set to TRUE, the SKY cube is stretched (polynomial of degree `stretch_degree`, resampling method specified by `stretch_resampling`) before the sky tweaking computation. The stretching polynomial is computed by fitting the sky lines found in the OBJ cube with those found in the SKY cube.

`--stretch_degree`
The degree of the stretching polynomial.

`--stretch_resampling`
The stretching resampling method.

7.2.5.6 Output Frames

KMOS type	DO Category	Comments
F3I	OBJECT_S	Sky-corrected object cubes

7.2.5.7 Examples

```
$ esorex kmos_sky_tweak tweak.sof
                                    with reduce.sof containing:
                                    objects1.fits              CUBE_OBJECT
                                    objects2.fits              CUBE_OBJECT
                                    sky.fits                   CUBE_SKY
```

7.3 Common Utilities

7.3.1 kmo_arithmetic: Basic Arithmetic

Perform basic arithmetic on cubes.

7.3.1.1 Description

With this recipe simple arithmetic operations, like addition, subtraction, multiplication, divison and raising to a power can be performed.
Since FITS files formatted as F1I, F2I and F3I can contain data (and eventually noise) of either just one IFU or of all 24 IFUs, kmo_arithmetic behaves differently in these cases.

When the number of IFUs is the same for both operands, the first IFU of the first operand is processed with the first IFU of the second operand.
When the second operand has only one IFU while the first operand has more IFUs, then the all the IFUs of the first operand are processed individually which the IFU of the second operand.
If an operand contains noise and the other doesn't, the noise will not be processed.

Noise is only propagated if both operand contain noise extensions. If the second operator is a scalar noise is also propagated, of course.

If two cubes are given as operands, they will be combined according to the given operator.If a cube is given as first operand and an image as second, then it operates on each slice of the cube; similarly if a spectrum is given as the second operand, it operates on each spectrum of the cube; and a number as the second operand operates on each pixel of the cube.

Basic parameters:

`--operator`

Any of the following operations to perform: "+", "-", "*" or "/" (also "^" when the 2nd operator is a scalar)

`--scalar`

To be provided if a frame should be processed together with a scalar

`--file_extension`

Define a string to append to the product filename ARITHMETIC in order to get an unique filename

7.3.1.2 Flow Chart

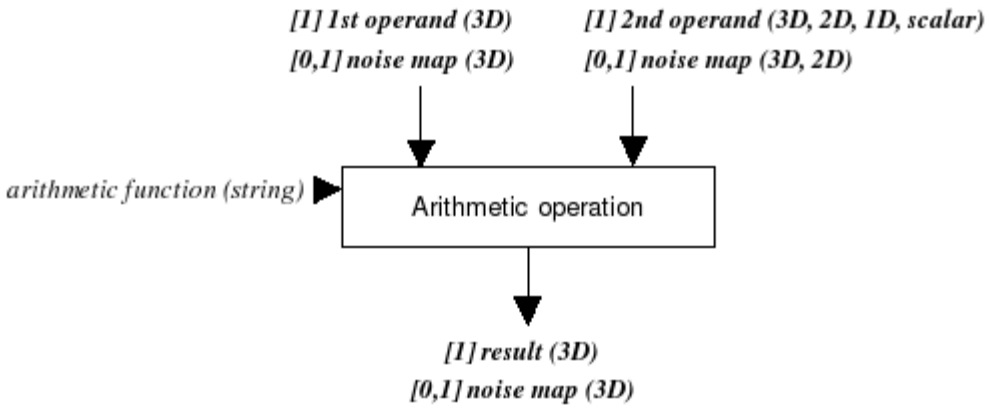

Figure 44**:** Flow chart of the recipe kmo_arithmetic

The processing steps are:

1. Two operands are combined according to the arithmetic function given (+, -, /, *).
2. The first operand is always a 3D fits frame, the second operand can have different dimensions:
    a. 3D: The cubes are combined normally as described above.
    b. 2D: The image operates on each spatial slice of the first cube.
    c. 1D: The spectrum operates on each spectrum of the first cube.
    d. scalar: The number operates on each pixel in the first cube.
3. Optionally noise maps can be provided for each operand. If done so, they will be combined according to the operation applied to the data (see also section 2.2.2).

7.3.1.3 Input Frames

KMOS type	Amount	Comments
F3I, F2I, F1I, F2D or RAW	1	data frame, with or without noise
F3I, F2I, F1I, F2D or RAW	0, 1	data frame, with or without noise

7.3.1.4 Fits Header Keywords

None specific

7.3.1.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
operator	string	"+", "-", "*", "/", "^"	""	*(mandatory)*
scalar	double	any	-DBL_MAX	*(mandatory, if only one file is supplied)*
file_extension	string	any	""	*(optional)*

7.3.1.6 Output Frames

KMOS type	Comments
F3I	1st operator is F3I and 2nd one is either F3I, F2I, F1I or scalar
F2I	1st operator is F2I and 2nd one is either F2I, F1I or scalar
F1I	1st operator is F1I and 2nd one is either F1I or scalar
F2D	1st operator is F2D and 2nd one is either F2D or scalar or 1st operator is RAW and 2nd one is either RAW or scalar

7.3.1.7 Examples

```
$ esorex kmo_arithmetic --operator="*" --scalar=9.7 F3I.fits
$ esorex kmo_arithmetic --operator="^" --scalar=9.7 F2D.fits
$ esorex kmo_arithmetic --operator="+" F3I_1.fits F3I_2.fits
```

7.3.2 kmo_copy: Copy Cube Sections

Copy a section of a cube to another cube, image or spectrum.

7.3.2.1 Description

With this recipe a specified region of an IFU-based cube (F3I), image (F2I) or vector (F1I) can be copied to a new FITS file. One can copy just a plane out of a cube (any orientation) or a vector out of an image etc. By default the operation applies to all IFUs. The input data can contain noise frames which is then copied in the same manner as the input data.
It is also possible to extract a specific IFU out of a KMOS FITS structure with 24 IFU extensions or 48 extensions if noise is present (see example in 7.3.2.7).

Basic parameters:

`--ifu`
Use this parameter to apply the operation to a specific IFU.

`--x`
`--y`
`--z`
These are the start values in each dimension. The first pixel is adressed with 1.

`--xsize`
`--ysize`
`--zsize`
These are the extents in each dimension to copy.

`--autocrop`
If set to TRUE all borders containing NaN values are cropped. Vectors will be shortened, images and cubes can get smaller. In this special case following parameters can be omitted: `--x`, `--y`, `--z`, `--xsize`, `--ysize` and `--zsize`.

7.3.2.2 Flow Chart

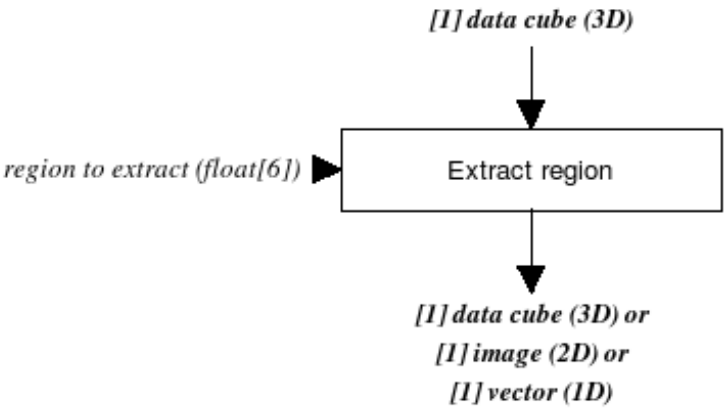

Figure 45**:** Flow chart of the recipe kmo_copy

The specified range (in all dimensions) of the input data is copied and returned. If the specified ranges in one or two dimensions are reduced to a single value, then an image or a vector will be returned, respectively.

7.3.2.3 Input Frames

KMOS type	Amount	Comments
F3I orF2I or F1I	1	data frame, with or without noise

7.3.2.4 Fits Header Keywords

None specific

7.3.2.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
ifu	int	1 ≤ ifu ≤ NEXTEND	-1	*optional* If ifu is specified, the recipe operates only on the specified IFU.
x, y, z	int	1 ≤ x ≤ NAXIS1 1 ≤ y ≤ NAXIS2 1 ≤ z ≤ NAXIS3	1	*(mandatory if autocrop isn't set)*
xsize, *ysize,* *zsize*	int	1 < xsize ≤ NAXIS1-x 1 < ysize ≤ NAXIS2-y 1 < zsize ≤ NAXIS3-z	1	*(optional)* If one or more of these are omitted, a plane, a vector or a scalar is extracted. A scalar is returned in a vector of size 1.
autocrop	bool	TRUE, FALSE	FALSE	*optional* If set to TRUE, *x, y, z, xsize, ysize* and *zsize* are ignored.

7.3.2.6 Output Frames

KMOS type	Comments
F3I	for F3I as input and x, y, z, xsize, ysize, zsize defined

KMOS type	Comments
F2I	for F3I as input and x, y, z, xsize, ysize defined or x, y, z, xsize, zsize defined or x, y, z, ysize, zsize defined for F2I as input and x, y, xsize, ysize defined

KMOS type	Comments

F1I	for F3I as input and x, y, z, xsize defined or x, y, z, ysize defined or x, y, z, zsize defined or **x, y, z defined (vector of size 1)** for F2I as input and x, y, xsize defined or x, y, ysize defined for F1I as input and x, xsize defined or x defined (vector of size 1)

7.3.2.7 Examples

extract cube:

```
$ esorex kmo_copy --x=3 --y=2 --z=1 --xsize=2 --ysize=3
                --zsize=6 F3I.fits
```

extract plane:

```
$ esorex kmo_copy --x=3 --y=2 --z=1 --xsize=2 --ysize=3 F3I.fits
```

extract vector just of IFU 4:

```
$ esorex kmo_copy --x=3 --y=2 --z=1 --ysize=3 –ifu=4 F3I.fits
```

extract whole IFU 4:

```
$ esorex kmo_copy --x=1 --y=1 --z=1 --xsize=<NAXIS1>
           --ysize=<NAXIS2> --zsize=<NAXIS3> --ifu=4 F3I.fits
```

extract scalar:

```
$ esorex kmo_copy --x=3 --y=2 --z=1 F3I.fits
```

autocrop:

```
$ esorex kmo_copy --autocrop=TRUE --ifu=8 F3I.fits
```

7.3.3 kmos_extract_spec: Extracting Spectra

Extract a spectrum from a cube.

7.3.3.1 Description

This recipe extracts a spectrum from a datacube. The datacube must be in F3I KMOS FITS format (either with or without noise). The output will be a similarly formatted F1I KMOS FITS file.

Basic parameters:

`--mask_method`

There are several ways to define the region to consider for spectrum calculation:

- **integrated** ***(default)***
  A circular mask with defined centre and radius is created (`--centre` and `--radius` have to be defined). This mask is applied to all extensions.
- **mask**
  An arbitrary mask can be provided (for example the mask created by kmo_sky_mask can be used). The mask must be in F2I KMOS FITS format, mustn't contain noise and must have as many extensions as the input cube. The mask can be binary as well as it can contain float values, so a weighted mask is also possible. (0: pixels is ignored, 1: pixel is included) The mask must be of the same size that the input datacube.
- **optimal**
  The mask is created automatically by fitting a normalised profile (using kmo_fit_profile) to the image of the datacube (using kmo_make_image the datacube is summed up in spectral direction according to the specified `--cmethod`). This profile is then used as mask input. When `--save_mask` is set to true the mask is saved on disk. The remaining parameters not described here apply to the fitting of the profile.

If the spectra of several objects in a IFU should be extracted, `--mask_method`="mask" is recommended. With several calls to kmos_extract_spec using different masks all spectra can be extracted.

Advanced parameters:

`--centre`

`--radius`

see `--mask_method` = "integrated"

`--save_mask`

see `--mask_method` = "optimal"

`--cmethod`

Applies only if `–mask_method` = "optimal"

Following methods of frame combination are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:

  ```
  val > mean + stdev * cpos_rej
  ```

  and

  ```
  val < mean - stdev * cneg_rej
  ```

  where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2).

- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.
- **sum**
  At each pixel position the sum is calculated.
- **min_max**
  The specified number of minimum and maximum pixel values will be rejected.
  `--cmax` and `--cmin` apply to this method.

`--cpos_rej`
`--cneg_rej`
`--citer`
see `--cmethod` = ”ksigma”

`--cmax`
`--cmin`
see `--cmethod` = “min_max”

7.3.3.2 Flow Chart

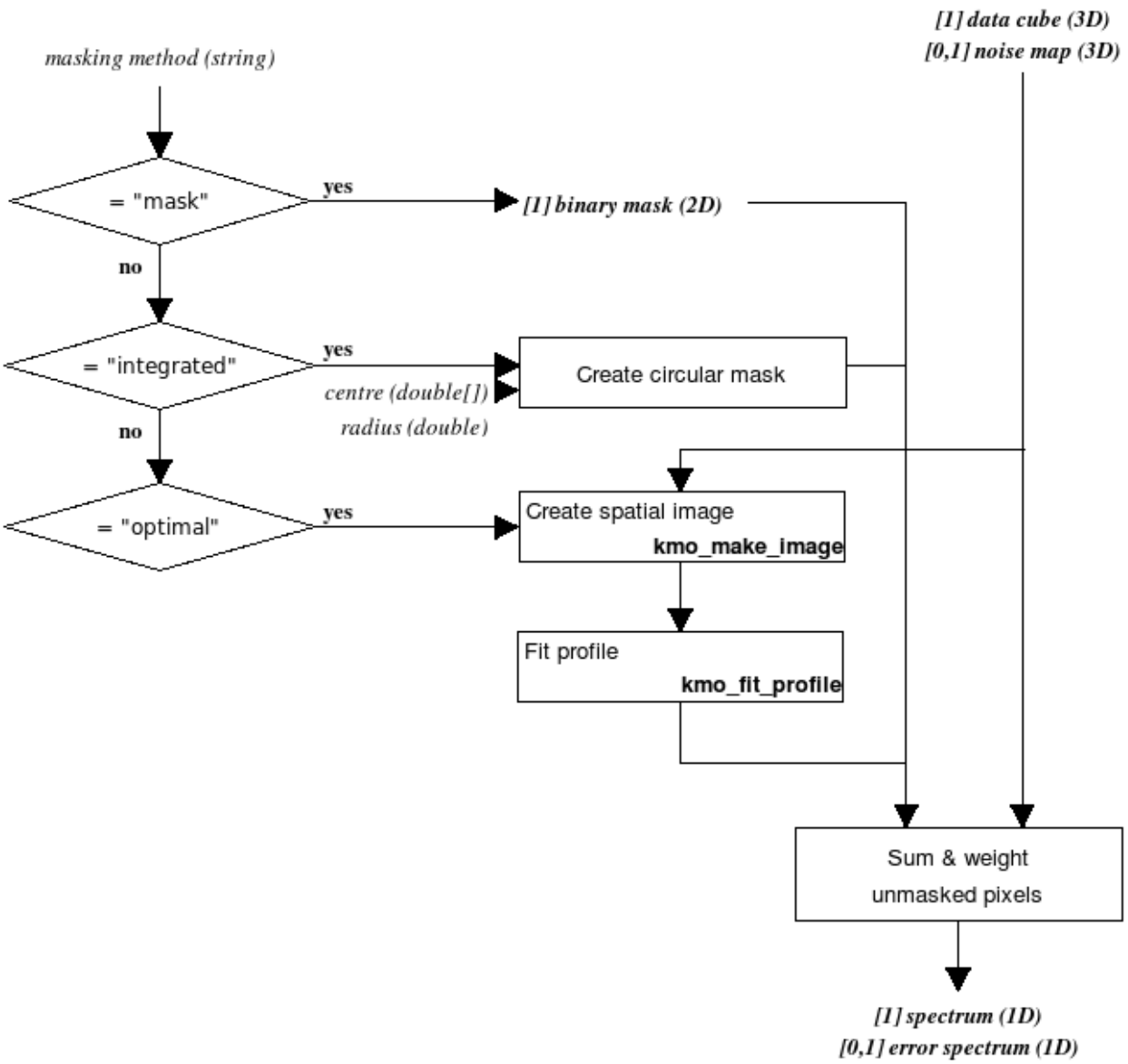

Figure 46**:** Flow chart of the recipe kmos_extract_spec

The processing steps are:

1. A mask is generated (or taken as input) where sky is 0.0 and object is 1.0:
    a. "optimal" method
        I. The data cube is collapsed using kmo_make_image.
        II. From the resulting image the signal to noise, based on a Gaussian fit using kmo_fit_profile, is estimated.
        III. The fit will be scaled in a way that the maximum value equals one. The result is a mask with float values.
    b. "integrated" method
        I. A binary mask with specified centre and radius is defined.
    c. "mask" method
        I. The binary input mask is taken.
2. All unmasked pixels in each spatial slice are summed and weighted all along the spectral axis.
3. An optional noise map is masked the same way as the input data and combined as described in Sect. 2.2.2.

4. If there are several objects in a single cube, their spectra can be extracted separately using different masks.

7.3.3.3 Input Frames

KMOS type	Amount	Comments
F3I	1	cube with or without noise
F2I	0 or 1	*(optional, applies only when --mask_method =”mask”)*

This recipe also accepts also a path to a FITS file instead of a sof-file.

7.3.3.4 Fits Header Keywords

None specific

7.3.3.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
mask_method	string	“optimal” “integrated” “mask”	“integrated”	*(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
centre	double[2]	[0 ≤ x ≤ NAXIS1, 0 ≤ y ≤ NAXIS2]	[7.5,7.5]	The centre of the circular mask [pixel] *(mandatory, if --mask_method = ”integrated”)*
radius	double	radius ≥ 0	3.0	The radius of the circular mask [pixel] *(mandatory, if --mask_method = ”integrated”)*
save_mask	bool	true false	false	True if the calculated mask should be saved. *(optional, applies only when --mask_method = “optimal”)*
cmethod	string	“ksigma” “min_max” “average” “median” “sum”	“ksigma”	The averaging method to apply *(optional)*
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = “ksigma”)*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only when --cmethod = “ksigma”)*
cmax cmin	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip

				with min/max-clipping *(optional, applies only when --cmethod = "min_max")*

7.3.3.6 Output Frames

KMOS type	PRO Category	Comments
F1I	EXTRACT_SPEC	Extracted spectrum
F2I	EXTRACT_SPEC_MASK	The calculated mask *(optional, if --mask_method="optimal" and --save_mask=true)*

7.3.3.7 Examples

```
$ esorex kmos_extract_spec --mask_method="integrated"
                        --centre="3.0,4.5" --radius=4 cube.fits
$ esorex kmos_extract_spec --mask_method=optimal --save_mask cube.fits
$ cp extract_spec_mask.fits extract_spec_mask_tmp.fits
$ esorex kmos_extract_spec --mask_method="mask" cube.fits
                                      extract_spec_mask_tmp.fits
```

IMPORTANT : Note that the passed mask name must be different than extract_spec_mask.fits because this is precisely the name of the produced mask. If used, it would be overwritten during the recipe execution and cause an error.

Note also that the method "mask" currently only works on 24 extensions reconstructed cubes, and not on a combined cube. This will be addressed in a coming release.

7.3.4 kmo_fit_profile: Fitting Spectral and Spatial Profiles

Fit spectral line profiles as well as spatial profiles with a simple function - for example to measure resolution or find the centre of a source.

7.3.4.1 Description

This recipe creates either spectral or spatial profiles of sources using different functions to fit. Spectral profiles can be created for F1I frames (if WCS is defined in the input frame, the output parameters are in respect to the defined WCS).
Spatial profiles can be created for F2I frames (any WCS information is ignored here).
If the frames contain no noise information, constant noise is assumed for the fitting procedure.

Basic parameters:
`--method`
F1I frames can be fitted using either "gauss", "moffat" or "lorentz" function.
F2I frames can be fitted using either "gauss" or "moffat" function.

Advanced parameters:
`--range`
For F1I frames the spectral range can be defined. With available WCS information the range can be provided in units (e.g. “1.2;1.5”), otherwise in pixels (e.g. “112;224).
For F2I frames the spatial range can be defined as follow: “x1,x2;y1,y2”

7.3.4.2 Flow Chart

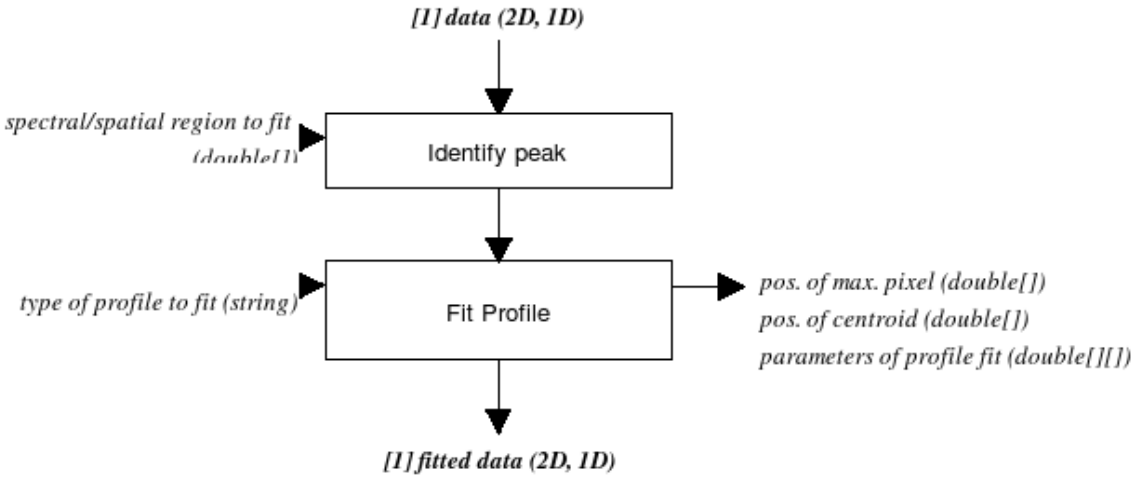

Figure 47**:** Flow chart of the recipe kmo_fit_profile

The processing steps are:

1. The region to fit is defined by the spectral (1D) or spatial (2D) interval provided. In this interval, the peak is identified.
2. Then a function is fitted to the interval according to a defined profile (Gaussian, Moffat, Lorentzian). Output parameters are the position (either lambda-position or pixel number depending if WCS data is provided in the headers of the input data frames) of the maximum pixel, the position of the centroid and the parameters of the function fit.

7.3.4.3 Input Frames

KMOS type	Amount	Comments
F1I or F2I	1	data frame, with or without noise

This recipe also accepts also a path to a FITS file instead of a sof-file.

7.3.4.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	Int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
CRPIX1	double	any	*(optional for F1I frames)*
CRVAL1	double	any	*(optional for F1I frames)*
CDELT1	double	any	*(optional for F1I frames)*

7.3.4.5 Configuration Parameters

Name	Type	valid values	Default	Comments
method	string	"gauss", "moffat", "lorentz"	"gauss"	*(optional,* *"lorentz" applies only to F1I frames)*
range	string	"x1,x2" (for F1I) or "x1,x2; y1,y2" (for F2I)	""	F1I frames with WCS: values are in microns F1I frames without WCS: values denote pixel positions (zero based). F2I frames: values denote pixel positions (base 1 for images, FITS convention) *(optional,* *default is the whole range)*

7.3.4.6 Output Frames

KMOS type	PRO Category	Comments
F1I or F2I	FIT_PROFILE	Fitted 1D-profile or Fitted 2D-profile *(in both cases without noise)*

7.3.4.7 Examples

```
$ esorex kmo_fit_profile f1i_with_noise.fits
```

7.3.5 kmo_make_image: Making Images

Collapse a cube to create a spatial image.

7.3.5.1 Description

This recipe collapses a cube along the spectral axis using rejection. By default all spectral slices are averaged.
Errors are propagated for the same spectral ranges as for the input data if a noise map is provided.

Basic parameters:

`--range`

The spectral range can be delimited to one or several sub-ranges like "1.8,1.9" or "1.8,1.9; 2.0,2.11"

`--cmethod`

Following methods of collapsing a cube are available:

- **ksigma** ***(default)***
  An iterative sigma clipping. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:
  ```
  val > mean + stdev * cpos_rej
  ```
  and
  ```
  val < mean - stdev * cneg_rej
  ```
  where `--cpos_rej`, `--cneg_rej` and `--citer` are the corresponding configuration parameters. In the first iteration median and percentile level are used (See Sec. 8.2).
- **median**
  At each pixel position the median is calculated.
- **average**
  At each pixel position the average is calculated.
- **sum**
  At each pixel position the sum is calculated.
- **min_max**
  The specified number of minimum and maximum pixel values will be rejected.
  `--cmax` and `--cmin` apply to this method.

Advanced parameters:

`--threshold`

Optionally an OH spectrum can be provided. In this case a threshold can be defined. The wavelengths of values above the threshold level in the OH spectrum are omitted in the input frame. This parameter can be combined with the `--range` parameter. Negative threshold values are ignored.

`--cpos_rej`
`--cneg_rej`
`--citer`

see `--cmethod` = "ksigma"

`--cmax`
`--cmin`

see `--cmethod` = "min_max"

7.3.5.2 Flow Chart

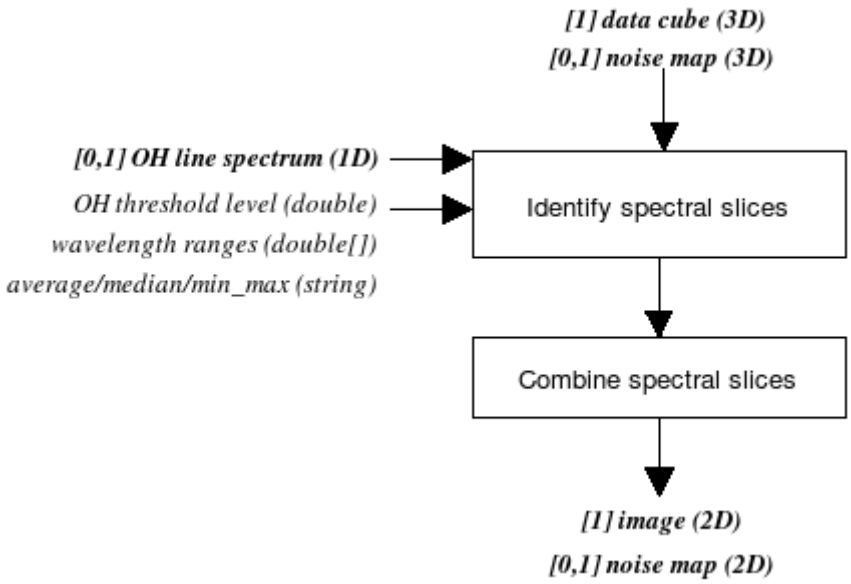

Figure 48**:** Flow chart of the recipe kmo_make_image

The processing steps are:

1. If a OH line spectrum is provided, the spectral slices which are to be combined are identified according to the threshold level and the wavelength ranges applied to the spectrum (i.e. if the wavelength of the spectral slice lies in between a predefined range or above the threshold level in the OH line spectrum, it is omitted).
2. The identified spectral slices are averaged to create a spatial image (Either applying a median or averaging using rejection or min_max rejecting a predefined number of max- and min-values).
3. Optionally a noise map matching the data cube can be provided, it will be combined along the same spectral ranges as defined above (see also section 2.2.2) and output as a 2d noise map.

7.3.5.3 Input Frames

KMOS type	Amount	Comments
F3I	1	data frame, with or without noise
F1S	0 or 1	the OH line spectrum. *(opt)*

This recipe also accepts also a path to a FITS file instead of a sof-file.

7.3.5.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
CRPIX3	double	any	
CRVAL3	double	any	
CDELT3	double	any	

7.3.5.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
range	string	"start1,end1;start2,end2;…"	""	The spectral ranges to combine *(optional, applies only if a OH-spectrum is provided)*
threshold	double	any, if *threshold* < 0 then no thresholding is applied	0.1	The OH threshold level *(optional, applies only if a OH-spectrum is provided)*
cmethod	string	"ksigma" "min_max" "average" "median" "sum"	"ksigma"	The averaging method to apply *(optional)*
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional, applies only when --cmethod = "ksigma")*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional, applies only when --cmethod = "ksigma")*
cmax cmin	int	cmax ≥ 0 cmin ≥ 0	1 1	The number of maximum and minimum pixel values to clip with min/max-clipping *(optional, applies only when --cmethod = "min_max")*

7.3.5.6 Output Frames

KMOS type	PRO Category	Comments
F2I	MAKE_IMAGE	Collapsed data cubes

7.3.5.7 Examples

```
$ esorex kmo_make_image data.fits
$ esorex kmo_make_image data_noise.fits
$ esorex kmo_make_image --cmethod="median" data_noise.fits
$ esorex kmo_make_image --cmethod="average" --cpos_rej =2.2
                        --cneg_rej=1.7 --citer=2 data_noise.fits
$ esorex kmo_make_image --method="min_max" --cmax=20 --cmin=10
                          data_noise.fits
$ esorex kmo_make_image data.fits oh_spec.fits
$ esorex kmo_make_image --range="1.8,1.9;2.0,2.1"
                            data.fits oh_spec.fits
```

7.3.6 kmo_noise_map: Noise Estimation

Generate a noise map from a raw frame.

7.3.6.1 Description

The noise in each pixel of the input data is estimated using gain and readnoise. The readnoise is expected to be in the primary header (ESO DET CHIP RON), the gain (ESO DET CHIP GAIN) has to be in each of the subsequent headers of each detector frame. The output is the initial noise map of the data frame.

7.3.6.2 Flow Chart

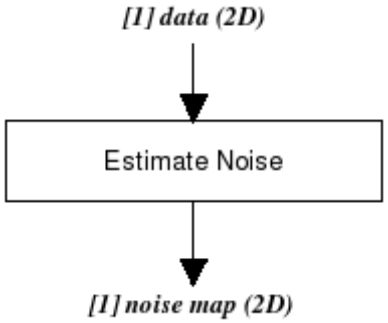

Figure 49**:** Flow chart of the recipe kmo_noise_map

The noise in each pixel of the input data is estimated according to the method described in Sect. 2.2.1. The output is the initial noise map of the data frame.

7.3.6.3 Input Frames

KMOS type	Amount	Comments
RAW	1	raw data frame

This recipe also accepts also a path to a FITS file instead of a sof-file.

7.3.6.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	Int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
ESO DET CHIP GAIN	double	any	
ESO DET CHIP RON	double	any	

7.3.6.5 Configuration Parameters

None

7.3.6.6 Output Frames

KMOS type	PRO Category	Comments
F2D	NOISE_MAP	Initial noise map

7.3.6.7 Examples

```
$ esorex kmo_noise_map RAW.fits
```

7.3.7 kmo_rotate: Rotating a Cube

Rotate a cube spatially.

7.3.7.1 Description

This recipe rotates a cube spatially (CCW). If the rotation angle isn't a multiple of 90 degrees, the output cube will be interpolated and get larger accordingly.
By default all IFUs will be rotated.

Basic parameters:

`--rotations`

This parameter must be supplied. It contains the amount of rotation to apply. The unit is in degrees. If it contains one value (e.g. "3.5") all IFUs are rotated by the same amount. Positive angels rotate counterclockwise. If 24 values are supplied each IFU is rotated individually (e.g. "2.3;15.7;…;-3.3").

`--imethod`

The interpolation method to apply when rotating an angle not being a multiple of 90. There are two methods available:

- BCS: Bicubic spline
- NN: Nearest Neighbor (currently disabled)

`--ifu`

If a single IFU should be rotated, it can be defined using the `--ifu` parameter (`--rotations` parameter contains only one value).

Advanced parameters:

`--flux`

Specify if flux conservation should be applied.

`--extrapolate`

By default the output frame grows when rotating an angle not being a multiple of 90. In this case none of the input data is lost. When it is desired to keep the same size as the input frame this parameter can be set to TRUE and the data will be clipped.

7.3.7.2 Flow Chart

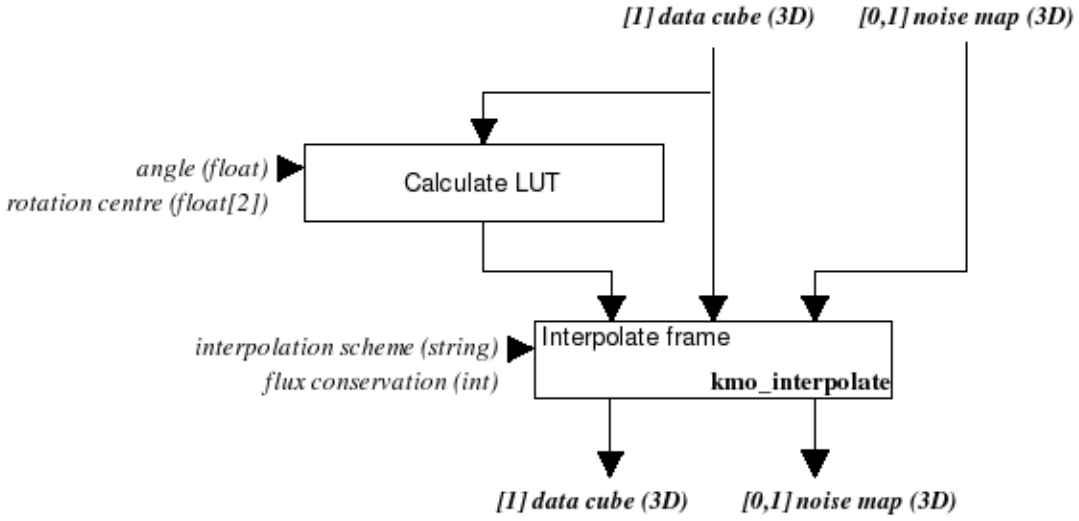

Figure 50**:** Flow chart of the recipe kmo_rotate

The processing steps are:

1. First the LUT representing the spatial rotation is calculated.
2. Then the data cube and the optional noise map are interpolated according the LUT.

Additionally the interpolation scheme can be chosen and if flux conservation should be applied.

7.3.7.3 Input Frames

KMOS type	Amount	Comments
F3I	1	data frame, with or without noise

This recipe also accepts also a path to a FITS file instead of a sof-fil

7.3.7.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	Int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
CRPIX1, CRPIX2	double	any	
CRVAL1, CRVAL2	double	any	
CDELT1, CDELT2	double	any	
CD-Matrix	double	any	

7.3.7.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
rotations	string	string with 1 or 24 elements [degrees] e.g. “2.3;15.7;…;-3.3”	“”	The rotations for all specified IFUs *(mandatory)*
imethod	string	“BCS” “NN”	“BCS”	Interpolation method: BCS: Bicubic spline NN: Nearest Neighbor *(optional, applies only when rotation angle isn’t a multiple of 90 degrees)*
ifu	int	24 ≥ ifu ≥ 0	0	The ifu to rotate. 0 rotates all ifus the same amount *(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
flux	bool	TRUE, FALSE	FALSE	Apply flux

				conservation
extrapolate	bool	TRUE, FALSE	FALSE	FALSE: Output frame will be larger than the input TRUE: Output and input frame have the same size, data will be clipped *(optional, applies only when rotation angle isn't a multiple of 90 degrees*

7.3.7.6 Output Frames

KMOS type	PRO Category	Comments
F3I	ROTATE	Rotated cube

7.3.7.7 Examples

```
$ esorex kmo_rotate --ifu=8 --rotations="93.87" data.fits
$ esorex kmo_rotate -rotations="1.1;3.8;-4.5;……;18,9" data.fits
```

7.3.8 kmo_shift: Translating a Cube

Shift a cube spatially.

7.3.8.1 Description

This recipe shifts a cube spatially. A positive x-shift shifts the data to the left, a positive y-shift shifts upwards, where a shift of one pixel equals 0.2 arcsec. The output will still have the same dimensions, but the borders will be filled with NaNs accordingly.
To adjust only the WCS without moving the data the `--wcs-only` parameter has to be set to TRUE. The WCS is updated in the same way as if the data would have moved as well. This means that the point at (x,y) has the same coordinates as the point (x+1,y+1) after updating the WCS (the WCS moved in the opposite direction).

Basic parameters:

`--shifts`

This parameter must be supplied. It contains the amount of shift to apply. The unit is in arcsec. If the `--shifts` parameter contains only two values (x,y), all IFUs will be shifted by the same amount. If it contains 48 values (x1,y1;x2,y2;...;x24,y24), the IFUs are shifted individually.

`--imethod`

The interpolation method to apply when the shift value isn't a multiple of the pixel scale. There are two methods available:

- BCS: Bicubic spline
- NN: Nearest Neighbor

`--ifu`

If a single IFU should be shifted, it can be defined using the `--ifu` parameter (`--shifts` parameter contains only two values).

Advanced parameters:

`--flux`

Specify if flux conservation should be applied when applying a subpixel shift.

`--extrapolate`

By default no extrapolation is applied. At the borders NaN values are introduced. When choosing "BCS" as interpolation method and applying a sub-pixel shift, extrapolation can be switched on.

`--wcs-only`

By default data and WCS are shifted in sync. If this parameter is set to TRUE only the WCS is updated (i.e. if someone thinks that the IFU isn't pointing exactly to the correct coordinates).

7.3.8.2 Flow Chart

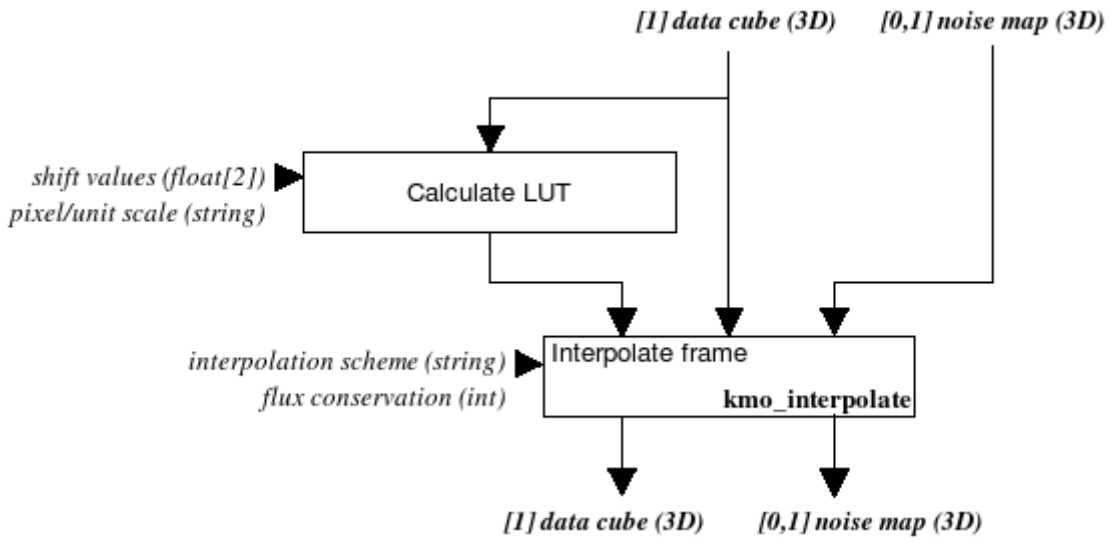

Figure 51**:** Flow chart of the recipe kmo_shift

The processing steps are:

1. First the LUT representing the shift is calculated.
2. Then the data cube and the optional noise map are interpolated according the LUT. Additionally the interpolation scheme can be chosen and if flux conservation should be applied.

7.3.8.3 Input Frames

KMOS type	Amount	Comments
F3I	1	data frame, with or without noise

This recipe also accepts also a path to a FITS file instead of a sof-fil

7.3.8.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	Int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
CRPIX1, CRPIX2	double	any	
CRVAL1, CRVAL2	double	any	
CDELT1, CDELT2	double	any	
CD-Matrix	double	any	

7.3.8.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
shifts	string	string with 2 or 48 elements [arcsec] e.g. [x1, y1; x2, y2;…]	“”	The shifts for each spatial dimension for all specified IFUs *(mandatory)*
imethod	string	“BCS” “NN”	“BCS”	Interpolation method: BCS: Bicubic spline NN: Nearest Neighbor *(optional, applies only when the shift isn’t a multiple of the pixel scale)*
ifu	int	24 ≥ ifu ≥ 0	0	The ifu to shift. 0 shifts all ifus the same amount *(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
flux	bool	TRUE, FALSE	FALSE	Apply flux conservation *(optional)*
extrapolate	bool	TRUE, FALSE	FALSE	FALSE: shifted IFU will be filled with NaNs at the borders TRUE: shifted IFU will be extrapolated at the borders *(optional, applies only when method=BCS and doing sub pixel shifts)*
wcs-only	bool	TRUE, FALSE	FALSE	FALSE: data and WCS are shifted together TRUE: only the WCS is shifted

7.3.8.6 Output Frames

KMOS type	PRO Category	Comments
F3I	SHIFT	Shifted cube

7.3.8.7 Examples

```
$ esorex kmo_shift --ifu=8 --shifts=”0.2,0.11” data.fits
$ esorex kmo_shift -shifts=”0.4,0.2;-0.01,-0.09;……;0.1;0.1” data.fits
```

7.3.9 kmo_sky_mask: Creating a Mask of Sky Pixels

Create a mask of spatial pixels that indicates which pixels can be considered as sky.

7.3.9.1 Description

This recipes calculates masks of the skies surrounding the objects in the different IFUs of a reconstructed F3I frame. In the resulting mask pixels belonging to objects have value 1 and sky pixels have value 0.

The noise and the background level of the input data cube are estimated using the mode calculated in kmo_stats. If the results aren't satisfactory, try changing `--cpos_rej` and `--cneg_rej`. Then pixels are flagged in the data cube which have a value less than the mode plus twice the noise (val < mode + 2*sigma). For each spatial pixel the fraction of flagged pixels in its spectral channel is determined.
Spatial pixels are selected where the fraction of flagged spectral pixels is greater than 0.95 (corresponding to the 2*sigma above).

The input cube can contain noise extensions, but they will be ignored. The output doesn't contain noise extensions.

Basic parameters:
`--fraction`
The fraction of pixels that have to be greater than the threshold can be defined with this parameter (value must be between 0 and 1).

`--range`
If required, a limited wavelength range can be defined (e.g. "1.8,2.1).

Advanced parameters:
`--cpos_rej`
`--cpos_rej`
`--citer`
An iterative sigma clipping is applied in order to calculate the mode (using kmo_stats). For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:

```
val > mean + stdev * cpos_rej
```

and

```
val < mean - stdev * cneg_rej
```

In the first iteration median and percentile level are used.

7.3.9.2 Flow Chart

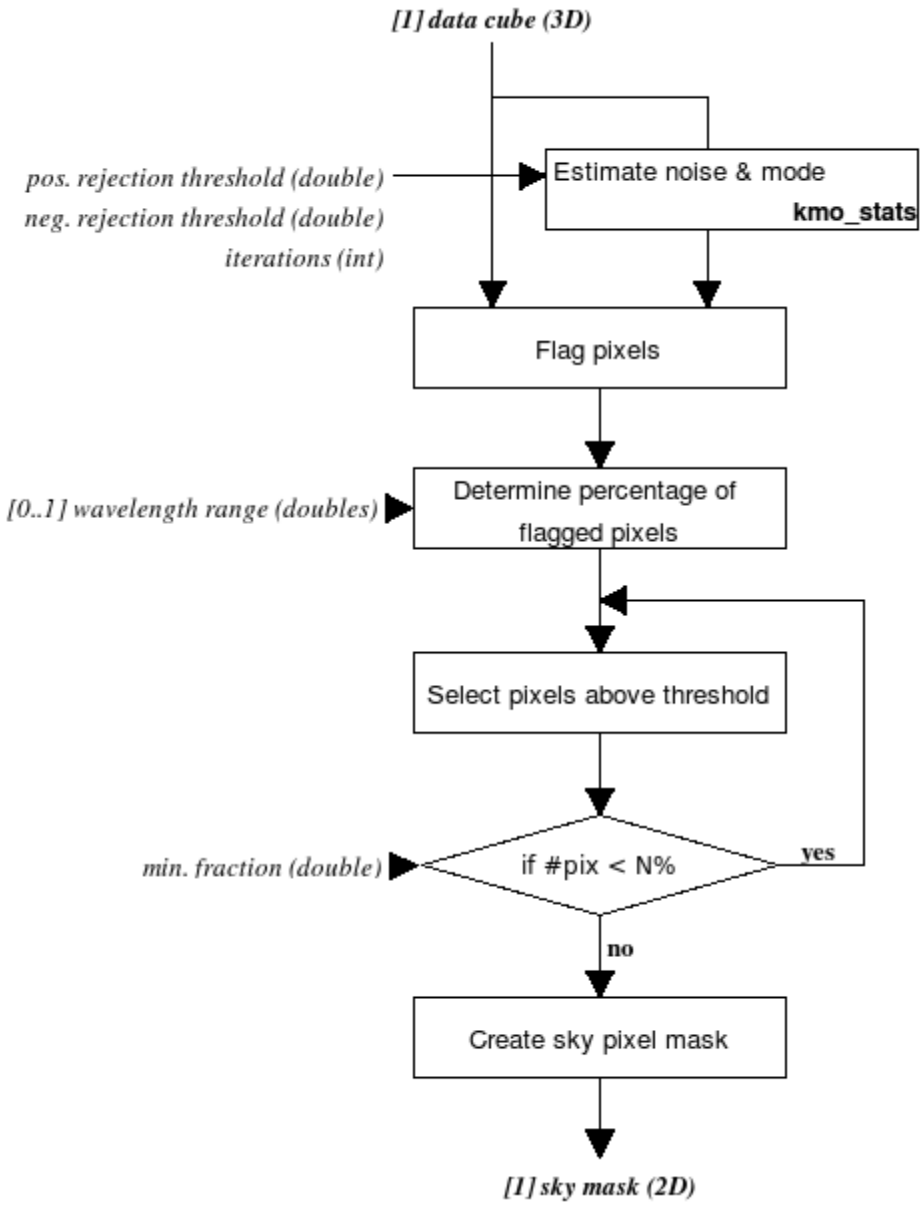

Figure 52**:** Flow chart of the recipe kmo_sky_mask

The processing steps are:

1. The noise and the background level (mode) of the input data cube are estimated. Note that although the noise varies with wavelength, a single estimate of the noise is sufficient for the purpose here.
2. Flag pixels in the data cube which have a value less than the mode plus twice the noise (val < mode + 2σ)
3. For each spatial pixel the fraction of flagged pixels in its spectral channels is determined. If required, a limited wavelength range can be provided for this step.
4. Spatial pixels are selected where the fraction of flagged spectral pixels is greater than 0.95 (corresponding to the 2σ above)
5. If less than a specified percentage of spatial pixels are included, then increase the selection to include this many.

Create a mask indicating 'sky' pixels (sky = 0, object = 1).

7.3.9.3 Input Frames

KMOS type	Amount	Comments
F3I	1	one reconstructed frame

This recipe also accepts also a path to a FITS file instead of a sof-file.

7.3.9.4 Fits Header Keywords

Primary Header

Keyword	Type	Value	Comments
MINDIT	double	~2.5	Estimated value
NDIT	Int	1	
EXPTIME	double	any	

Sub Headers

Keyword	Type	Value	Comments
CRPIX3	double	any	
CRVAL3	double	any	
CDELT3	double	any	

7.3.9.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
fraction	double	$1.0 \geq fraction \geq 0.0$	0.95	Minimum fraction of spatial pixels to select as sky *(optional)*
range	string	"start,end"	""	Min & max spectral range to use in sky pixel determination (microns) *(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional)*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional)*

7.3.9.6 Output Frames

KMOS type	PRO Category	Comments
F2I	SKY_MASK	The sky mask frame

7.3.9.7 Examples

```
$ esorex kmo_sky_mask f3i.fits
$ esorex kmo_sky_mask –fraction=0.6 f3i.fits
$ esorex kmo_sky_mask --range=”1.8,1.9” f3i.fits
```

7.3.10 kmo_stats: Basic Statistics

Perform basic statistics on a KMOS-conform fits-file.

7.3.10.1 Description

This recipe performs basic statistics on KMOS-conform data-frames of type F2D, F1I, F2I and F3I either with or without noise and RAW. Optionally a 2D mask can be provided to define a region on which the statistics should be calculated on (mask 0: exclude pixel, mask 1: include pixel). A mask can't be provided for statistics on F1I frames.
The output is stored in a vector of length 11. The vector represents following values:

1. Number of pixels
2. Number of finite pixels
3. Mean
4. Standard Deviation
5. Mean with iterative rejection (i.e. mean & sigma are calculated iteratively, each time rejecting pixels more than +/-N sigma from the mean)
6. Standard Deviation with iterative rejection
7. Median
8. Mode (i.e. the peak in a histogram of pixel values)
9. Noise (a robust estimate given by the standard deviation from the negative side of the histogram of pixel values)
10. Minimum
11. Maximum

The same numerical operations are applied to the noise as with the data itself.

Basic parameters:

`--ext`

These parameters specify with extensions to process. The value 0, which is default, calulates all extensions.

Advanced parameters:

`--cpos_rej`
`--cpos_rej`
`--citer`

An iterative sigma clipping is applied in order to calculate the mode. For each position all pixels in the spectrum are examined. If they deviate significantly, they will be rejected according to the conditions:

```
val > mean + stdev * cpos_rej
```

and

```
val < mean - stdev * cneg_rej
```

In the first iteration median and percentile level are used.

7.3.10.2 Flow Chart

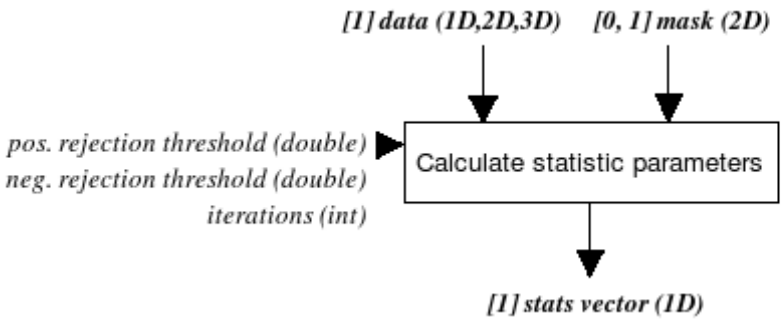

Figure 53**:** Flow chart of the recipe kmo_ stats

The input data and an optional mask (2D) are taken as inputs and a vector of length 11 is returned as output.

7.3.10.3 Input Frames

KMOS type	Amount	Comments
F3I,F2I,F1I,F2D,B2D,RAW	1	one frame, with or without noise
F2I,F2D,B2D,RAW	0 or 1	mask *(optional)*

This recipe also accepts also a path to a FITS file instead of a sof-file.

7.3.10.4 Fits Header Keywords

None

7.3.10.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
ext	int	ext ≥ 0	0	Specifies which extensions to calculate. 0 calculates them all *(optional)*

Advanced parameters

Name	Type	valid values	Default	Comments
cpos_rej cneg_rej	double	cpos_rej ≥ 0, cneg_rej ≥ 0	3.0 3.0	The positive and negative rejection thresholds for bad pixels *(optional)*
citer	int	citer ≥ 1	3	The number of iterations for kappa-sigma-clipping. *(optional)*
ifu	int	ifu ≥ 0	0	Specifies which extensions to calculate. 0 calculates them all *(optional, applies only for F1I, F2I and F3I frames)*

det	int	det ≥ 0	0	Specifies which extensions to calculate. 0 calculates them all *(optional, applies only for F2D and RAW frames)*

7.3.10.6 Output Frames

KMOS type	DO Category	Comments
F1I	STATS	The calculated statistics parameters

7.3.10.7 Examples

```
$ esorex kmo_stats F3I.fits
$ esorex kmo_stats –ext=1 F3I.fits F2I.fits
```

7.3.11 kmo_fits_strip: Stripping FITS files

Strip noise, rotator and/or empty extensions from a processed KMOS fits frame.

7.3.11.1 Description

With this recipe KMOS fits frames can be stripped in following way:

Basic parameters:
--noise
All noise extensions will be removed. Only the data extensions remain.

--angle
Applies only to calibration products from kmos_flat and kmos_wave_cal.
All extensions matching provided angle are kept, the others are removed.
Supply a single integer value.

--empty
All empty extensions will be removed.

--extension
Supply a comma-separated string with integer values indicating the extensions to keep. The other extensions are removed (any data or noise information is disregarded, the values are interpreted absolutely)

The parameters --noise, --angle and --empty can be combined.
When --extension is specified, all other parameters are ignored.
When no parameter is provided, no output will be generated.

7.3.11.2 Flow Chart

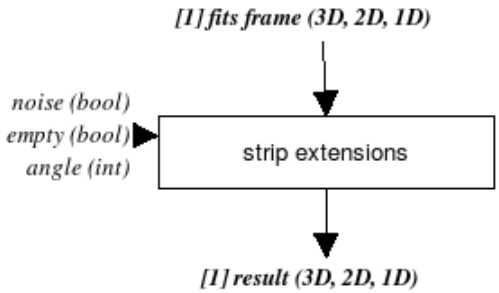

Figure 54**:** Flow chart of the recipe kmo_ fits_strip

7.3.11.3 Input Frames

KMOS type	Amount	Comments
F3I,F2I,F1I,F2D	1	one frame, with or without noise

This recipe also accepts also a path to a FITS file instead of a sof-file.

7.3.11.4 Fits Header Keywords

None

7.3.11.5 Configuration Parameters

Basic parameters

Name	Type	valid values	Default	Comments
noise	bool	TRUE, FALSE	FALSE	If set to TRUE, all noise extensions are stripped *(optional)*
angle	int	angle >=0 and angle < 360	-1	Aplies to calibration frames with several rotatr angles included. If set to a valid value only the specified rotator angle is kept, the others are removed
empty	bool	TRUE, FALSE	FALSE	If set to TRUE, all empty extensions are stripped *(optional)*
extension	string	e.g. “1,4,6”	“”	IFU IDs provided mustn’t be bigger than the actual number of extensions *(optional)*

7.3.11.6 Output Frames

KMOS type	PRO Category	Comments
F3I,F2I,F1I,F2D	STRIP	The stripped frame

7.3.11.7 Examples

```
$ esorex kmo_fits_strip --noise F3I_data_noise.fits
$ esorex kmo_fits_strip --empty F3I_data_empty.fits
$ esorex kmo_fits_strip --angle=120 xcal_HHH.fits
```

8 Data Reduction Library Functions

All recipes described in Section 7 are implemented as functions with similar names inside the library. Their descriptions have not been repeated here. By implementing them as functions allows one to create an appropriate simple wrapper so that they can be used either as recipe plugins or for use within KMCLIPM, without having to repeat the functional part of the code.

In addition, there are a few extra functions, which are defined as such because they are used repeatedly in various recipes or fulfil another special task. These are the functions described explicitly in this chapter.

8.1 Acquisition Reduction for RTD

Recipe name	used in recipe/function	uses recipe/function
kmclipm_rtd_image	-	kmo_make_image kmo_fit_profile kmos_reconstruct

`kmclipm_rtd_image` is intended to be used only by the Instrument Control Software (ICS). In order to use it the function

```
kmclipm_set_cal_path(const char *path, int test_mode)
```

has to be called once, defining the path where the xcal-, ycal- and lcal-calibration files are stored and whether we are in test mode (default: `test_mode = FALSE`) or not.
The calibration files are generated using the recipes kmos_flat and kmos_wave_cal (see Sec. 7.1.2 and 0) and have manually to be copied to the specified directory in order to use the real time display (RTD) in ICS.
To create the necessary raw frames for the above mentionned recipes, the templates

KMOS_spec_cal_calunit and
KMOS_spec_cal_wave

have to be executed. There the number of rotator offsets has to be specified (for the moment beeing 6 offsets are recommended). So we get for 5 bands and 6 angles and 3 different calibration files a total of 90 calibration files.
The naming of the calibration files follows this convention:
e.g. xcal_xxx_yyy_z.fits
x: grating for every detector
y: filter for every detector
z: rotator angle

8.1.1 Description

In general, only a few bright stars are observed with a few IFUs (with short DIT). The other IFUs point to faint objects that will not necessarily be visible with the short DIT used for acquisition. Thus a vector is provided indicating the IFUs, which are to be processed. However, in some cases, the calculations and image reconstruction will be performed for all IFUs (initial tests, calibrations, etc).

8.1.2 Flow Chart

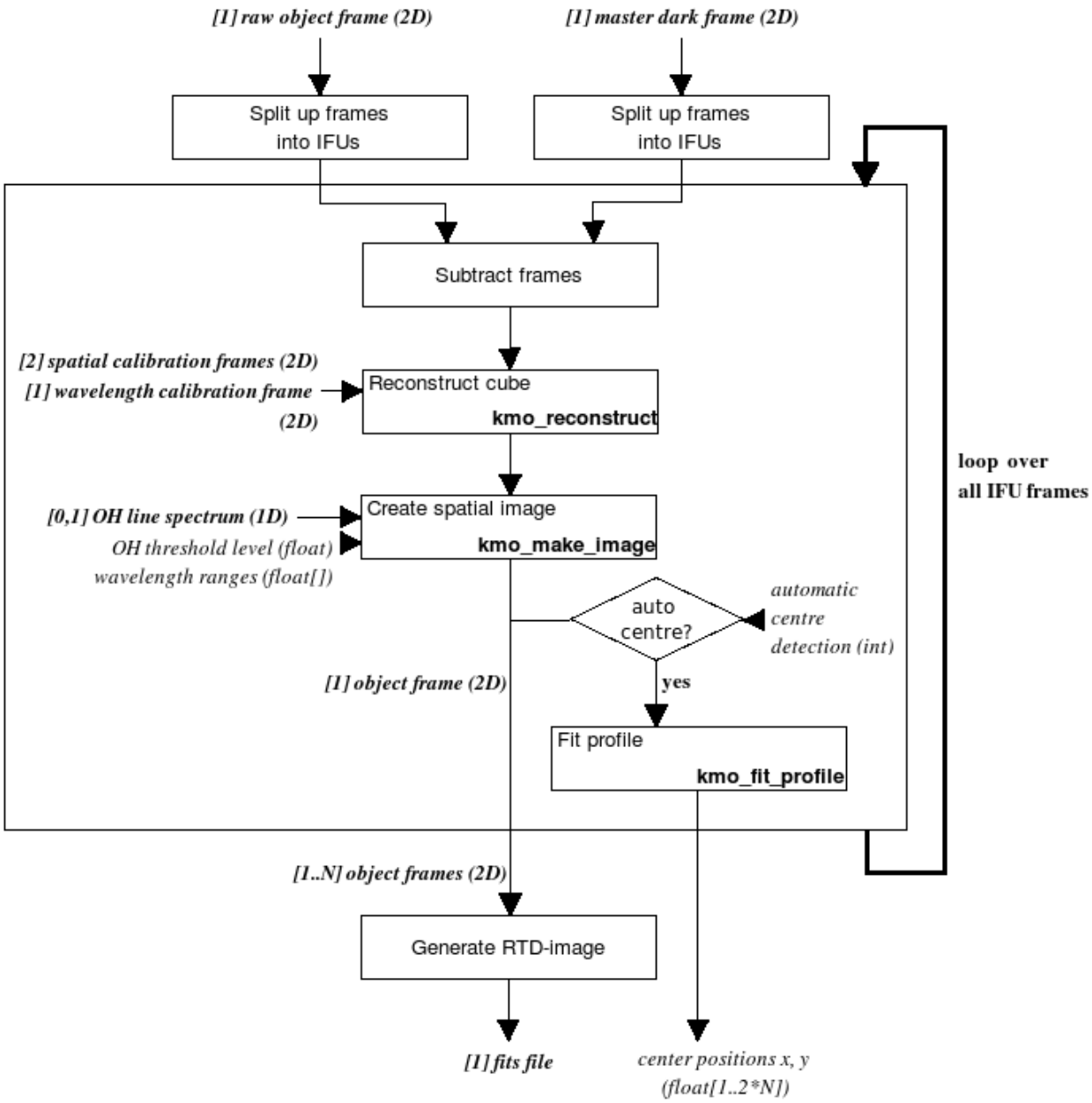

Figure 55**:** Flow chart of the recipe **kmclipm_rtd_image**

The processing steps are:

1. From the raw object frame and the master dark (or a specified sky frame) the desired IFU frame is extracted.
2. The two IFU frames are subtracted.
3. The resulting frame is reconstructed into a cube using the bad pixel mask, the spectral curvature calibration frame and the wavelength calibration frame.
4. The cube is collapsed along the spectral axis within specified wavelength range. If required wavelengths across OH sky emission lines are omitted.
5. If automatic centres are required, they will be extracted now. The resulting x- & y-values are stored in a vector.
6. The steps above are repeated for each IFU to process.
7. The resulting images are merged into a single combined image.

8.2 Combine frames using pixel rejection

Recipe name	used in recipe/function	uses recipe/function
kmclipm_combine_frames	kmos_dark kmos_flat kmos_illumination	-

Combines data frames with or without noise and either (re)calculates or propagates noise.

8.2.1 Description

This function is always used when several input frames have to be combined into one. For each pixel position the pixel values at this position of every frame are put into a vector. This vector is to be averaged according one of the following methods available:

- Kappa-sigma clipping
  Any value of the vector which deviates significantly will be rejected. This method is iterative.
  (value > mean + σ * *pos_rejection_threshold* or
  value < mean - σ * *neg_rejection_threshold*)
  In the first iteration median and percentile level are used.
- Min-max clipping
  The specified number of minimum and maximum values of the vector will be rejected. This method is applied once.
- Average
  The average of all values of the vector is calculated.
- Median
  The median of all values of the vector is calculated.
- Sum
  The sum of all values of the vector is calculated.

The above mentionned methods act all the same regardless the number of input data frames. For reasonable noise estimations it is recommended to provide at least three or more frames. If less than three frames are provided the noise estimation is performed as depicted in Table 1 below.

8.2.2 Flow Chart

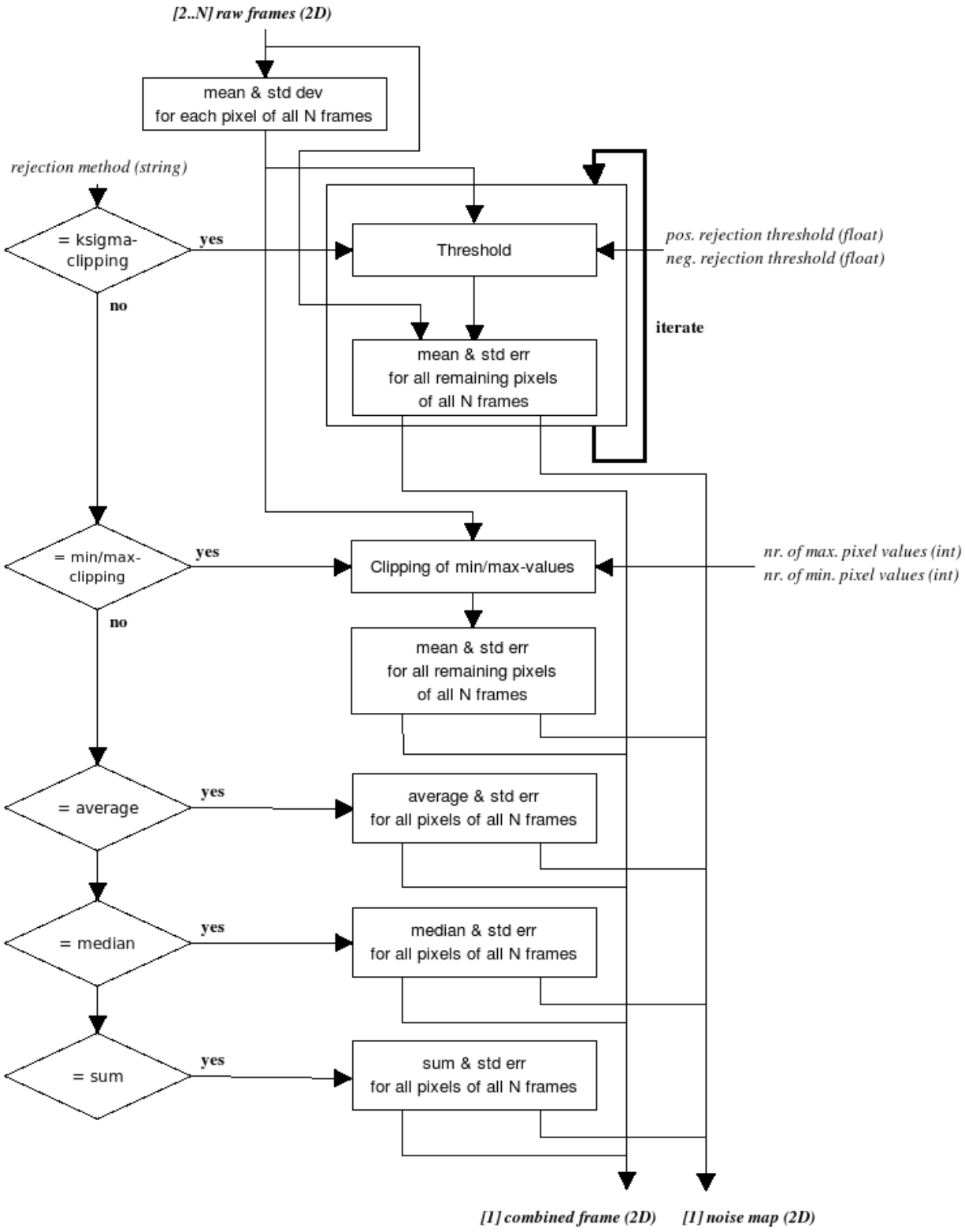

Figure 56: Flow chart for **kmclipm_combine_frames**

The processing steps are:

1. Depending on the method chosen the frames will be combined differently:
    a. Kappa-sigma clipping
        i. Two thresholds are calculated

ii. All pixels above or below the thresholds are rejected
iii. These steps are repeated as many times as desired

b. Min-max clipping
   i. The desired number of minimum and maximum values is clipped
c. Average
   i. For all pixel positions the average of the values is calculated
d. Median
   i. For all pixel positions the median of the values is calculated
e. Sum
   i. For all pixel positions the sum of the values is calculated

	≥3 frames	**2 frames**	**1 frame**
	$avg_{data} = combine(data_{in})$	$avg_{data} = combine\ (data_{in})$	$avg_{data} = data_{in}$
with noise$_{in}$	***for 'median' method:*** $avg_{noise} = \frac{stdev_{median}(data_{in})}{\sqrt{n}}$ ***for all other methods:*** $avg_{noise} = \frac{stdev(data_{in})}{\sqrt{n}}$	***for 'sum' method:*** $avg_{noise} = \sqrt{\frac{noise_{in1}^2 + noise_{in2}^2}{2}}$ ***for all other methods:*** $avg_{noise} = \frac{1}{2}\sqrt{\frac{noise_{in1}^2 + noise_{in2}^2}{2}}$	$avg_{noise} = noise_{in}$
	$avg_{data} = combine\ (data_{in})$	$avg_{data} = combine\ (data_{in})$	$avg_{data} = data_{in}$
w/o noise$_{in}$	$avg_{noise} = \frac{stdev(data_{in})}{\sqrt{n}}$	$avg_{noise} = \frac{\lvert data_{in1} - data_{in2} \rvert}{\sqrt{2}}$	$avg_{noise} = stdev(data_{in})$

Table 1 The function *combine*() stands for kmclipm_combine_frames() and handles the input data as described above. n is the number of input frames.

PART III: DRS MAINTAINANCE & DEVELOPMENT

In this section different aspects regarding the creation of new calibration data and the maintainance of the KMOS DRS pipeline for further development are described.

9 Updating the Calibration Data

After instrument maintainance or after a warming up/cooling down-cycle the instrument, the arms and spectrographs are expected to have different characteristics than before. In this case it is strongly recommended to renew the calibration files. Manipulation of the arms particularly affects all spatial calibration files (BADPIXEL_DARK, BADPIXEL_FLAT, FLAT_EDGE, ILLUM_CORR, MASTER_DARK, MASTER_FLAT, TELLURIC, XCAL, YCAL), manipulation of the spectrographs affects all spectral calibrations (WAVE_BAND, REF_LINES, LCAL, ILLUM_CORR, TELLURIC).
Besides this the spatial and spectral calibration frames are expected to be stable in time. So we recommend to renew these files every few months to a year (exact intervals have to be defined during operation) in normal operating conditions.

9.1 Alphabetical listing of all calibration files

There are 5 bands (H, HK, IZ, K, YJ). For each band there are 6 rotator angles (0, 60, 120, 180, 240, 360 degrees). Therefore the band-specific calibrations will amount to 15 files.[1]
The "Calib DB" depicted in **Error! Reference source not found.** consists of the frames listed below:

PRO CATG	size	#	comments
ARC_LIST	8.5 KB	5	Arcline lists
ATMOS_MODEL	35 KB	5	Atmospheric models
BADPIXEL_DARK	48 MB	1	Badpixel frame
BADPIXEL_FLAT	289 MB	5	Badpixel frame after flat-fielding
FLAT_EDGE	1.5 MB	5	Table with edge fitting parameters
LCAL	289 MB	5	Spectral calibration file
MASTER_DARK	97 MB	1	Master dark frame
MASTER_FLAT	577 MB	5	Master flat frames
OH_SPEC	300 KB	5	
REF_LINES	8.5 KB	1	Table with arcline fitting parameters
SOLAR_SPEC	50 KB	5	Solar spectra
SPEC_TYPE_LOOKUP	8.5 KB	1	Spectral lookup table
XCAL	289 MB	5	Spatial calibration file No. 1
YCAL	289 MB	5	Spatial calibration file No. 2
WAVE_BAND	8.5 KB	1	Fits table with start-/end-values of wavebands

[1] The filter resp. grating setting can be extracted from following keywords (It is guaranted by the ICS that these keywords always contain the same value):

ESO INS FILT1 ID	ESO INS GRAT1 ID
ESO INS FILT2 ID	ESO INS GRAT2 ID
ESO INS FILT3 ID	ESO INS GRAT3 ID

The rotator angle can be extracted from:
ESO OCS NAANGLEx

Σ	**8.8 GB**	**55 files**

9.1.1 Subset of calibration files needed for Karma and RTD (kmclipm)

The "IWS calib (subset)" consists of a subset of above frames:

PRO CATG	size	#	comments
LCAL	289 MB	5	**Important:** on IWS the original filenames have to be used and not the archived filenames (e.g. lcal_HHH_HHH.fits and kmos_wave_band.fits)
XCAL	289 MB	5	
YCAL	289 MB	5	
WAVE_BAND	8.5 KB	1	
Σ	**4.3 GB**	**16 files**	

9.2 Calibration Schedule

9.2.1 Every few months to a year or after instrument maintainance

BADPIXEL_DARK
BADPIXEL_FLAT
FLAT_EDGE
ILLUM_CORR
MASTER_DARK
MASTER_FLAT
TELLURIC
XCAL
YCAL
LCAL

These calibration frames can be produced using the KMOS data reduction pipeline. As input data new calibration exposures have to be produced using the following ICS templates with default settings (the templates switch the filters and gratings automatically where appropriate):

Template ID	DO category	Associated recipe	DO category
KMOS_spec_cal_dark[2]	DARK	kmos_dark	MASTER_DARK BADPIXEL_DARK
KMOS_spec_cal_unit[3]	FLAT_ON FLAT_OFF	kmos_flat	MASTER_FLAT BADPIXEL_FLAT XCAL YCAL FLAT_EDGE
KMOS_spec_cal_wave[3]	ARC_ON ARC_OFF	kmos_wave_cal	LCAL DET_IMAGE_WAVE

To check the quality of the calibration frames the QC parameters can be checked and compared to former values, best to be stored in an external database. Additionally a quick visual check consists of looking at the generated DET_IMG_WAVE frames from kmos_wave_cal-recipe. Obvious errors in the wavelength calibration are easy to identify.

The estimated execution time for the first three templates involved in this procedure is about 3.5 hours. About the same time is needed to execute the associated pipeline recipes.

9.2.2 After instrument maintainance

9.2.2.1 WAVE_BAND

This file can be updated using the DRS pipeline and a text file containing the actual values.

9.2.2.2 REF_LINES

Introduction

The KMOS spectral calibration recipe uses a two step approach to fit the calibration lamp images to the spectral line lists. In the first step it tries to fit a few selected well isolated (reference) lines. Using the position of these lines a first attempt to fit the position of the lines versus their wavelength is done. For the rest of the lines this estimate is used to detect them in a much smaller search range.
An IDL tool is available to create and maintain a table of reference lines for each band each table holding the lines for the three detectors.

2 single execution at any rotator angle

3 multiple executions at following rotator angles: 0, 60, 120, 180, 240, 300 degrees

Prerequisites

Trace files:

The IDL tool to generate the reference table needs some data input files which hold the trace over the detector for each slitlet of each IFU in lambda direction. These data files are created using the KMOS pipeline running the KMOS_WAVE_CAL recipe. To trigger the creation of the data files the environment variable KMOS_WAVE_CAL_DATA_PREFIX must be set. The string you choose will prefix all data files. The file name will be completed by the recipe with band ID, IFU and slitlet number. It is a good choice to start the KMOS_WAVE_CAL_DATA_PREFIX with a "." character to hide the trace data files in standard directory views.
The trace data files will be created in the current working directory.
When no reference table calibration file exists yet or it is too bad to process all slitlets it is necessary to use a different line estimation method in the KMOS_WAVE_CAL recipe. Again an environment variable is used, KMOS_WAVE_LINE_ESTIMATE. Setting it to value of "0" shall result in a stable run of the KMOS_WAVE_CAL recipe to generate the trace data files.
Don't forget to remove the environment variables after the trace data files are generated.

Examples:

bash shell:

```
$ export    KMO_WAVE_CAL_DATA_PREFIX=.wave_cal_data
$ export    KMO_WAVE_LINE_ESTIMATE=0
$ esorex    kmos_wave_cal wavecal.sof
$ unset     KMO_WAVE_CAL_DATA_PREFIX
$ unset     KMO_WAVE_LINE_ESTIMATE
```

csh shell:

```
$ setenv    KMO_WAVE_CAL_DATA_PREFIX .wave_cal_data
$ setenv    KMO_WAVE_LINE_ESTIMATE 0
$ esorex    kmos_wave_cal wavecal.sof
$ unsetenv  KMO_WAVE_CAL_DATA_PREFIX
$ unsetenv  KMO_WAVE_LINE_ESTIMATE
```

Line lists:

The IDL tool to generate the reference table needs the same line list files as used by the KMOS_WAVE_CAL recipe of the pipeline. The combined AR and NE line list files are expected to have the file naming convention:

```
kmos_ar_ne_list_%s.fits
```

with `%s` being the lower case band name (h, k, hk, yj or iz).

Usage of the IDL tool

The IDL tool to generate the reference table is distributed as compressed TAR file. Create a new empty directory, use it as working directory and untar the TAR file:

```
$ mkdir <dirName>
$ cd <dirName>
$ tar -cxf <tarfile>
```

There are several ways to start the IDL tool

1. `$ idl main.pro`
   Use this way if your IDL installation allows the call of IDL batch file as argument. In this case you can edit *main.pro* to setup your default values for the arguments.
2. `$ idl -idl_startup main.pro`
   This way should work with any IDL installation. Again you can edit *main.pro* to setup your default values for the arguments.

3. `$ idl -vm=kmos_wave_calib.sav`
   This way should work with any IDL installation. All required arguments must be specified in the command line.
4. `$ idl -vm=`*`installationDir/`*`kmos_wave_calib.sav`
   This way should work with any IDL installation. This way your working directory can be different. All required arguments must be specified in the command line.

Following arguments are required by the IDL tool:

cal_dir	Specifies the directory where the line list calibration files will be found
data_dir	Specifies the directory where the trace data file will be found.
prefix	Specifies the name prefix of the trace data files.
table	Specifies the name (including the directory path) of the reference lines table

Following arguments are optional:

table	Specifies a FITS table with a set of reference line to start with. The default is an empty table.
pattern	Specifies file pattern for line list FITS tables. The pattern must contain at least one "`%s`" placeholder for the band name. The default is the pattern `kmos_ar_ne_list_%s.fits`

Default values for your installation can be fixed in the file *main.pro*.

Examples:

```
$ idl main.pro -args –table=../ref_lines.fits
```

The other argument values are defined in main.pro.

```
$ idl -idl_startup main.pro -args -table=../ref_lines.fits \
                                --prefix=test
```

The other argument values are defined in main.pro.

```
$ idl -vm=kmos_wave_calib.sav -args –table=../ref_lines.fits\
                                --prefix=.wave_cal_data \
                                -cal_dir=../calFiles \
                                -data_dir=../data
```

9.3 How to create XCAL, YCAL and LCAL

To create proper XCAL, YCAL and LCAL frames at the standard 6 rotator angles of 0°, 60°, 120°, 180°, 240° and 300° following templates have to be executed: KMOS_spec_cal_dark, KMOS_spec_cal_calunitflat and KMOS_spec_cal_wave. Best practice is to run these templates in a single OB with their default values.
Furthermore the following static calibration frames of type `ARC_LIST`, `REF_LINES` and `WAVE_BAND` are needed. They can be found in the SVN repository `kmos-calib` or in the pipeline deliverable.

```
kmos_ar_ne_list_h.fits
kmos_ar_ne_list_hk.fits
kmos_ar_ne_list_k.fits
kmos_ar_ne_list_iz.fits
kmos_ar_ne_list_yj.fits
kmos_wave_band.fits
kmos_wave_ref_table.fits
```

The following workflow creates all needed calibration frames for the RTD, for the pipeline workstation and for anyone using the KMOS DRS pipeline. The needed python script

`kmos_calib.py` can either be found in the SVN repository `kmosp/tools/` or in the `bin` folder of the installed pipeline (e.g. using the distributed kmos-kit).

10 Environment variables

Here all environment variables influencing the behaviour of the pipeline are described.

10.1 Variables influencing wavelength calibration

KMO_WAVE_CAL_DATA_PREFIX
Used in kmos_wave_cal recipe.
If this environment variable is set, special FITS files will be created to support the editing of the reference lines calibration file. The value of this environment variable will be used as prefix for those files. As there will be generated more than 2000 files this prefix usually starts with a "." character to hide them from file listings. The final file name will be:

{PREFIX}_{BAND}_ifu_{IFUNUMBER}_slitlet_{SLITLETNUMBER}.fits

10.2 Variables influencing reconstruction

KMO_WAVE_RECONSTRUCT_METHOD (default if not set: "lwNN")
Used in kmos_wave_cal recipe.
Specifies the reconstruction method for creating the reconstructed detector image. Following values are allowed:

- NN nearest neighbor
- lwNN linear weighted nearest neighbor (default)
- swNN square weighted nearest neighbor
- MS Modified Shephards method
- CS cubic spline
-

KMCLIPM_PRIV_RECONSTRUCT_LUT_MODE (default if not set: FILE)
Influences the handling of the LUT. There are four modes. They are explained in detail in section 6.4.

- NONE
- MEMORY
- FILE
- BOTH

10.3 Variables influencing RTD reconstruction

RTD_CHECK_FOR_SATURATION
If defined the detector image given to reconstruction will be checked for saturated pixels. Saturated pixels will be added to the bad pixel mask.

KMO_RECONSTRUCT_BADPIXEL_VALUE
If this variable is not set bad pixel will be ignored. If it is set all bad pixels will be set to this value before reconstruction.

RTD_RECONSTRUCT_METHOD (default if not set: NN)
If this variable is not set RTD reconstruction will use simple nearest neighbor reconstruction. It can be set to:

- NN nearest neighbor (default)
- LWNN linear weighted nearest neighbor
- SWNN square weighted nearest neighbor
- MS Modified Shephards method
- CS cubic spline

10.4 Variables influencing the debugging

KMO_TEST_VERBOSE (default if not set: no error messages shown)
Switches on and off error-messages for the unit tests globally. It is only seen if a test fails or not. Either et this variable or look at the generated log files for debugging the unit tests.

KMCLIPM_ DEBUG (default if not set: no fit-parameters displayed or stored)
Prints gauss-fit parameters to the console and saves them as well in a file at KMCLIPM_DEBUG/tmp/kmclipm_fitpar.txt

This variable is somehow obsolete, since now all fit aprameters are returned from kmclipm_rtd_image() anyway. This way the ICS software has full control over the decision how to handle a reconstructed and collapsed cube.

Appendix A Data Processing Tables

Recipe	Template	Classification Keywords	Calibration Database	Data Products	QC1 Parameters
kmos_dark	KMOS_spec_cal_dark	DO cat = DARK DPR.TYPE = DARK DPR.CATG = CALIB DPR.TECH = IMAGE	-	Master Dark frame Preliminary Bad pixel mask Dark Current Read noise	Mean Bias Mean Read Noise Number of bad pixels Mean Dark Current
	Processing: iterative mean of frames; identify bad pixels				
	FITS keywords: DIT, MINDIT				
kmos_flat	KMOS_spec_cal_calunit	DO cat = FLAT_ON DPR.TYPE = FLAT,LAMP DPR.CATG = CALIB DPR.TECH = SPECTRUM DO cat = FLAT_OFF DPR.TYPE = FLAT,OFF DPR.CATG = CALIB DPR.TECH = IMAGE	Preliminary Bad pixel mask (from kmos_dark)	Master Flat Spectral Curvature Calibration frames Final Bad pixel mask	Mean shift of slitlet edges RMS shift of slitlet edges Lamp efficiency Number of saturated pixels in flatfield Mean S/N in flatfield Mean change in 0^{th} order coefficients RMS change in 0^{th} order coefficients Mean change in 1^{st} order Y coefficients RMS change in 1^{st} order Y coefficients
	Processing: subtract mean of on & off frames; identify pixels that are bad or not illuminated; fit functions to spectral traces; generate frame where the pixel value corresponds to the spatial position (in arcsec) of that pixel				
	FITS keywords: INS.FILTi.NAME, INS.LAMP3.ST, INS.LAMP4.ST				
kmos_illumination	KMOS_spec_cal_skyflat	DO cat = FLAT_SKY DPR.TYPE = FLAT,SKY DPR.CATG = CALIB DPR.TECH = IFU	Final Bad pixel mask Master Flat frame Spectral Curvature Calibration frame Wavelength Calibration frame	Illumination Correction frame	Spatial uniformity of flatfield Max deviation of an IFU identification of that IFU Max non-uniformity within an IFU identification of that IFU
	Processing: average frames; reconstruct cubes; collapse to images; normalise				
	FITS keywords: INS.FILTi.NAME				
kmos_wave_cal	KMOS_spec_cal_wave	DO cat = ARC_ON DPR.TYPE = WAVE,LAMP DPR.CATG = CALIB DPR.TECH = SPECTRUM DO cat = ARC_OFF DPR.TYPE = WAVE,OFF DPR.CATG = CALIB DPR.TECH = IMAGE	Final Bad pixel mask Arc line wavelength table	Wavelength Calibration frame	Arc lamp efficiency Number of saturated pixels in arc frame Spectral Resolution Mean change in 0^{th} order coefficients RMS change in 0^{th} order coefficients Mean change in 1^{st} order Y coefficients RMS change in 1^{st} order Y coefficients
	Processing: subtract on & off frames; fit functions to arc line traces; generate frame where the pixel value corresponds to the wavelength (in microns) of that pixel				
	FITS keywords: INS.FILTi.NAME, INS.LAMP1.ST, INS.LAMP2.ST				

Recipe	Template	Classification Keywords	Calibration Database	Data Products	QC1 Parameters
kmos_std_star	KMOS_spec_cal_std	DO cat = STD DPR.TYPE = OBJECT,SKY,STD,FLUX DPR.CATG = CALIB DPR.TECH = IFU	Final Bad pixel mask Master flat frame Wavelength Calibration frame Spectral Curvature Calibration frame Illumination Correction Frame Model Atmospheric Transmission Spectrum Solar Spectrum Spectral Type Lookup Table	Telluric Correction Spectrum Images of the stars (for seeing measurement) Flux Calibration (if star magnitude given)	Mean Zeropoint Mean & Std Dev Throughput Mean Spatial Resolution Straightness of corrected trace
	Processing: subtract object & sky frames; reconstruct cube; extract spectrum; correct stellar imprint; calculate flux calibration				
	FITS keywords: INS.FILTi.NAME, OCS.ARMi.TYPE				
kmos_sci_red	KMOS_spec_obs_nodtosky KMOS_spec_obs_stare KMOS_spec_obs_mapping	DO cat = SCIENCE DPR.TYPE = OBJECT,SKY DPR.CATG = SCIENCE DPR.TECH = IFU	Final Bad pixel mask Master flat frame Wavelength Calibration frame Spectral Curvature Calibration frame Illumination Correction Frame Telluric Correction Spectrum	Reduced Science Cube	none
	Processing: subtract object & sky frames; reconstruct cube; divide out telluric imprint; calibrate flux				
	FITS keywords: INS.FILTi.NAME, OCS.ARMi.ALPHA, OCS.ARMi.DELTA, OCS.ARMi.TYPE, OCS.ARMi.NAME				
kmo_rtd_image	triggered by CLIP	N/A	Final Bad pixel mask Master Dark frame Wavelength Calibration frame Spectral Curvature Calibration frame OH line wavelength table	Reconstructed images (to display on RTD)	none
	Processing: subtract object & sky/dark frames; reconstruct cube; excise regions near OH lines; collapse spectral axis to create image				
	FITS keywords: N/A				
kmos_molecfit_model		DO cat = STD, SCIENCE DPR.TYPE = ATMOS_PARAM, BEST_FIT_MODEL, BEST_FIT_PARM DPR.CATG = CALIB DPR.TECH = IFU	Kernel library	Atmospheric model	None
	Processing: sky frames; extract spectrum; calculate atmospheric model				
	FITS keywords: HIERACH KMOS MOLECFIT MODEL PARAM <list_parameters>				
kmos_molecfit_calctrans		DO cat = STD, SCIENCE DPR.TYPE = TELLURIC_DATA, TELLURIC_CORR DPR.CATG = CALIB DPR.TECH = IFU	Kernel library	Telluric correction data	None
	Processing: sky frames; extract spectrum; calculate telluric correction with the atmospheric model				
	FITS keywords: HIERACH KMOS MOLECFIT MODEL PARAM <list_parameters>				
kmos_molecfit_correct		DO cat = STD, SCIENCE DPR.TYPE = SINGLE_SPECTRA,SINGLE_CUBES DPR.CATG = SCIENCE DPR.TECH = IFU	Response	Telluric Correction spectrum	
	Processing: sky frames; apply telluric correction				
	FITS keywords: N/A				

Appendix B The KMOS data interface dictionary

The column dependency indicates that the QC parameter will be different for (i.e. depends on) each detector ('D'), each IFU ('I') and/or each bandpass ('B') respectively.

Table of (possibly) generated keywords by the calibration recipes of the DRS:

name	*header*	*unit*	*data type*	*depen-dency*	*description*
HIERARCH ESO PRO ARMx NOTUSED	primary	-	string	I	This keyword is only present when a recipe wasn't able to process a specific IFU ([] IFU set inactive by <recipe_name>)
HIERARCH ESO PRO BOUND IFUi_L	primary	pix	int	I	This keyword contains the left bound of the area on the detector containing IFU i. This keyword is generated in kmos_flat and stored in the xcal-frame for every active IFU. This information is reused when reconstructing.
HIERARCH ESO PRO BOUND IFUi_R	primary	pix	int	I	This keyword contains the right bound of the area on the detector containing IFU i. See also comment above.
HIERARCH ESO PRO ROT NAANGLE	extension	deg	double	D	This keyword is set by kmos_flat and kmos_wave_cal to indicate which extension belongs to which rotator angle.

Table of generated QC keywords by the calibration recipes (see section 5.1 for more detailed information):

name	*header*	*unit*	*data type*	*depen-dency*	*description*
kmos_dark					
HIERARCH ESO QC DARK	extension	adu	double	D	mean value of Master Dark
HIERARCH ESO QC DARK MEDIAN	extension	adu	double	D	median value of Master Dark
HIERARCH ESO QC RON	extension	adu	double	D	mean value of noise of Master Dark
HIERARCH ESO QC RON MEDIAN	extension	adu	double	D	median value of noise of Master Dark
HIERARCH ESO QC DARKCUR	extension	e^-/s	double	D	iterative mean dark current in Master Dark divided by gain
HIERARCH ESO QC BADPIX NCOUNTS	extension	-	int	D	number of bad pixels in Master Dark
kmos_flat					
HIERARCH ESO QC FLAT EFF	extension	e^-/s	double	DB	relative brightness of flatfield lamp
HIERARCH ESO QC FLAT SAT NCOUNTS	extension	-	int	DB	number of saturated pixels in Master Flat
HIERARCH ESO QC FLAT SN	extension	-	double	DB	signal-to-noise in Master Flat

HIERARCH ESO QC GAP MEAN	extension	pix	double	DB	mean gap width between slitlets
HIERARCH ESO QC GAP SDV	extension	pix	double	DB	standard deviation of gap width between slitlets
HIERARCH ESO QC GAP MAXDEV	extension	pix	double	DB	maximum deviation of gap width between slitlets
HIERARCH ESO QC SLIT MEAN	extension	pix	double	DB	mean slitlet width
HIERARCH ESO QC SLIT SDV	extension	pix	double	DB	standard deviation of slitlet width
HIERARCH ESO QC SLIT MAXDEV	extension	*pix*	double	DB	maximum deviation of slitlet width
HIERARCH ESO QC BADPIX NCOUNTS	extension	-	int	D	number of bad pixels in Master Flat
kmos_wave_cal					
HIERARCH ESO QC ARC AR EFF	extension	e^-/s	double	B	relative brightness of argon arclamp
HIERARCH ESO QC ARC NE EFF	extension	e^-/s	double	B	relative brightness of neon arclamp
HIERARCH ESO QC ARC SAT NCOUNTS	extension	-	int	B	number of saturated pixels in arc frame
HIERARCH ESO QC ARC AR POS MEAN	extension	km/s	double	DB	mean of all Argon reference line position offsets (measured vs. expected)
HIERARCH ESO QC ARC AR POS MAXDIFF	extension	km/s	double	DB	maximum offset of measured vs. expected Argon reference line position
HIERARCH ESO QC ARC AR POS MAXDIFF ID	extension	-	int	DB	identification of the IFU which has the maximum offset
HIERARCH ESO QC ARC AR POS STDEV	extension	km/s	double	DB	mean standard deviation of position offset for Argon reference line
HIERARCH ESO QC ARC AR POS 95%ILE	extension	km/s	double	DB	mean 95%ile of position offset for Argon reference line
HIERARCH ESO QC ARC AR FWHM MEAN	extension	km/s	double	DB	mean of FWHM for Argon reference line
HIERARCH ESO QC ARC AR FWHM STDEV	extension	km/s	double	DB	mean stdev of FWHM for Argon reference line
HIERARCH ESO QC ARC AR FWHM 95%ILE	extension	km/s	double	DB	mean 95%ile of FWHM for Argon reference line
HIERARCH ESO QC ARC NE POS MEAN	extension	km/s	double	DB	mean of all Neon reference line position offsets (measured vs. expected)
HIERARCH ESO QC ARC NE POS MAXDIFF	extension	km/s	double	DB	maximum offset of measured vs. expected Neon reference line position
HIERARCH ESO QC ARC NE POS MAXDIFF ID	extension	-	int	DB	identification of the IFU which has the maximum offset
HIERARCH ESO QC ARC NE POS STDEV	extension	km/s	double	DB	mean standard deviation of position offset for Neon reference line
HIERARCH ESO QC ARC NE POS 95%ILE	extension	km/s	double	DB	mean 95%ile of position offset for Neon reference line
HIERARCH ESO QC ARC NE FWHM MEAN	extension	km/s	double	DB	mean of FWHM for Neon reference line
HIERARCH ESO QC ARC NE FWHM STDEV	extension	km/s	double	DB	mean stdev of FWHM for Neon reference line
HIERARCH ESO QC ARC NE FWHM 95%ILE	extension	km/s	double	DB	mean 95%ile of FWHM for Neon reference line
kmos_illumination					
HIERARCH ESO QC SPAT UNIF	primary	adu	double	B	uniformity across all illumination corrections
HIERARCH ESO QC SPAT MAX DEV ID	primary	-	int	B	identification of the IFU whose illumination correction deviates most from unity
HIERARCH ESO QC SPAT MAX DEV	primary	adu	double	B	value of this deviation
HIERARCH ESO QC SPAT MAX NONUNIF ID	primary	-	int	B	identification of the IFU which has the most non-uniform illumination correction
HIERARCH ESO QC SPAT MAX NONUNIF	primary	adu	double	B	standard deviation of the illumination correction for this IFU
kmos_std_star					
HIERARCH ESO QC ZPOINT	extension	mag	double	DB	zeropoint (magnitude) [stored in extension headers of telluric]
HIERARCH ESO QC THRUPUT	extension	-	double	DB	throughput of KMOS (i.e. ratio of number of photons detected to number

					expected from the standard star) [stored in extension headers of telluric]
HIERARCH ESO QC THRUPUT MEAN	primary	-	double	B	mean of throughput of all detectors [stored in primary header of telluric]
HIERARCH ESO QC THRUPUT SDV	primary	-	double	B	standard deviation of throughput of all detectors [stored in primary header of telluric]
HIERARCH ESO QC SPAT RES	extension	-	double	DB	spatial resolution (FWHM) [stored in extension headers of std_image]
HIERARCH ESO QC STD TRACE	extension	pix	double	DB	a measure of how straight the corrected trace of a star is (i.e. how well the spectral curvature has been corrected) [stored in extension headers of std image]
HIERARCH ESO QC NR STD STARS	primary	-	int	I	the number of standard stars in a standard star exposure [stored in primary headers of all output frames]
HIERARCH ESO QC SNR	extension	-	double	B	the signal to noise ratio [stored in extension headers of noise spec]

Table of generated keywords by kmo_fit_profile (see section 0 for more detailed information):

name	*header*	*unit*	*data type*	*depen-dency*	*description*
HIERARCH ESO PRO FIT MAX PIX	extension	pix	double	I	Position of the maximum (1D fit)
HIERARCH ESO PRO FIT MAX PIX X	extension	pix	double	I	Position of the maximum in x (2D fit)
HIERARCH ESO PRO FIT MAX PIX Y	extension	pix	double	I	Position of the maximum in y (2D fit)
HIERARCH ESO PRO FIT CENTROID	extension	pix	double	I	Position of the centroid (1D fit)
HIERARCH ESO PRO FIT CENTROID X	extension	pix	double	I	Position of the centroid in x (2D fit)
HIERARCH ESO PRO FIT CENTROID Y	extension	pix	double	I	Position of the centroid in y (2D fit)
HIERARCH ESO PRO FIT RADIUS X	extension	pix	double	I	Radius in x of fitted 2D profile
HIERARCH ESO PRO FIT RADIUS Y	extension	pix	double	I	Radius in y of fitted 2D profile
HIERARCH ESO PRO FIT OFFSET	extension	adu	double	I	Background/offset
HIERARCH ESO PRO FIT INTENS	extension	adu	double	I	Intensity of the function
HIERARCH ESO PRO FIT SIGMA	extension	pix	double	I	Sigma of the gauss function
HIERARCH ESO PRO FIT ALPHA	extension	-	double	I	Alpha of fitted Moffat function
HIERARCH ESO PRO FIT BETA	extension	-	double	I	Beta of fitted Moffat function
HIERARCH ESO PRO FIT SCALE	extension	adu	double	I	Scale of fitted Lorentz function
HIERARCH ESO PRO FIT ROT	extension	deg	double	I	Rotation angle (clockwise)
HIERARCH ESO PRO FIT ERR CENTROID	extension	pix	double	I	Error in position of the centroid (1D fit)
HIERARCH ESO PRO FIT ERR CENTROID X	extension	pix	double	I	Error in position of the centroid in x (2D fit)
HIERARCH ESO PRO FIT ERR CENTROID Y	extension	pix	double	I	Error in position of the centroid in y (2D fit)
HIERARCH ESO PRO FIT ERR RADIUS X	extension	pix	double	I	Error in radius in x of fitted 2D profile
HIERARCH ESO PRO FIT ERR RADIUS Y	extension	pix	double	I	Error in radius in y of fitted 2D profile
HIERARCH ESO PRO FIT ERR OFFSET	extension	adu	double	I	Error in background/offset
HIERARCH ESO PRO FIT ERR ROT	extension	deg	double	I	Error in rotation angle (clockwise)
HIERARCH ESO PRO FIT ERR INTENS	extension	adu	double	I	Error in intensity of the function

HIERARCH ESO PRO FIT ERR SIGMA	extension	pix	double	I	Error in sigma of the gauss function
HIERARCH ESO PRO FIT ERR ALPHA	extension	-	double	I	Error in alpha of fitted Moffat function
HIERARCH ESO PRO FIT ERR BETA	extension	-	double	I	Error in beta of fitted Moffat function
HIERARCH ESO PRO FIT ERR SCALE	extension	adu	double	I	Error in scale of fitted Lorentz function
HIERARCH ESO PRO FIT RED CHISQ	extension	-	double	I	Reduced chi square error of the fit

___oooOOOooo___