

Document Title: SPARK INstructional Guide
for KMOS data

Document Number: VLT-TRE-KMO-146611-009

Issue: 0.5 

Date: 31.01.13

Document Prepared By:	R. Davies A. Agudo Berbel E. Wiezorrek	Signature and Date:
Document Approved By:	N. Förster Schreiber	Signature and Date:
Document Released By:	A. Fairley	Signature and Date:



UK
Astronomy Technology Centre



SPARK INstructional Guide for KMOS data

Doc No:	VLT-TRE-KMO-146611-009
Version:	0.5
Author	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

Change Record

Issue	Date	Section(s) Affected	Description of Change/Change Request Reference/Remarks
0.5	31.01.13	All	Draft
1.0			First Issue



SPARK INstructional Guide for KMOS data

Doc No:	VLT-TRE-KMO-146611-009
Version:	0.5
Author	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

TABLE OF CONTENTS

1	SCOPE.....	4
2	GETTING STARTED WITH ESOREX	4
2.1	INSTALLATION.....	4
2.2	USING THE SOFTWARE.....	5
2.2.1	<i>ESOREX & Recipes</i>	5
2.2.2	<i>Static Calibration Files</i>	6
2.2.3	<i>SPARKplug</i>	7
2.2.4	<i>easySPARK scripts</i>	7
3	HANDLING KMOS DATA.....	8
3.1	DATA FORMAT AND PROPERTIES	8
3.2	HEADER KEYWORDS.....	9
3.3	IFU ORIENTATION AND PIXEL ARRANGEMENT	10
3.4	IMPACT OF FLEXURE.....	11
4	PROCESSING CALIBRATIONS.....	11
4.1	DARKS.....	12
4.2	FLATS.....	12
4.3	ARCS.....	14
4.4	ILLUMINATION CORRECTION	15
4.5	STANDARD STARS.....	15
4.5.1	<i>Flux Calibration</i>	16
4.5.2	<i>Telluric Calibration</i>	17
5	SCIENCE REDUCTION	18
5.1	MONOLITHIC PIPELINE.....	18
5.1.1	<i>Mapping & Mosaics</i>	19
5.2	WORK-FLOW ONE STEP AT A TIME.....	20
5.2.1	<i>easySPARK_reconstruct</i>	21
5.3	ALTERNATIVES & OPTIMISATION	22
5.3.1	<i>Residual Sky Subtraction</i>	22
5.3.2	<i>Wavelength Corrections</i>	22
5.3.3	<i>Background Matching</i>	22
5.3.4	<i>Edge Effects</i>	23
5.3.5	<i>Improving Cosmetics</i>	23
5.3.6	<i>Weird Things</i>	23
6	NOTES ON RECONSTRUCTION	24
6.1	INTERPOLATION METHODS.....	24
6.2	MULTI-RECONSTRUCT	25
7	OTHER USEFUL RECIPES.....	26
7.1	SIMPLE MATHEMATICS.....	26
7.2	BASIC STATISTICS.....	26
7.3	MAKE IMAGES	26
7.4	EXTRACT SPECTRA	27
7.5	ROTATE CUBES	27
7.6	COPY CUBE SECTIONS	27



1 Scope

SPARK is the Software Package for Astronomical Reduction with KMOS. It includes the official pipeline release, as well as some additional perl and shell scripts that can help make using it easier.

This document describes how to get started using SPARK to process KMOS data without reading the full manual. It does not include everything, only the essential things you need to know together with some useful tips. KMOS is a complex instrument and, inevitably, so is the data and the data processing. We have tried to keep it as simple as possible. The guide may seem long, but it takes you through step-by-step, providing examples to follow. So just start, and work your way through it. We hope it is useful, both for beginners and as a reference.

If you use this software, please cite the following reference (which will be updated during 2013):

Davies R., Agudo Berbel A., Wierorrek E., Ott T., Foerster-Schreiber N.M., 2010
in “Ground-based and Airborne Instrumentation for Astronomy III”
eds. McLean I., Ramsay S., Takami H.,
Proc. SPIE, 7735

2 Getting started with ESOREX

The pipeline can be run on the command line using ESOREX (ESO’s recipe execution tool; see <http://www.eso.org/sci/software/cpl/esorex.html>).

In principle, one can also use GASGANO which provides some file sorting capability and a graphical interface to ESOREX. We prefer to use the tools provided with SPARK: either the SPARKplug to sort files and to create the file lists needed by ESOREX, or the set of easySPARK scripts to do this in an automated way. But file lists can also be made by hand using any editor. If you want to use GASGANO and need help, please contact ESO’s User Support Department.



2.1 Installation

SPARK is distributed by ESO as a kit (`kmos-kit-x.x.x.tar.gz`) containing the official pipeline recipes, some additional tools, the manual, and a set of standard calibration files. The calibration files are huge and are only included for the automatic data processing at Paranal. To avoid users having to download such a large dataset, MPE will also distribute their own kit without the calibrations. In either case, users should create their own calibrations in order to obtain the best results.

The software runs on all major Unix-based operating systems, as well MacOSX. For installation, the script `install_pipeline.sh`, included in the kit, has to be executed. At installation, a target directory (e.g. `/share/KMOSpipeline`) and a calibration directory (e.g. `/share/KMOScalib`) must be specified. Note that throughout this guide we use `/share` as the path to the KMOS directories.

Installing GASGANO is a tad more tricky since it requires the path to the java runtime. We do not discuss the use of GASGANO in this guide. If you also do not need GASGANO, you can delete the `gasgano` tar-file and the `install_pipeline.sh` script will just skip it.

As a first check to see if all the necessary libraries have been installed correctly, just type:



SPARK INstructional Guide for KMOS data

Doc No:	VLT-TRE-KMO-146611-009
Version:	0.5
Author:	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

```
> esorex
```

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

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

(the FFTW isn't distributed with the kit and isn't needed for KMOS).

After installation, add `/share/KMOSpipeline/bin` to your PATH environment variable (in `.tcshrc` or `.bashrc`) and copy all the scripts from `kmoss-kit-x.x.x/kmoss-x.x.x/tools/easySPARK` and from `kmoss-kit-x.x.x/kmoss-x.x.x/tools/SPARKplug` to `/share/KMOSpipeline/bin`. Ensure that all easySPARK scripts and SPARKplug.pl are executable (if they are not then execute

```
> chmod a+x easySPARK_* SPARKplug.sh in /share/KMOSpipeline/bin).
```

For SPARKplug.pl to run, the 'perl-tk' module must be installed. (For MacOSX using MacPorts, install the port 'p5-tk').

If ESOREX doesn't behave as described in this manual, some configurations can be done manually. The most comprehensible way is to type:

```
> esorex --create-config=true
```

This creates `.esorex/esorex.rc` in your HOME directory which can be edited in any text editor and provides a multitude of configuration possibilities. For example set

```
esorex.caller.suppress-prefix=TRUE
```

in order to override the standard ESOREX file-naming convention which defaults to `out_XXX.fits`.

2.2 Using the software

2.2.1 ESOREX & Recipes

Help for esorex is provided by the command:

```
> esorex -help
```

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

```
Usage: esorex [esorex-options] recipe [recipe-options] sof
```

And a list of the recipes available is given by:

```
> esorex -recipes
```

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

```
List of Available Recipes :
```

```
kmo_arithmetic      : Perform basic arithmetic on cubes
kmo_combine         : Combine reconstructed cubes
kmo_copy            : Copy a section of a cube to another cube, image or
                    spectrum
kmo_dark            : Create master dark frame & bad pixel mask
kmo_dev_setup       : Create aligned KMOS files out of test frames
kmo_extract_spec    : Extract a spectrum from a cube.
kmo_fits_check      : Check contents of a KMOS fits-file
kmo_fits_stack      : Creates KMOS conform fits-files
kmo_fits_strip      : Strip noise and/or rotator extensions from a processed
                    KMOS fits frame
kmo_fit_profile     : 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
kmo_flat            : Create master flatfield frame and badpixel map to be
```



SPARK INstructional Guide for KMOS data

Doc No:	VL-TRE-KMO-146611-009
Version:	0.5
Author	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

```
used during science reduction
kmo_illumination      : Create a calibration frame to correct spatial
                        non-uniformity of flatfield.
kmo_make_image        : Collapse a cube to create a spatial image
kmo_multi_reconstruct : Combine reconstructed cubes
kmo_noise_map         : Generate a noise map from a raw frame
kmo_reconstruct       : Performs the cube reconstruction using different
                        interpolation methods.
kmo_rotate            : Rotate a cube spatially
kmo_sci_red           : Reconstruct and combine data frames dividing
                        illumination and telluric correction.
kmo_shift             : Shift a cube spatially
kmo_sky_mask          : Create a mask of spatial pixels that indicates which
                        pixels can be considered as sky.
kmo_stats             : Perform basic statistics on a KMOS-conform fits-file
kmo_std_star          : Create the telluric correction frame.
kmo_wave_cal          : Create a calibration frame encoding the spectral
                        position (i.e. wavelength) of each pixel on the
                        detector.
```

Not all of the recipes are required to run the pipeline; some aim instead to provide useful tools for manipulating KMOS data, which can otherwise be awkward due to the use of numerous extensions.

Detailed help on any individual recipe (an outline of its purpose, a list of input files required, a list of the output files produced, and a description of the various optional parameters) can be found by, for example:

```
> esorex -man kmo_flat
```

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

NAME

```
kmo_flat -- Create master flatfield frame and badpixel map to be used during
           science reduction
```

SYNOPSIS

```
esorex [esorex-options] kmo_flat [kmo_flat-options] sof
```

DESCRIPTION

```
<blah>
```

2.2.2 Static Calibration Files

The KMOS data reduction recipes require a number of calibrations that should not need to change. These include, for example, arc-line lists, look-up tables etc. The user should confirm that these are available. A full list of these is:

```
kmos_ar_ne_list_h.fits    kmos_atmos_h.fits      kmos_solar_h_2400.fits
kmos_ar_ne_list_hk.fits  kmos_atmos_hk.fits     kmos_solar_hk_1100.fits
kmos_ar_ne_list_iz.fits  kmos_atmos_iz.fits     kmos_solar_k_1700.fits
kmos_ar_ne_list_k.fits   kmos_atmos_k.fits
kmos_ar_ne_list_yj.fits  kmos_atmos_yj.fits

kmos_spec_type.fits      kmos_wave_band.fits    kmos_wave_ref_table.fits
```



2.2.3 SPARKplug

Most recipes require a 'set of files' (sof) as input. The SPARKplug is a graphical data organizer that assists in preparing these lists, and insuring that all the necessary calibration files are included. It is not required – and is not part of the official pipeline release – but does make this step easier. It is started with the command below, with directories for the raw and calibration files set as examples

```
> SPARKplug.pl -cal=/share/KMOScalib -raw=/share/KMOSdata
```

The SPARKplug has sufficient file sorting capability to make this task relatively straightforward as long as the static calibration files, the raw data files, and the processed calibration products are kept in appropriate directories. One just needs to decide for which recipe the sof list is required, and the SPARKplug will show the set of appropriate raw and calibration files from which to choose. Being able to sort the files by name, band, etc, makes this very quick and simple. Files can be selected and de-selected. The sof list can be edited manually, and saved, or used directly with the recipe.

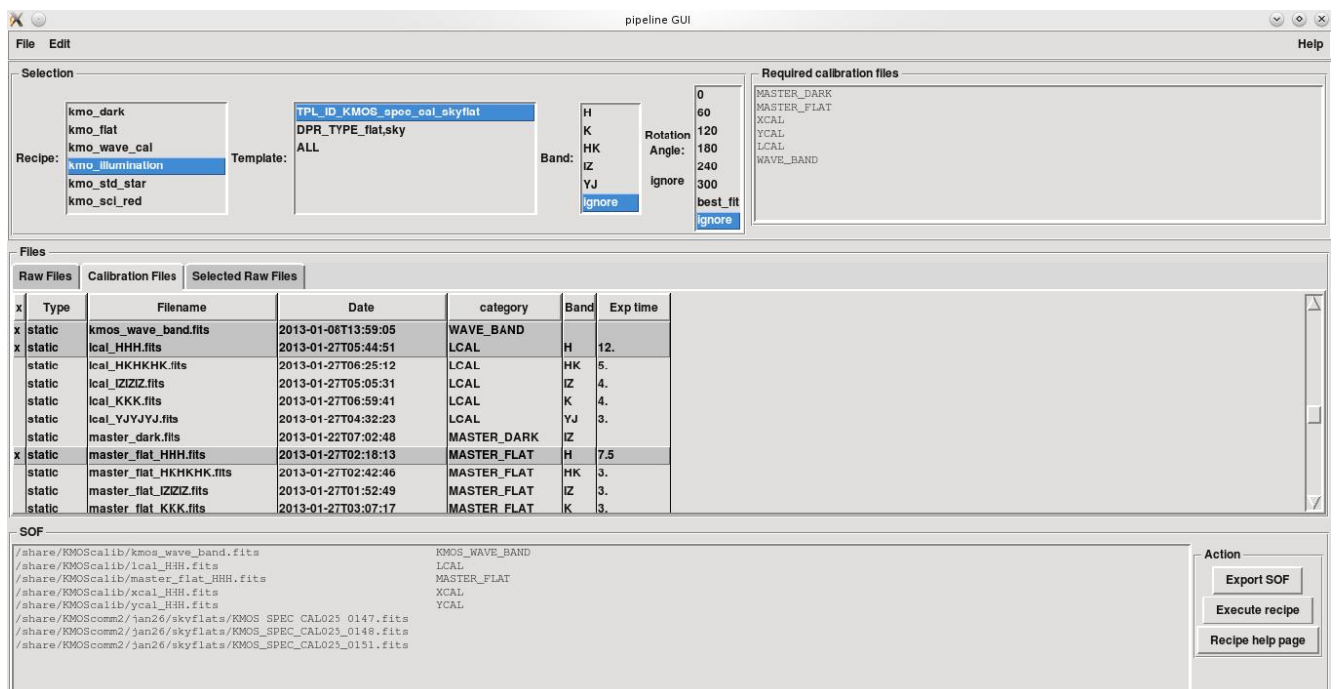


Figure 1: The SPARKplug tool can assist in preparing the 'set of files' required by most recipes.

2.2.4 easySPARK scripts

Normally calibration data is obtained in a well-defined standard procedure and creating sof-files to reduce these is therefore a quite repetitive task. These scripts aim to create sof-files and run ESOREX on them in a fairly automated manner.

All the scripts require a single file (path and name) as input and extract the other associated exposures via the `TPL.START` keyword, which is identical for all exposures generated in a single template. Furthermore, the environment variable `KMOS_CALIB` has to be set to a path containing the static calibration files (see Sec. 2.2.2). If any dynamic calibration file (like e.g. `XCAL`) is not found in the working directory, `KMOS_CALIB` is queried as well.

In order to obtain help on a specific script, just execute the script without an argument. If just the sof-files should be created without running ESOREX, just provide `sof` as additional parameter.

For the calibration recipes, the following scripts are available:

```
easySAPRK_dark.sh
easySAPRK_flat.sh
easySAPRK_wave_cal.sh
easySAPRK_illumination.sh
easySAPRK_std_star.sh
```

```
easySAPRK_calibration.sh
```

Unifies dark, flat and wave_cal. Here, because the keyword OBS.START is also examined, the script only works when all the calibration files have been generated in a single observation block (which is normally the case).

For standard use-cases there exist as well scripts which are rather self-explanatory:

```
easySPARK_reconstruct.sh
easySPARK_mapping.sh (to come)
etc.
```

3 Handling KMOS data

3.1 Data Format and Properties

The KMOS instrument has 3 similar segments, and so each exposure yields 3 frames. The data are stored in fits extensions. Since each segment has 8 IFUs, the reconstructed data will have 24 extensions (or 48 if noise is propagated). One can quickly see how many extensions a file has and what format the data is stored as, using:

```
> esorex kmo_fits_check KMOS_SPEC_CAL022_0004.fits

<blah>
+++++
FORMAT:          RAW
NAXIS:           2
NAXIS1:          2048
NAXIS2:          2048
NOISE:           FALSE
BADPIX:          FALSE
NR. EXT:         3 (excluding primary header)
  NR. DATA:     3
  NR. NOISE:     0
  NR. BADPIX:    0

VALID RAW FILE!
+++++
[ INFO ] esorex: [tid=000] 0 products created
```

If you want to view the data, we recommend using QFitsView, which can be downloaded from <http://www.mpe.mpg.de/~ott/QFitsView>. Other viewers that are able to read extensions can also be used.



3.2 Header Keywords

KMOS data has a lot of keywords, both in the primary header and the extension headers. We recommend using `dfits` and `fitsort` (both part of the `qfits` package from ESO) to list relevant keywords in the data. An example of usage is:

```
> dfits -x 0 KMOS*fits | fitsort tpl.id det.seq1.dit ins.filt1.id ocs.rot.naangle
```

FILE	TPL.ID	DET.SEQ1.DIT	INS.FILT1.ID	OCES.ROT.NAANGLE
KMOS_SPEC_CAL022_0001.fits	KMOS_spec_cal_stdstar	20.0000000	H	-13.593
KMOS_SPEC_CAL022_0008.fits	KMOS_spec_cal_stdstar	20.0000000	YJ	9.478
KMOS_SPEC_DARK021_0016.fits	KMOS_spec_cal_dark	10.0000000	Block	-60.000

Or to list all QC parameters in a processed file:

```
> dfits -x 0 cube.fits | grep QC
```

For these data, the '-x 0' is important since it will then look at the headers in all extensions.

A list of the most useful keywords in the RAW frames, and where to find them, is:

<i>keyword in RAW frame</i>	<i>location</i>	<i>description</i>
<code>dpr.type</code>	p	Type of observation (e.g. object,sky / flat,lamp / dark / etc)
<code>obs.start</code>	p	Date/time at which the OB was started
<code>tpl.start</code>	p	Date/time at which the template (within the OB) was started
<code>tpl.id</code>	p	Name of template used for observations
<code>date-obs</code>	p	Date/time at which exposure was started
<code>obs.id</code>	p	Unique identifier for OB
<code>obs.targ.name</code>	p	Name (from KARMA catalogue) of observation target
<code>det.seq[1-3].dit</code>	p	Integration time for detector [1-3]
<code>det.ndit</code>	p	Number of integrations averaged during exposure
<code>det.ndsamples</code>	p	Number of non-destructive samples during integration
<code>ins.filt[1-3].id</code>	p	Name of filter [1-3] (IZ / YJ / H / HK / K / Block)
<code>ins.grat[1-3].id</code>	p	Name of grating [1-3] (IZ / YJ / H / HK / K)
<code>ins.lamp1.st</code>	p	Keyword only included if status of argon lamp is ON
<code>ins.lamp2.st</code>	p	Keyword only included if status of neon lamp is ON
<code>ins.lamp3.st</code>	p	Keyword only included if status of flatfield lamp is ON
<code>ocs.rot.offangle</code>	p	Orientation of KMOS field wrt North
<code>ocs.rot.naangle</code>	p	Orientation of KMOS instrument wrt Nasmyth platform
<code>tel.parang.[start/end]</code>	p	Parallactic angle at start / end of exposure
<code>tel.airm.[start/end]</code>	p	Airmass at start / end of exposure
<code>tel.targ.alpha</code> <code>ocs.targ.alpha</code>	p	Right ascension of assigned telescope pointing (field centre). KARMA defines 2 field centres.
<code>tel.targ.delta</code> <code>ocs.targ.delta</code>	p	Declination of assigned telescope pointing (field centre). KARMA defines 2 field centres.
<code>ocs.arm[1-24].alpha</code>	p	Right ascension of pointing assigned to arm [1-24]. KARMA defines 2 pointings for each arm, associated with the 2 field centres.
<code>ocs.arm[1-24].delta</code>	p	Declination of pointing assigned to arm [1-24]. KARMA defines 2 pointings for each arm, associated with the 2 field centres.
<code>ocs.arm[1-24].name</code>	p	Name of target assigned to arm [1-24]
<code>ocs.arm[1-24].notused</code>	p	Keyword only present if arm is not used



ocs.targ.ditha	p	Relative offset (right ascension) of dither position with respect to the current defined pointing, in arcsec.
ocs.targ.dithd	p	Relative offset (declination) of dither position with respect to the current defined pointing, in arcsec.
ocs.arm[1-24].type	p	Type of exposure for this arm (O / S / R for object / sky / reference)
ocs.stdstar.mag	p	Magnitude of standard star, if it is given in the template. Applies only to files created with the stdstar templates.
ocs.stdstar.type	p	Spectral type of standard star, if it is given in the template. Applies only to files created with the stdstar templates.
extname	x	Name of extension: CHIP[1-3].INT1
det.chip.gain	x	Gain in e- per ADU of chip (=2.1)
naxis	x	Dimension of data in the extension
naxis[1-n]	x	Size of data axis [1-n]

A list of the most useful keywords in the pipeline products, and where to find them, is given below. QC parameters are excluded from this list, and instead a selection of the most useful ones is given in each of the respective sections of this document. To see all QC parameters in the header, use:

```
> dfits -x 0 product.fits | grep QC
```

<i>keyword in processed frame</i>	<i>location</i>	<i>description</i>
extname	x	Name of extension, e.g. IFU.1.DATA / IFU.1.NOISE / etc.
pro.catg	p	Type of product
pro.rot.naangle	x	Orientation of KMOS instrument (wrt Nasmyth platform) associated with extension; especially useful for master_flat.

3.3 IFU orientation and pixel arrangement

Across the detectors, the IFUs are numbered sequentially from left to right, across detector 1 to 3. The order (from left to right across a detector) of the spatial pixels within a slitlet, and the slitlets within an IFU, is more complex. These are arranged as shown below for the 24 IFU fields. The effect of this arrangement can be seen in the raw data and also sometimes in the reconstructed cubes. The spectral axis is approximately aligned with the columns. Long wavelengths are at the bottom of the detectors; short wavelengths at the top.

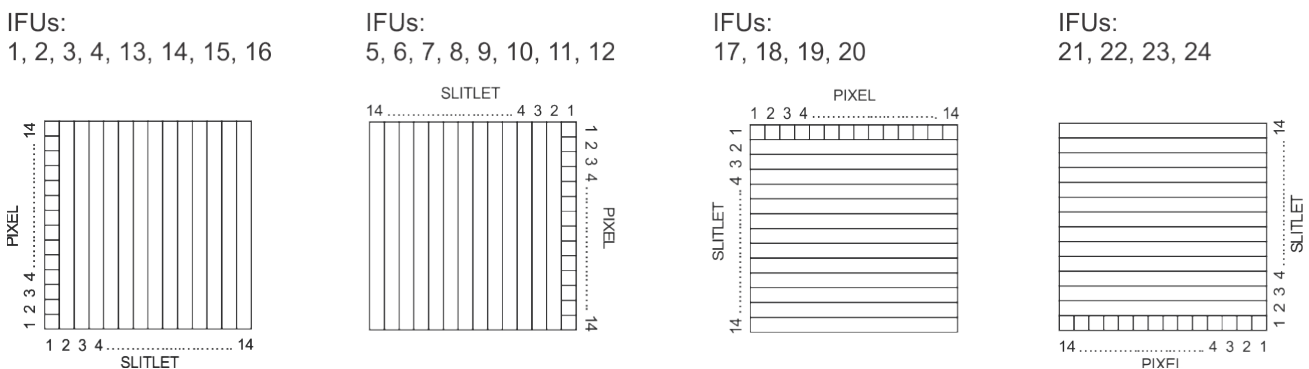



Figure 2: Order (from left to right on the detector) of the spatial pixels and slitlets in each IFU.

	SPARK INstructional Guide for KMOS data	
	Doc No:	VLT-TRE-KMO-146611-009
	Version:	0.5
	Author:	R.Davies, A. Agudo Berbel, E. Wiezorrek
	Date:	31.01.13

If KMOS is oriented to north (`ocs.rot.offangle = 0`), then the IFUs will all have north up and east to the left. If the offset angle is non-zero, then all the IFU fields are rotated by that angle.

The image quality across the slitlets is excellent, and is a true representation of the seeing. The image quality along the slitlets is affected by the KMOS optics and adds, in quadrature, about 0.2" to the resolution. This is most noticeable in the best seeing conditions.

Note that the image quality of IFUs 23 and 24 is not quite as good as the others; while the spectral resolution at the short end of IFUs 1-8 is slightly poorer than the rest.

3.4 Impact of Flexure

Like most instruments that have to cope with a changing gravity vector, KMOS has flexure. This has been quantified and amounts to about ± 1 pixel both spectrally (shifting the data up and down columns of the detector) and spatially (shifting the data across the rows of the detector). To account for this, standard calibrations are taken at 6 equally-spaced rotator angles. In addition, at the end of each night, calibrations are taken at a set of (at most 6) angles best suited to the observations that have been done. When processing data, the pipeline automatically selects the calibration at the closest available rotator angle to the data. This process is completely transparent to the user.

The impact of the residual flexure is relatively small, and so is ignored by the monolithic pipeline (described in Section 5.1).

Residual spectral flexure is measured from the OH lines on each individual frame, and corrected, by the advanced reconstruction recipe. This is a 2-step process and so does not lead to additional interpolations.

Residual spatial flexure, and also its non-repeatability which is of order 0.2 pixel, cannot generally be easily measured (the only exception being the sky flats, and the `kmo_illumination` recipe described in Section 4.4 does account for it). We expect that in most cases this should have little impact on the data. However, it may lead to edge effects, apparent as a slightly brighter or darker row of pixels along the slitlet edges. If this is critical to the analysis of the data, it may be possible to cross-correlate the raw data with the flatfield, and hence to match-up the slitlet edges on the detector frames.

4 Processing Calibrations

To create a complete set of processed calibrations, the recipes should be executed in the order given below because the later recipes make use of products from earlier ones. In addition, a consistent set of frames should be used, so that the set of files (`sof`) grows consistently as one progresses through the recipes. Note that frames that do not belong to a recipe are ignored, so there is no harm in having them propagated.

First set up a 'reduction session' by creating an appropriate directory structure. In the examples shown, the path `/share` is used, and the directories to create are:

<code>KMOScalib</code>	for static and processed calibration products
<code>KMOSdata</code>	contains all the raw data files (or links to them)
<code>KMOSscience</code>	will contain the processed products from the science observations



We recommend creating environment variables `KMOS_CALIB` and `KMOS_DATA` since the first is used in the easySPARK scripts and both of them can anyway be used in manually created sof-files.

Once this is done, copy the static calibrations into `KMOScalib` (or make links to them), move into `KMOScalib`, and run the calibration recipes as described below

HINTS

- Processing calibrations from multiple bands can take several hours. We recommend that the user creates all the ‘set of files’ (sof) lists first, and then sets all the recipes running overnight. If the directory structure above is followed, and `KMOScalib` is used as the working directory, then all the calibration products will appear there too, ready for the subsequent recipes.

4.1 Darks

Create a file called, for example, `dark_60s.sof` which contains a list of at least several dark exposures with the same exposure time (`det.seq1.dit` and `det.ndit`), together with the identifier `DARK`. The file will look something like this (but typically with 5 `DARK` frames):

```
> cat dark_60s.sof
/share/KMOSdata/KMOS_SPEC_DARK018_0012.fits      DARK
/share/KMOSdata/KMOS_SPEC_DARK018_0013.fits      DARK
/share/KMOSdata/KMOS_SPEC_DARK018_0014.fits      DARK
```

If you have set the environment variable `KMOS_DATA` then this can also be written as

```
$KMOS_DATA/KMOS_SPEC_DARK018_0012.fits      DARK
$KMOS_DATA/KMOS_SPEC_DARK018_0013.fits      DARK
$KMOS_DATA/KMOS_SPEC_DARK018_0014.fits      DARK
```

Then execute the `kmo_dark` recipe:

```
> esorex kmo_dark dark_60s.sof
```

This will create `master_dark.fits`, and a preliminary bad pixel mask `badpixel_dark.fits` that is used by `kmo_flat`.

HINTS

- If you want to rename the files, e.g. to include the exposure time in the name, you need to do this yourself
- Dark frames can be identified either from the file name or from the `dpr.type` keyword as `DARK`
- Ignore the `ins.grat[1-3].id` keyword in dark frames – it has no meaning for them
- Having said this, dark frames can be reconstructed even though there is no associated waveband (see Section 6).
- The parameters `pos_bad_pix_rej` and `neg_bad_pix_rej` can be used to adjust the sigma level at which pixels are flagged as bad; e.g.

```
> esorex kmo_dark -pos_bad_pix_rej=25 dark_60s.sof
```
- The dark current is extremely low, and cannot be measured even in frames of several hundred seconds.
- The readnoise is lowest for exposure times of about 300 sec.



4.2 Flats

Create a sof list containing the lamp on and lamp off flatfield frames, as well as the preliminary bad pixel mask – together with their identifiers. Note that due to flexure, flats and arcs are usually taken at



SPARK INstructional Guide for KMOS data

Doc No:	VLT-TRE-KMO-146611-009
Version:	0.5
Author	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

6 rotator angles so the list may be long. To make a list of the appropriate raw frames with relevant information, use a command like

```
> dfits KMOS*CAL*fits | fitsort tpl.id ins.filt1.id det.seq1.dit ocs.rot.naangle | grep calunitflat
```

The resulting sof may look like this (but with 3 `FLAT_OFF` and 18 `FLAT_ON` frames)

```
> cat flat_k.sof
/share/KMOScalib/badpixel_dark.fits          BADPIXEL_DARK
/share/KMOSdata/KMOS_SPEC_CAL020_0020.fits  FLAT_OFF
/share/KMOSdata/KMOS_SPEC_CAL020_0035.fits  FLAT_ON
```

Note that because `KMOScalib` is the working directory, and the order of the files in the sof list does not matter, `flat_k.sof` could also look like this:

```
/share/KMOSdata/KMOS_SPEC_CAL020_0035.fits  FLAT_ON
badpixel_dark.fits                          BADPIXEL_DARK
/share/KMOSdata/KMOS_SPEC_CAL020_0020.fits  FLAT_OFF
```

Then execute the `kmo_flat` recipe:

```
> esorex kmo_flat flat_k.sof
```

Five files will be produced, which are tagged with the waveband used. The waveband tag is repeated 3 times, once for each of the instrument segments.

```
master_flat_###.fits      Normalised flatfields, typically with 36 extensions (data and noise
                           frames for each detector, and for 6 rotator angles).

xcal_###.fits,           Frames containing the x and y coordinates within an IFU (an integer
ycal_###.fits            given in milli-arcsec from the centre of that IFU field) for every
                           illuminated pixel on the detector. The number after the decimal point is
                           the IFU identification. These frames have 18 extensions (3 detectors, 6
                           rotator angles).

badpixel_flat_###.fits   Map of bad or non-illuminated pixels (18 extensions as above).

flat_edge_###.fits      Coefficients of fits to the left and right edges of the slitlets in the IFUs
                           (144 extensions for 24 IFUs and 6 rotator angles).
```

HINTS

- You should not need to rename any of these files.
- Flatfield frames can be identified with the `dpr.type` keyword as `FLAT`, `LAMP` and `FLAT, OFF`
- Make sure you use frames taken together as a set, rather than mixing data from different dates.
- It is planned that the flat and arc calibrations taken after a night will match the rotation angles used during that night. To get the best results, one should use this matching set of flats and arcs to process the data.
- The flatfield illumination is not uniform, and so it is recommended to include an illumination correction when processing science data (see Section 4.4).
- Useful QC parameters include:

<code>QC.FLAT.SAT.NCOUNTS</code>	number of saturated pixels
<code>QC.FLAT.SN</code>	mean signal-to-noise of illuminated regions
<code>QC.SLIT.MEAN</code>	mean slit width in pixels



4.3 Arcs

Create a sof list containing the on/off arc-lamp frames, together with the required calibration products produced by `kmo_flat` and a few static calibration files. To make a list of the appropriate raw frames with relevant information, use a command like

```
> dfits KMOS*CAL*fits | fitsort tpl.id ins.filt1.id det.seq1.dit ocs.rot.naangle | grep cal_wave
```

The resulting sof may look like this (but with 6 `ARC_ON` frames for the 6 rotator angles). The order of the files does not matter, but they must be tagged correctly.

```
> cat arc_iz.sof
/share/KMOSdata/KMOS_SPEC_CAL020_0125.fits    ARC_OFF
/share/KMOSdata/KMOS_SPEC_CAL020_0130.fits    ARC_ON
/share/KMOScalib/badpixel_flat_IIZIZI.fits     BADPIXEL_FLAT
/share/KMOScalib/flat_edge_IIZIZI.fits         FLAT_EDGE
/share/KMOScalib/master_flat_IIZIZI.fits       MASTER_FLAT
/share/KMOScalib/xcal_IIZIZI.fits              XCAL
/share/KMOScalib/ycal_IIZIZI.fits              YCAL
/share/KMOScalib/kmos_ar_ne_list_iz.fits       ARC_LIST
/share/KMOScalib/kmos_wave_band.fits           WAVE_BAND
/share/KMOScalib/kmos_wave_ref_table.fits      REF_LINES
```

Then execute the `kmo_wave_cal` recipe:

```
> esorex kmo_wave_cal arc_iz.sof
```

Two files will be produced, which are tagged with the waveband used.

```
lcal_###.fits      Frame containing wavelength (in microns) for every illuminated pixel on
                   the detector. This frame has 18 extensions (3 detectors, 6 rotator angles).
det_img_wave_###.fits Reconstructed arc-lamp frame, reformatted so that slitlets and IFUs are
                   side-by-side (a pseudo detector image). This is wavelength calibrated, so
                   arc lines should exactly follow the rows, allowing one to quickly and
                   easily verify that the recipe has been successful.
```

HINTS

- Arclamp frames can be identified with the `dpr.type` keyword as `WAVE,LAMP` and `WAVE,OFF`
- Typically arcs are taken immediately after flats, and we would strongly recommend using those frames – to ensure a consistent set of calibration products.
- The order of the files in the sof list does not matter; and it is not necessary to specify the full path for files that are in the working directory. One can also make use of environment variables instead of writing out the full path each time.
- Useful QC parameters include:

<code>QC.ARC.SAT.NCOUNTS</code>	number of saturated pixels
<code>QC.ARC.AR.POS.MEAN</code>	mean offset (in km/s) of reference argon line
<code>QC.ARC.AR.FWHM.MEAN</code>	mean FWHM (in km/s) of reference argon line
<code>QC.ARC.NE.POS.MEAN</code>	mean offset (in km/s) of reference neon line
<code>QC.ARC.NE.FWHM.MEAN</code>	mean FWHM (in km/s) of reference neon line
- The 3 calibrations steps dark, flat and arc can be prepared in one go with `easy_calibration.sh` as long as a consistent set of calibration data is used.



4.4 Illumination Correction

Prepare a sof list containing the sky-flat frames, together with suitable dark frames, and the required calibration files. To make a list of the appropriate raw frames with relevant information, use a command like

```
> dfits KMOS*CAL*fits | fitsort tpl.id ins.filt1.id det.seq1.dit | grep skyflat
```

The `MASTER_FLAT` and `XCAL/YCAL/LCAL` files should match the wavelength of the sky flats. The resulting sof may look like this (but with typically 3 `SKY_FLAT` frames):

```
> cat skyflat_h.sof
/share/KMOSdata/KMOS_SPEC_CAL018_0209.fits      FLAT_SKY
/share/KMOScalib/master_dark.fits              MASTER_DARK
/share/KMOScalib/master_flat_HHH.fits          MASTER_FLAT
/share/KMOScalib/xcal_HHH.fits                 XCAL
/share/KMOScalib/ycal_HHH.fits                 YCAL
/share/KMOScalib/lcal_HHH.fits                 LCAL
/share/KMOScalib/kmos_wave_band.fits           WAVE_BAND
```

Then execute the `kmo_illumination` recipe:

```
> esorex kmo_illumination skyflat_h.sof
```

One file will be produced, which is tagged with the waveband used.

```
illum_corr_###.fits  Frame containing images of the internal flatfield uniformity for each IFU.
                    This frame has 48 extensions (data and noise for each of the 24 IFUs.)
```

HINTS

- Skyflat frames can be identified with the `dpr.type` keyword as `FLAT`, `SKY`.
- The first sky flat in a series is a test exposure to set the integration time. The subsequent 3 are the ones to use.
- The rotator angle does not matter since the flatfield spatial uniformity is independent of the orientation of the KMOS instrument.
- The parameter `range` can be used to specify a particular (set of) wavelength range(s) over which the illumination correction should be derived, e.g.

```
> esorex kmo_illumination -range='1.50,1.75' skyflat_h.sof
```
- Useful QC parameters include:

`QC.SPAT.UNIF` RMS spatial uniformity of the internal flatfield

4.5 Standard Stars

The final calibration is the standard star, and the recipe for this is basically a full science reduction (as in Section 5.1) with some extra bits added on. Reduction is the same for both the `KMOS_spec_cal_stdstar` and `KMOS_spec_cal_stdstarscipatt` templates. The only difference here is whether 3 or 24 IFUs are processed – and therefore whether there are 3 or 24 raw files (flagged as `STD`) in the sof list. Note that at least 2 `STD` frames are required to enable sky subtraction, but the recipe will also work if only one `STD` frame is provided. Begin by making the sof list, which will look something like this:

```
> cat std_hip012345_k.sof
/share/KMOSdata/KMOS_SPEC_CAL333_0023.fits      STD
/share/KMOSdata/KMOS_SPEC_CAL333_0024.fits      STD
/share/KMOSdata/KMOS_SPEC_CAL333_0025.fits      STD
/share/KMOScalib/xcal_KKK.fits                   XCAL
/share/KMOScalib/ycal_KKK.fits                   YCAL
/share/KMOScalib/lcal_KKK.fits                   LCAL
```



SPARK INstructional Guide for KMOS data

Doc No:	VLT-TRE-KMO-146611-009
Version:	0.5
Author	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

```
/share/KMOScalib/master_flat_KKK.fits      MASTER_FLAT
/share/KMOScalib/illum_cor_KKK.fits       ILLUM_CORR
/share/KMOScalib/kmos_wave_band.fits      WAVE_BAND
```

Then execute the `kmo_std_star` recipe:

```
> esorex kmo_std_star -save-cubes std_hip012345_k.sof
```

When the `save-cubes` option is set, the reduced cubes will be written to file, so that 5 files are created. Writing out the cubes allows you to extract the spectra yourself if, for example, you want to use a different aperture or to do additional cosmetic cleaning on the cubes.

```
std_cube_###.fits      Frame containing cubes of the standard star. There are 48 extensions (data
                        and noise for 24 IFUs), but not necessarily all will contain data.
std_image_###.fits     Collapsed images of the standard stars (24 extensions). Only the extensions
                        for IFUs used to observe the star will contain data.
std_mask_###.fits     Spatial masks showing which pixels were used to extract the spectrum
                        (which are those within the FWHM).
star_spec_###.fits    The extracted (integrated) spectra.
(telluric_###.fits    The derived telluric correction spectrum – see Section 4.5.2)
```

HINTS

- Standard star frames can be identified with the `dpr.type` keyword as `OBJECT`, `SKY`, `STD`, `FLUX` (all one designation).
- The recipe selects a sky exposure for each IFU independently (the closest in time to the star exposure), from among the exposures given in the `sof` list. This is reported in the output text.
- The recipe will choose the calibrations at the closest available rotator angle (`OBS.ROT.NAANGLE`) to the observations.
- Reconstruction is done using cubic-spline method (see Section 6.2) by default. We would recommend not to change this for the standard star.
- Flux calibration (see Section 4.5.1) is performed using the total flux in the IFU; but the extracted spectrum is integrated only from pixels within the measured FWHM (which encloses about half the total flux).
- Useful QC parameters include:

```
QC.SPAT.RES          FWHM of the star in arcsec (in std_image_###.fits)
```

4.5.1 Flux Calibration

If a magnitude for the star is given, the same recipe will perform a flux calibration and calculate the zeropoint. The magnitude should match the band used for the observations and can be set in the template using `P2PP`, in which case this calculation is done automatically. You can check this by looking for the magnitude keyword:

```
> dfits KMOS_SPEC_CAL026_0029.fits | fitsort ocs.stdstar.mag
```

Alternatively, it can be set as a parameter when executing the recipe (which will override the magnitude keyword):

```
> esorex kmo_std_star -save-cubes -mag=6.61 std_hip012345_k.sof
```

Note that for the HK band, the magnitudes for both bands should be given (H first, K second) separated by a comma with no spaces:

```
> esorex kmo_std_star -save-cubes -mag='6.71,6.61' std_hip012345_hk.sof
```


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

$$\text{mag} = \text{qc.zpoint} - 2.5 \log_{10}(\text{cts/sec})$$

where `mag` is the magnitude of a source that has a mean count rate of `cts/sec` per spectral pixel. You can then convert the magnitude to a flux density. If you want a line flux, integrate the counts over the line and multiply the result by the spectral size of a pixel (given by the `CDELTA3` keyword in the cubes or the `CDELTA1` keyword in the extracted spectra).

Near-infrared magnitudes for stars are widely available from 2MASS. And so for estimating the zeropoint in the YJ, J, HK, and K bands, the 2MASS bandpasses are used. In addition, 2MASS zero magnitude flux densities are used for the throughput estimates. These are taken from Cohen, Wheaton, & Megeath (2003; AJ, 126, 1090). Since the *z* band is poorly defined, for the IZ band we use a pseudo-monochromatic 1 μm flux density. One way to estimate this is to interpolate it from the KHJIR magnitudes, where the latter 2 come from the USNO-B1 catalogue. The parameters used for KMOS are summarised in the table below.

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

4.5.2 Telluric Calibration

If the spectral type is provided then it may be possible to create a normalised telluric spectrum. This can be set in the template using P2PP. You can check this by looking for the spectral type keyword:

```
> dfits KMOS_SPEC_CAL026_0029.fits | fitsort ocs.stdstar.type
```

Alternatively, it can be set as a parameter when executing the recipe:

```
> esorex kmo_std_star -save-cubes -startype='B8III' std_hip022112_k.sof
```

There are only a limited number of cases for which this software attempts to make a telluric spectrum:

G (ideally G2V) stars in the H, HK, or K bands: the recipe will divide out a solar spectrum and correct for the blackbody temperature associated with the spectral type.

O, B, A, and F stars in any band: the recipe will fit and subtract the strongest H and He absorption lines (making use of an approximate atmospheric model to help).


For all other cases, you will have to do this step yourself.

To process the data for G stars, you will need to add the following 2 lines to the sof list:

```
/share/KMOScalib/kmos_solar_h_2400.fits SOLAR_SPEC
/share/KMOScalib/kmos_spec_type.fits SPEC_TYPE_LOOKUP
```

To process the data for OBAF stars, you will need to add the following 2 lines to the sof list:

```
/share/KMOScalib/kmos_atmos_k.fits ATMOS_MODEL
/share/KMOScalib/kmos_spec_type.fits SPEC_TYPE_LOOKUP
```

	SPARK INstructional Guide for KMOS data		Doc No:	VLT-TRE-KMO-146611-009
			Version:	0.5
			Author:	R.Davies, A. Agudo Berbel, E. Wieszorrek
			Date:	31.01.13

If you decide to try this, the recipe will create an additional product:
`telluric_###.fits` The normalised telluric spectrum.

5 Science Reduction

Once you have a full set of calibrations, move into the `KMOSscience` directory to process the science frames. This can be done either using the monolithic pipeline (Section 5.1) which is the most straightforward way; or using the recipes one at a time (5.2) which gives more flexibility, as well as enabling you to use your own routines – which may be extra processing steps or replacements for pipeline recipes.

Before starting work on the science observations, move into `KMOSscience`, which is now your working directory.

5.1 Monolithic pipeline

This recipe performs all the standard processing steps: sky subtraction, flat fielding, illumination correction, reconstruction, telluric correction, shifting, and finally combining. It is straightforward to use, but does not allow the user much flexibility. To use this recipe, the first step is, as for the calibrations, to create a sof list. This can contain observations from several OBs, and will look something like:

```
> cat sci_obs.sof
/share/KMOSdata/KMOS_SPEC_OBS023_0006.fits      SCIENCE
/share/KMOSdata/KMOS_SPEC_OBS023_0007.fits      SCIENCE
/share/KMOScalib/xcal_YJYJYJ.fits               XCAL
/share/KMOScalib/ycal_YJYJYJ.fits               YCAL
/share/KMOScalib/lcal_YJYJYJ.fits               LCAL
/share/KMOScalib/master_flat_YJYJYJ.fits        MASTER_FLAT
/share/KMOScalib/illum_cor_YJYJYJ.fits          ILLUM_CORR
/share/KMOScalib/telluric_YJYJYJ.fits           TELLURIC
/share/KMOScalib/kmos_wave_band.fits            WAVE_BAND
```

All the observations are called `SCIENCE`, with no differentiation between sky and object. This is because any particular frame may include both object and sky data, depending on the arm assignments.

The `ILLUM_CORR` and `TELLURIC` files are optional. If `ILLUM_CORR` is omitted, there will be no correction for spatial uniformity of the internal flatfield; if `TELLURIC` is omitted, there can be no correction for atmospheric transmission.

Then execute the `kmo_sci_red` recipe:
`> esorex kmo_sci_red sci_obs.sof`

There are 2 sets of output:

`sci_reconstructed_###.fits` Processed and reconstructed cubes, matching the input `SCIENCE` files. The tag in this case is the name of the input file. Only input files with at least one ‘object’ IFU (i.e. `OCS.ARMi.TYPE='O'`) will appear as an output file; and in these files, only ‘object’ IFUs will be processed, the other extensions will be empty.



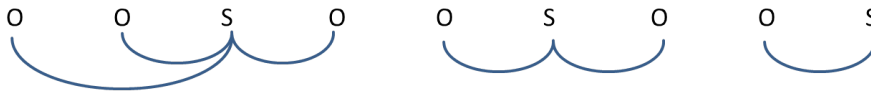
Doc No:	VL-TRE-KMO-146611-009
Version:	0.5
Author:	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

sci_combined.fits

A single file containing the final set of cubes, constructed by shifting and combining the data for each IFU.

HINTS

- Science frames can be identified with the `dpr.type` keyword as `OBJECT`, `SKY` (all one designation).
- For each IFU, object and sky exposures are distinguished by the `OCS.ARMi.TYPE` keyword. For each object exposure in an IFU, the pipeline automatically selects the sky exposure in the same IFU taken closest in time (and reports this in the log). An example of this way of deciding the Object/Sky pair assignments for a sequence of frames is:



This is not necessarily what one might choose, so it will soon be possible for the user to edit this 'object/sky assignment table'. Note that the templates provide sequences such as: OS OS OS, OSO OSO OSO, OOSOO OOSOO, etc.

- The default interpolation method is cubic spline. Depending on the circumstances, other methods may be preferred – see Section 6.2.
- If you are interested in only 1 object or only in specific IFUs, and want to save time processing, these can be specified as parameters.

```
> esorex kmo_sci_red -name='gal21' sci_obs.sof
```

will process only IFUs labelled with `OCS.ARM[1-24].NAME='gal21'`.

```
> esorex kmo_sci_red -ifus="3;14;3;14" sci_obs.sof
```

will process only IFU 3 from the 1st SCIENCE frame, IFU 14 from the 2nd, IFU 3 again from the 3rd, and IFU 14 again from the 4th. There must, of course, be exactly 4 SCIENCE exposures given. In both cases, sky frames are identified as before.
- Various options are available for specifying the offsets when shifting the cubes (see Section 5.2 for details). This is done with the `method` parameter, which can be set to `'header'`, `'none'`, `'center'`, or `'user'`. For example, if you are processing data taken at different times, and the sources are clearly visible in individual exposures, you may prefer to derive shifts from the sources themselves. In this case you might try:

```
> esorex kmo_sci_red -method='center' sci_obs.sof
```
- The parameters and input files used by this, or any other, recipe to generate the output files can be found by looking for `PRO` keywords in the primary header of the products:

```
> dfits sci_reconstructed_KMOS_SPEC_OBS023_0006.fits | grep PRO
```

5.1.1 Mapping & Mosaics

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. This means that they can be reduced by the monolithic pipeline with the single command

```
> esorex kmo_sci_red sci_obs.sof
```

as given above. However, the user should note that the pipeline makes no attempt to perform background matching between the individual IFU pointings, nor to deal with any edge effects. These rather complex tasks are left to the user, since how they are done depends on the individual data set.

It is often useful to know which IFUs in which exposures makes up the various parts of the patchwork mosaic. Figure 3 and Figure 4 show this information for the 8-arm and 24-arm mapping modes respectively.

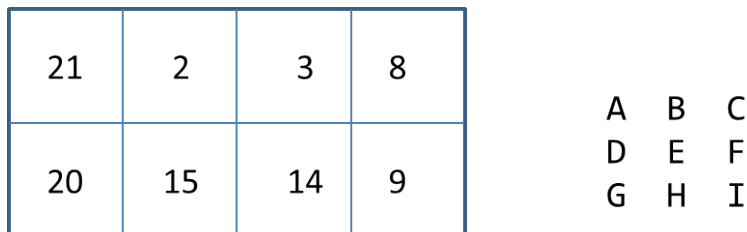


Figure 3: Left – Arrangement of the IFUs used for the Mapping8 mosaic mode. Right – order (from A to I) of the 9 dithers performed during the Mapping8 mode. The IFUs are separated by 8.1” and each dither is 2.7” so that, at the end, there is a 0.1” (half-pixel) overlap between adjacent pieces.

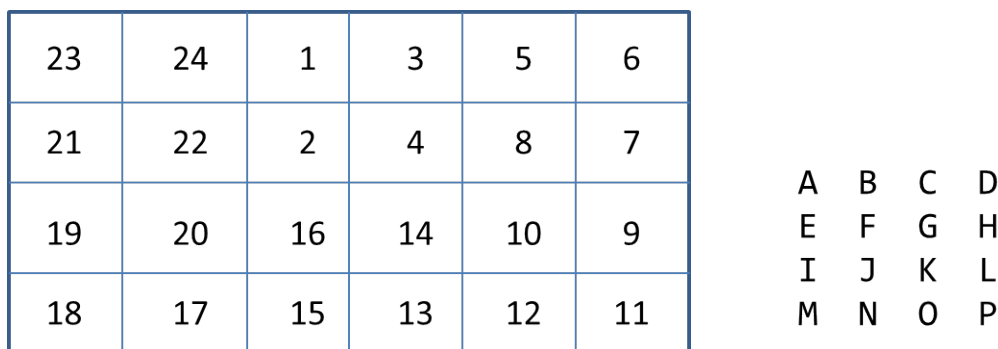


Figure 4: Left – Arrangement of the IFUs used for the Mapping24 mosaic mode. Right – order (from A to P) of the 16 dithers performed during the Mapping24 mode. The IFUs are separated by 10.8” and each dither is 2.7” so that, at the end, there is a 0.1” (half-pixel) overlap between adjacent pieces.

5.2 Work-flow one step at a time

The monolithic pipeline performs the standard steps for a scientific reduction. These steps can be performed one at a time. The following example shows how.

First find out the Nasmyth angle at which the observations were taken and extract the matching flatfield frames from the MASTER_FLAT. Here the -71° ($=289^\circ$) of the data most closely matches 300° in the calibration frames. The data at this angle are extracted using the recipe `kmo_fits_strip`.

```
> dfits KMOS_SPEC_OBS025_0034.fits | fitsort ocs.rot.naangle
FILE OCS.ROT.NAANGLE
KMOS_SPEC_OBS025_0034.fits -70.724
> esorex kmo_fits_strip -noise=TRUE -angle=300 master_flat_YJYJYJ.fits
> mv strip.fits flat_YJ_300.fits
```

Then for each object frame, subtract the sky, divide by the flatfield, apply the illumination correction, and reconstruct it. For the last step here, we use the shell script `easySPARK_reconstruct.sh` which simplifies things by automatically generating the required sof list and renaming the output file.

However you can also create the sof yourself and use `kmo_reconstruct`.

```
> esorex kmo_arithmetic -op='-`' KMOS_SPEC_OBS025_0034.fits KMOS_SPEC_OBS025_0035.fits
> mv arithmetic.fits frame0034s.fits
> kmo_arithmetic -op='/' frame0034s.fits flat_YJ_300.fits
> mv arithmetic frame0034sf.fits
> kmo_arithmetic -op='/' frame0034sf.fits illum_corr_YJYJYJ.fits
> mv arithmetic frame0034sfi.fits
```

You are now ready to reconstruct the data. To do this, create an sof list (e.g. called

`reconstruct_0034.sof`) which contains the following files:

```
frame0034sfi.fits OBJECT
```



SPARK INstructional Guide for KMOS data

Doc No:	VLT-TRE-KMO-146611-009
Version:	0.5
Author:	R.Davies, A. Agudo Berbel, E. Wiezorrek
Date:	31.01.13

```
$KMOS_CALIB/xcal_YJYJYJ.fits      XCAL
$KMOS_CALIB/ycal_YJYJYJ.fits      YCAL
$KMOS_CALIB/lcal_YJYJYJ.fits      LCAL
$KMOS_CALIB/kmos_wave_band.fits    WAVE_BAND
```

And then execute the 2 commands

```
> esorex kmo_reconstruct reconstruct_0034.sof
mv cube_object.fits cube_OBS025_035.fits
```

The method used for the interpolation can be specified using the `method` parameter, for example

```
> esorex kmo_combine -method='CS' reconstruct_OBS025_035.sof
```

A short discussion of the interpolation methods available is given in Section 6.2.

Once all the frames have been reconstructed, the cubes can then be combined by listing them in a sof (no tag is required), and executing `kmo_combine`. For this recipe, one can specify either which IFUs to combine (1 per file in the sof list) or an object name (the recipe then finds which IFUs have this object name):

```
> esorex kmo_combine -method='header' -name='gal21' -cmethod='median' objectcubes.sof
```

The `method` parameter specifies how the dither offsets should be determined; and the `cmethod` parameter indicates how the pixel values should be combined.

However, better results can often be achieved if the user includes a few special routines to optimise some of the steps. Some options are described in Section 5.3.

5.2.1 easySPARK_reconstruct

A shell script `easySPARK_reconstruct.sh` included in the KMOS kit provides a convenient way to reconstruct cubes by automatically generating the required sof list, and renaming the output file. To use it, first make sure the environment variable `KMOS_CALIB` is set to be the full path of the `KMOScalib` directory (in the examples here `/share/KMOScalib`), and check that `easySPARK_reconstruct.sh` is in your path (see Section 2.2.4). Then simply execute a command like:

```
> easySPARK_reconstruct.sh KMOS_OBS025_0103.fits
```

It will create an sof list called `reconstruct_OBS025_035.sof` which contains the following files:

```
KMOS_SPEC_OBS025_0035.fits      OBJECT
$KMOS_CALIB/xcal_YJYJYJ.fits      XCAL
$KMOS_CALIB/ycal_YJYJYJ.fits      YCAL
$KMOS_CALIB/lcal_YJYJYJ.fits      LCAL
$KMOS_CALIB/kmos_wave_band.fits    WAVE_BAND
```

And then perform the 2 commands

```
> esorex kmo_reconstruct reconstruct_OBS025_035.sof
mv cube_object.fits cube_OBS025_035.fits
```

The method used for the interpolation can be specified as a parameter, for example

```
> easySPARK_reconstruct.sh KMOS_OBS025_0103.fits CS
```

This script can also be used on pre-processed data (although watch out for how it re-names the output cube), for example:

```
> easySPARK_reconstruct.sh frame0034sfi.fits
```

5.3 Alternatives & Optimisation

5.3.1 Residual Sky Subtraction

If the atmospheric OH lines do not subtract out well, you will need to use a routine to apply a wavelength dependent scaling to a sky cube to correct the residuals. The method we use is described in Davies 2007 (MNRAS, 375, 1099). This will be implemented as a recipe, but is currently only available as an IDL script. Please contact the authors of this document if you wish to use it.

First you need to create a sky cube for all 24 IFUs, by subtracting a matched dark frame, flatfielding the result, applying the illumination correction, and reconstructing. The steps are exactly as described above in Section 5.2. Having done this, run the `skysubxtn.pro` script. Instructions are given with the script.

5.3.2 Wavelength Corrections

Flexure within KMOS as it rotates means that, even when applying the correction above, the OH lines do not always subtract out and can sometimes leave P-Cygni like residuals. If this happens, it is necessary to apply a wavelength correction to the spectral calibration. This will be implemented as a recipe that reconstructs each cube individually, measures the wavelength offset from the OH lines, applies this to the `LOCAL` calibration and then does the reconstruction again. This 2-step process means that the final cube has only been interpolated once.

However, currently, this procedure is only available as an IDL script that post-processes the reconstructed cubes. It is implemented as part of `skysubxtn.pro` mentioned above. Please contact the authors of this document if you wish to use it.

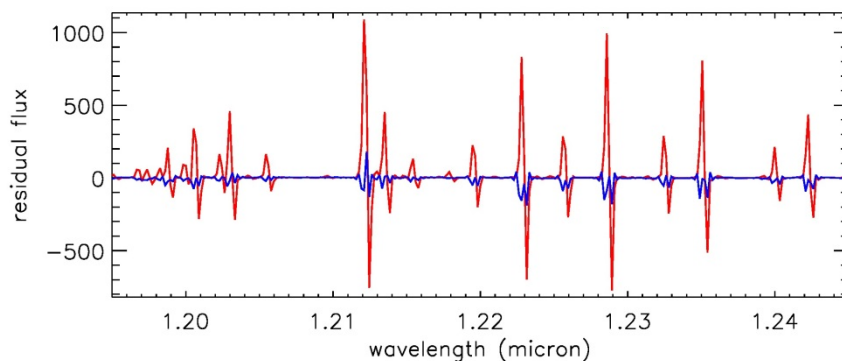


Figure 5: Example of spectrum segment without (red) and with (blue) application of the OH wavelength and scaling corrections described above.

5.3.3 Background Matching

The varying sky brightness may mean that there is a small residual background difference between exposures. If a source within a given IFU is either spatially or spectrally compact, it is possible to measure the background level and correct it.

Until a recipe is implemented, an IDL script called `kmossbkg.pro` is available to do this. Please contact the authors of this document if you wish to use it.

. CAUTION

This routine cannot be applied blindly, since its success depends on how much an object fills a data cube. In particular, it will not work with spatially extended continuum sources. The user must decide whether it is appropriate for their data.

5.3.4 Edge Effects

These may become apparent in some circumstances due to a slight mismatch between the position on the detectors of flatfield traces and science data (residual spatial flexure). A simple way to deal with this is to trim off the edges. You can either use `kmo_copy`, or just set them to NaN.

5.3.5 Improving Cosmetics

For a number of reasons, there may well be many deviant pixels in the reconstructed cubes. These can be effectively cleaned using a 3D version of van Dokkum's L.A.Cosmic routine (van Dokkum P., 2001; PASP, 113, 1420). We note that because of the strong OH lines in the raw data, and the possible presence of continuum sources, the routine is more effective (and safer to use) when applied to the reconstructed cubes.

An IDL script called `lac3dxtn.pro` is available for this. Please contact the authors of this document if you wish to use it.

CAUTION

While the routine has been tested successfully with its default parameters on a variety of sources, it is the user's responsibility to check that it removes only bad pixels and does not impact the source itself.

5.3.6 Weird Things

Do your reconstructed data have stripes like those shown in Figure 6?

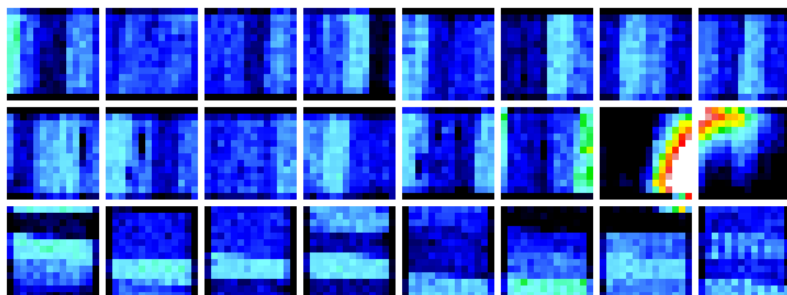


Figure 6: images created by collapsing cubes from one H-band exposure in a mosaic. IFUs 1-8 are from left to right along the top row; IFUs 9-16 along the middle row, and IFUs 17-24 along the bottom row. There are (parts of) sources in only IFUs 15 & 16. The striping effect (with a period of 3-4 slitlets) is apparent in nearly all of the other others. And in IFU 24, one can see an odd-even effect across a few slitlets.

This is caused by varying levels in the read-out channels of the detectors. The effect is only ~ 1 count or so, but is an issue when observing very faint sources. There may be a way to correct for this – but it will involve processing the data twice: once so that the effect can be measured; then a second time after it has been corrected (which has to be done as the first step).

6 Notes on Reconstruction

The `XCAL`, `YCAL`, and `LCAL` files contain all the information necessary to reconstruct a cube. They provide the (x,y,λ) location in the cube of each illuminated detector pixel. The x and y locations are integer distances in milliarcsec along the horizontal and vertical axes from the centre of each IFU field; the λ location along the spectral axis is given in microns. The IFU identification is encoded in the `XCAL` and `YCAL` frames as the number after the decimal point. These 3 files are used to perform reconstruction as described below. You can also use them with your own reconstruction algorithm.

Note: a variety of tags can be used for the file to be reconstructed. For example, dark frames can be reconstructed even though they have no associated waveband. The `DARK` tag will cause the recipe to skip the waveband checks so that it can use the calibrations provided.

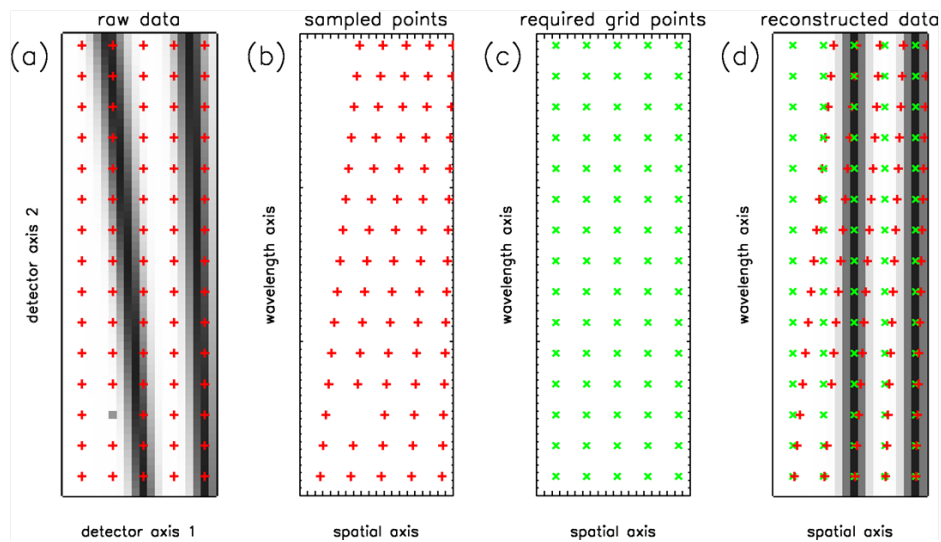


Figure 7: Concept for calibrating and interpolating, illustrated in 2D. (a) Observed data are sampled regularly (red points) on the detector frame; (b) These map to an irregular grid in the reconstructed cube (with bad pixels simply omitted from the set of valid data points); (c) The regular sampling in the final cube (green points) is, in principle, independent of the original sampling on the detector; (d) Each required grid point (green) is interpolated from the neighbouring sample points (red).

6.1 Interpolation Methods

Several interpolation methods are available. But if in doubt, use the default interpolation method which is cubic spline (CS). In this section, we write a few notes about the pros and cons of the various methods.

Nearest neighbour (NN) is a fast but approximate method, simply re-arranging the measured data values without actually performing any interpolation. It may be a good choice for noisy data (since it does not degrade the signal-to-noise at all) when the highest spatial or spectral resolution is not required. But points may be offset up to 0.5 pixels (spatially or spectrally) since this is how far away the neighbour can be – so the fidelity is poor.

Linear Distance Weighted Nearest Neighbours (lwNN) makes use of all pixels within a specified range – up to 26 neighbours for a neighbourhood range of 1 – and combines them, using weights that decrease linearly with distance. It is a simple method with simple error propagation, which yields reasonable results.

Square Distance Weighted Nearest Neighbours (swNN) is similar, but applies a weighting that depends on the inverse square of the distance, truncated at the edge of the neighbourhood range. The Modified Shepard's method (MS) is similar, but scales the weighting to zero at the edge of this range. This can be a good method if the spatial/spectral resolution elements are well enough sampled (e.g. if one combines data from many exposures during the reconstruction). Note that the swNN and MS methods lead to better spectral resolution but poorer wavelength fidelity than lwNN. Drizzling is expected to yield results much like these methods.

Cubic Spline (CS) actually performs 3 successive 1-D interpolations. As such, it is the only method that is not a true 3D interpolation. However, it gives the best results in terms of fidelity – in terms of maintaining the native resolution, and achieving precision. On the other hand error propagation is non-trivial, and is not done for this method.

6.2 Multi-reconstruct

The standard processing steps first reconstruct each cube, and then afterwards shift and combine them. Instead of performing this 2-step process, the calibration files `XCAL/YCAL/LCAL` allow one to put all raw data into a huge 'meta-detector' frame with their respective 'meta-calibrations' and reconstruct the entire dataset in one go. There may be some advantages to doing things this way. Most obviously, it avoids the additional interpolation during sub-pixel shifting, and it makes better use of dithered observations (which is especially important for the true 3D interpolation methods described above). If you want to experiment with this, then use the `kmo_multi_reconstruct` recipe.

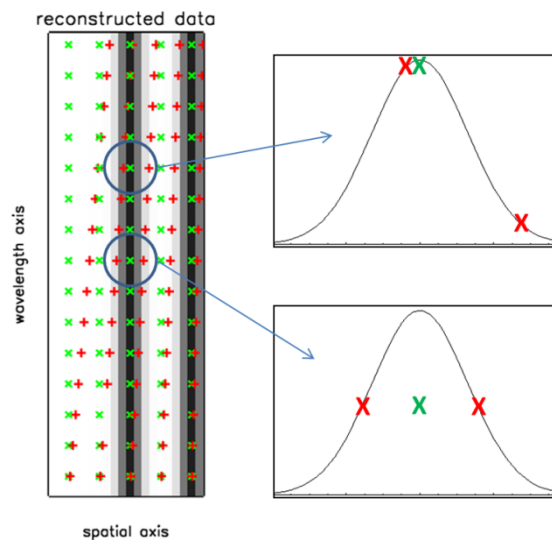


Figure 8: illustration of the major disadvantage of 3D interpolations. The upper circle shows a region where the interpolated point is close to a sample point; the lower circle shows the opposite situation where these are far from each other. In both cases, one is trying to interpolate the peak of a Gaussian trace; but the interpolated values are very different. Because 3D methods all make use of weighted sums of neighbouring pixels, the interpolated value can never be greater than the maximum of the surrounding sample points. For spatially compact continuum sources (e.g. stars), this can lead to a slow ripple pattern, or in the worst case several strong discontinuities, along the spectrum. This feature can only be overcome by better sampling, which is one advantage of the multi-reconstruct recipe.



7 Other Useful Recipes

The full list of recipes has already been given in Section 2.2.1. Here we highlight a few that might be useful.

7.1 Simple Mathematics

The recipe `kmo_arithmetic` has already been encountered in Section 5.2. It can be used in many situations. A few examples are given here. The output frame is always called `arithmetic.fits`.

Subtract a (raw) sky frame from an object frame:

```
> esorex kmo_arithmetic -op='-` objectframe.fits skyframe.fits
```

Divide the spectrum at each spatial position in cube by a telluric spectrum:

```
> esorex kmo_arithmetic -op='/' cube.fits telluric.fits
```

Add 2 cubes together:

```
> esorex kmo_arithmetic -op='+' cube1.fits cube2.fits
```

Raise a spectrum by some power, to account for differing airmass between object and standard star:

```
> esorex kmo_arithmetic -op='^' telluric.fits 1.1
```

Multiply a cube by a constant:

```
> esorex kmo_arithmetic -op='*' cube1.fits 6.3
```

7.2 Basic Statistics

Basic statistical properties of the data can be calculated using

```
> esorex kmo_stats KMOS_SPEC_CAL022_0004.fits
```


```
<blah>
```

```
[ INFO ] kmo_stats: [tid=000] -----  
[ INFO ] kmo_stats: [tid=000] |DET.1.DATA|DET.2.DATA|DET.3.DATA|  
[ INFO ] kmo_stats: [tid=000] 1. #pixels: | 4194304 | 4194304 | 4194304 |  
[ INFO ] kmo_stats: [tid=000] 2. #finite pix.: | 4194304 | 4194304 | 4194304 |  
[ INFO ] kmo_stats: [tid=000] 3. mean: | 100.0005 | 11.13203 | 8.771374 |  
[ INFO ] kmo_stats: [tid=000] 4. stdev: | 996.045 | 416.4 | 362.2933 |  
[ INFO ] kmo_stats: [tid=000] 5. mean w. rej.: | 1.187879 | -0.063260 | -0.250587 |  
[ INFO ] kmo_stats: [tid=000] 6. stdev w. rej.: | 1.997403 | 1.518527 | 1.530504 |  
[ INFO ] kmo_stats: [tid=000] 7. median: | 1.416667 | 0.0133333 | -0.186666 |  
[ INFO ] kmo_stats: [tid=000] 8. mode: | 0.7498566 | -0.136713 | -0.317835 |  
[ INFO ] kmo_stats: [tid=000] 9. noise est.: | 1.592248 | 1.447667 | 1.460024 |  
[ INFO ] kmo_stats: [tid=000] 10. min. value: | -385.6167 | -271.98 | -2016.683 |  
[ INFO ] kmo_stats: [tid=000] 11. max. value: | 107723.6 | 95957.1 | 110734.2 |  
[ INFO ] kmo_stats: [tid=000] -----
```

7.3 Make Images

The recipe `kmo_make_image` allows one to combine spectral slices of a cube (collapse the cube) to make an image. It is possible to specify one or more spectral ranges to use; and if an OH spectrum is provided, one can specify that spectral regions close to bright OH lines should be omitted. As an example, the command

```
> esorex kmo_make_image -range='1.52,1.54;1.57,1.59' cube_OBS022_0049_NN.fits
```

	SPARK INstructional Guide for KMOS data	
	Doc No:	VLT-TRE-KMO-146611-009
	Version:	0.5
	Author:	R.Davies, A. Agudo Berbel, E. Wiezorrek
	Date:	31.01.13

will create the file `make_image.fits` from `cube_OBS022_0049_NN.fits`, using data within the spectral ranges 1.52-1.54 μ m and 1.57-1.59 μ m.

7.4 Extract Spectra

The recipe `kmo_extract-spec` allows one to extract a spectrum from each cube in a file. Several methods are available to integrate the pixels: one can provide a spatial mask (which is multiplied into each spectral slice of the cube before spatially integrating the result); one can request that such a mask is generated automatically from the data; or one can specify a circular aperture (pixels whose centres lie within this aperture are used). These 3 options are specified by the `mask_method` parameter as 'mask', 'optimal', or 'integrated' (the default) respectively. An example could be

```
> esorex kmo_extract_spec -mask_method='optimal' -save-mask cube_OBS022_0049_NN.fits
```

In this example, the recipe creates 2 output files: the extracted spectrum, as well as the mask that was derived from the data and used to generate the spectrum.

7.5 Rotate Cubes

While the pipeline can handle shifting and combining data that is not aligned with north, it cannot (yet) deal with data at a variety of offset angles. The recipe `kmo_rotate` can be used to rotate cubes so that they north points in the same direction for all of them, or to rotate cubes so that north points upwards. An example of its usage is:

```
> esorex kmo_rotate -rotations=35 cube_OBS022_0049.fits
```

It is important to note that by default this recipe (and also `kmo_shift`) do not extrapolate. Thus, the spatial extent of the region with finite data values will decrease. If you do not want to this to happen, you can specify the `extrapolate` parameter:

```
> esorex kmo_rotate -rotations=35 -extrapolate cube_OBS022_0049.fits
```

7.6 Copy Cube Sections

To extract a section of a cube, you can use the recipe `kmo_copy`, specifying the starting point and size in each dimension in pixels. To extract a cube covering the same spatial extent, but only a limited wavelength range, one can do:

```
> esorex kmo_copy -z=1500 -zsize=300 cube_OBS022_0049.fits
```

This recipe also has a useful feature that it can strip off any edges that contain just NaN values:

```
> esorex kmo_copy -autocrop cube_OBS022_0049.fits
```

___oooOOOooo___