SPHERE
Spectro-Polarimetric
High-contrast
Exoplanet REsearch

REF: **VLT-TRE-SPH-14690-350/2/0**
Issue: Version 2
Date: 25 Jan 2009 Pages: 74

SPHERE

Data Reduction Pipeline Manual
DRAFT

Disclaimer: please note that this document as well as the software it describes is still in a pre-release state. MANY DETAILS WILL STILL CHANGE ! Document and software is provided for pure information purposes without any gaurantees or support and software can not yet be used as intended and described in the FDR document

Contr	Man-PA	Sci	Syst	INS	DRH	CPI	IRD	IFS	ZIM
					X				

Prepared by: **Name**: Ole Möller-Nilsson **Institute**: MPIA **Date**: 5 - 8 - 2010	*Signature:* Ole Möller-Nilsson
Approved by: **Name**: **Institute**: **Date**: 25 - 1 - 2009	*Signature:*
Released by: **Name**: **Institute**: **Date**: 25 - 1 - 2009	*Signature:*

Contributors: Alexey Pavlov, Joe Carson and Markus Feldt, MPIA

Contents

Chapter 1

Introduction

1.1 Purpose

The SPHERE pipeline is a subsystem of the VLT Data Flow System (DFS). It is used in two operational environments, for the ESO Data Flow Operations (DFO), and for the Paranal Science Operations (PSO), in the quick-look assessment of data, in the generation of master calibration data, in the reduction of scientific exposures, and in the data quality control. Additionally, the SPHERE pipeline recipes are made public to the user community, to allow a more personalised processing of the data from the instrument.

The purpose of this document is to describe a typical SPHERE data reduction sequence with the SPHERE pipeline. This manual is a complete description of the data reduction recipes offered by the SPHERE pipeline, reflecting the status of the SPHERE pipeline as of 1st April 2010 (version 0.4.0).

1.2 Scope of this document

This document describes the data reduction library for the Sphere instrument on VLT. It is part of the deliverables.

The main purpose of this document is to present and explain the data reduction software (in form of a library) for SPHERE in general and IFS, IRDIS and ZIMPOL in particular. The structure and content of this document follows the guidelines set out in the ESO document "Data Flow for VLT/VLTI Instruments Deliverables Specifications" (VLT-SPE-ESO-19000-1618).

The document presented here follows the layout presented in section 4.5.1 of the "Data Flow for VLT/VLTI Instruments Deliverables Specifications" closely with the exception of an added introduction (chapter 1 in this document) and an overview (chapter 2).

The present document describes the design of the data reduction software, including detailed descriptions of algorithms and functions and explains how to reduce SPHERE data with it. Since this is part of an ongoing development process in close contract with manufacture, testing and verification of the SPHERE hardware and instrument design this document describes only a current status.

Great care has been taken to be accurate and precise whenever possible, but it is unavoidable that there are several details regarding the software design and implementation that can only be considered preliminary.

The instrument and detector calibrations as discussed here assume that the hardware requirements for the various sub-systems and specified in the corresponding documents are met and that there are no unforeseen instrument signatures. The SPHERE hardware sub-systems has been designed with unprecedented care and it unlikely that such problems will occur, but some modifications to the data reduction recipes may be required once the optics and detector has been delivered.

1.3 Acknowledgments

We would like to acknowledge all !

1.4 Change Record

Issue	Revision	Paragr	Page	Date	Comment
0	1				Initial Draft
0	2			5-8-2010	IFS AIT Release I
0					
0					
0					
1					
2					

1.5 Applicable Documents

No.	**Document name**	**Document number, Iss./Rev.**
AD1	VLT instrumentation software specifications	VLT-SPE-ESO-17212-0001
AD2	Field and Pupil Rotation for the VLT Units	VLT Report No. 63, ESO 1990
AD3	Data Flow for VLT/VLTI Instruments: Deliverables Specification	VLT-SPE-ESO-19000-1618/2.0
AD4	Common Pipeline Library Technical Developers Manual	VLT-MAN-ESO-19500-3349
AD5	Common Pipeline Library User Manual	VLT-MAN-ESO-19500-2720
AD6	IFS Calibration Plan	VLT-PLA-SPH-14690-0200
AD7	Data Flow for VLT/VLTI Instruments: Deliverables Specification	VLT-SPE-ESO-19000-1618/2.0
AD8	IRDIS Data Reduction Library Design	VLT-TRE-SPH-14690-351
AD9	SPHERE Science Analysis Report	VLT-TRE-SPH-14609-235

1.6 Reference Documents

No.	**Document name**	**Document number, Iss./Rev.**
RD1	Efficient algorithms for robust feature matching.	D. M. Mount, N.S. Netanyahu, J. Le Moigne, Pattern Recognition vol. 32 (1999) pp. 17-38.
RD2	Frame combination techniques for ultra-high-contrast imaging	Carson et al. 2008, SPIE, 7014E, 115C
RD3	IRACproc: a software suite for processing and analyzing Spitzer/IRAC data	Schuster et al. 2006, SPIE, 6270E, 65S
RD4	HST Dither Handbook	Koekemoer et al. 2000, [Baltimore: STScI]
RD5	Frame combination using Drizzle	Fruchter, A.S and Hook, R.N 1997 in Proc. SPIE, Vol. 3164
RD6	Euro3D Format	M. Kissler-Pattig et al., Issue 1.2, May 2003
RD7	IFS Simulation report	VLT-TRE-SPH-14690-0195
RD8	IFS Calibration Plan	VLT-PLA-SPH-14690-0200
RD9	Gasgano User's Manual.	http://www.eso.org/gasgano/ VLT-PRO-ESO-19000-1932. 13, 15, 19, 28

1.7 Acronyms

Acronym	Meaning	Mathematical representation
ANSI-C	The standardized programming language C	
API	Advanced Programming Interface	
CCD	Charge Coupled Device	
CFITSIO	A library for accessing FITS files in C	
CPL	Common Pipeline Library	
CVS	Concurrent Version System	
DBI	Double Imaging Mode	
DC	Dark current	$DC(x,y)$
DF	Detector flat field – the pixel response to an input signal.	$DF(x,y)$
DIT	Detector Integration Time	
DRH	Data Reduction Handling	
DPI	Double Polarization Imaging	
DPR	Data Product	
ESO	European Southern Observatory	
FDR	Final Design Review	
FoV	Field of View	
FPN	Fixed Pattern Noise	$FPN(x,y)$
GSL	Gnu Scientific Library	
HDU	Hierarchical Detector Unit	
HST	Hubble Space Telescope	
HWP	Half-wave plate	
IF	Instrument flat field – the lenslet response to an input signal	$IF(x,y;\Delta x,\Delta y,\lambda)$
IFS	The SPHERE integral field spectrograph instrument	
IFU	Integral Field Unit.	
IRDIS	Sphere imaging instrument	
LRS	Low Resolution Spectroscopy	
LDT	Lenslet Description Table	
MRS	Medium Resolution Spectroscopy	
PAE	Preliminary Acceptance Europe	
PDR	Preliminary Design Review	
PDT	Pixel Description Table	
PRO	Product (FITS keywords)	
PSF	Point Spread Function	
QC	Quality Control	
RON	Read out noise	RON
SVN	“Subversion”– a revision control management system	
TBC	To be confirmed	
TBD	To be decided	
TF	Telescope flat field – the flat field response of the telescope	$TF(x,y;\Delta x,\Delta y,\lambda)$
VLT	Very Large Telescope	

Chapter 2

Overview

In collaboration with instrument consortia, the Data Flow Systems Department (DFS) of the Data Management and Operation Division is implementing data reduction pipelines for the most commonly used VLT/VLTI instrument modes. These data reduction pipelines have the following three main purposes:

- Data quality control: pipelines are used to produce the quantitative information necessary to monitor instrument performance.
- Master calibration product creation: pipelines are used to produce master calibration products (e.g. , combined dark frames, super-flats, wavelength dispersion solutions).
- Science product creation: using pipeline-generated master calibration products, science products are produced for the supported instrument modes.

The accuracy of the science products is limited by the quality of the available master calibration products and by the algorithmic implementation of the pipelines themselves. In particular, adopted automatic reduction strategies may not be suitable or optimal for all scientific goals.

Instrument pipelines consist of a set of data processing modules that can be called from the command line, from the automatic data management tools available on Paranal or from Gasgano. ESO offers two front-end applications for launching pipeline recipes, Gasgano [14] and EsoRex, both included in the pipeline distribution (see Appendix A, page 46). These applications can also be downloaded separately from http://www.eso.org/gasgano and http://www.eso.org/cpl/esorex.html. An illustrated introduction to Gasgano is provided in the "Quick Start" Section of this manual (see page 13).

The SPHERE instrument and the different types of SPHERE raw frames and auxiliary data are described in Sections 3, 6.1, and 7. A brief introduction to the usage of the available reduction recipes using Gasgano or EsoRex is presented in Section 4. In section 5 we advice the user about known data reduction problems. An overview of the data reduction, what are the input data, and the recipes involved in the calibration cascade is provided in Section 8. More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe is given in Section 9. In Section 10 the installation of the SPHERE pipeline recipes is described.

Chapter 3

The SPHERE Instrument: IFS, IRDIS and ZIMPOL

3.1 The IFS Instrument

See [XXX] for a description of IFS.

3.2 The IRDIS Instrument

See[XXX] for a description of IRDIS

3.3 The ZIMPOL Instrument

See [XXX] for a description of ZIMPOL

Chapter 4

Quick Start

This section describes the most immediate usage of the SPHERE pipeline recipes.

4.1 SPHERE pipeline recipes

The currently implemented SPHERE recipes are:

IFS:

sph_ifs_master_dark: creation of master dark frame

sph_ifs_master_detector_flat: creation of master detector flat frame from frames taken with the internal flat calibration lamps

sph_ifs_spectra_positions: assignment of spectral regions in standard dithering position and creation of pixel description table

sph_ifs_instrument_flat: creation of total master flat field (including instrument and detector)

sph_ifs_wave_calib: assigment of wavelengths to pixels using wavelength calibration frames taken with calibration lasers

sph_ifs_ifu_flat: (currently same as sph_ifs_instrument_flat) creation of IFU flat field as wave cube

sph_ifs_science_dr: data reduction recipe including full calibration, extraction of wavelength cube and simple ADI

sph_ifs_ron: measurement of the read out noise

sph_ifs_gain: measurement of the gain

sph_ifs_detector_persistence: measurement of the detector persistence

sph_ifs_cal_background: measurement of the instrument background

sph_ifs_distortion_map: measurement of the instrument background

IRDIS:

sph_ird_master_dark: creation of master dark frame

sph_ird_instrument_flat: creation of master flat field

sph_ird_science_dbi: data reduction recipe including full calibration, extraction of wavelength cube and simple ADI for DBI mode

sph_ird_science_ci: data reduction recipe including full calibration, extraction of wavelength cube and simple ADI for CI mode

4.2 An introduction to Gasgano and EsoRex

Before being able to call pipeline recipes on a set of data, the data must be opportunely classified, and associated with the appropriate calibrations. The Data Classification consists of tasks such as: "What kind of data am I?", e.g. , DARK, "to which group do I belong?", e.g., to a particular Observation Block or template. Data Association is the process of selecting appropriate calibration data for the reduction of a set of raw science frames. Typically, a set of frames can be associated if they share a number of properties, such as instrument and detector configuration. As all the required information is stored in the FITS headers, data association is based on a set of keywords (called "association keywords") and is specific to each type of calibration. The process of data classification and association is known as data organisation. The DO Category is the label assigned to a data type as a result of data classification. An instrument pipeline consists of a set of data processing modules that can be called from different host applications, either from the command line with Esorex, from the automatic data management tools available at Paranal, or from the graphical Gasgano tool. Gasgano is a data management tool that simplifies the data organisation process, offering automatic data classification and making the data association easier (even if automatic association of frames is not yet provided).

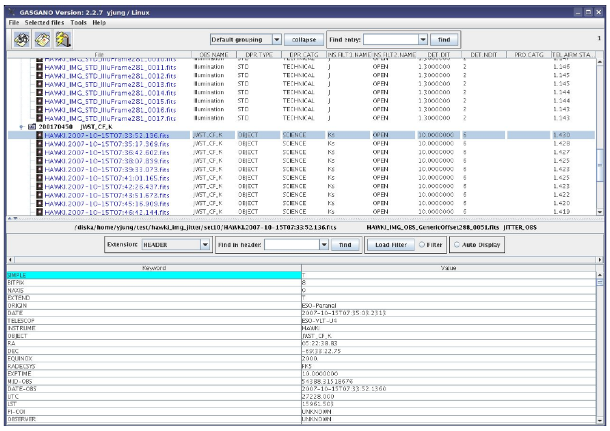
Gasgano determines the classification of a file by applying an instrument specific rule, while users must provide this information to the recipes when they are executed manually using Esorex from the command line. In addition, Gasgano allows the user to execute directly the pipeline recipes on a set of selected files.

4.2.1 Using Gasgano

To get familiar with the SPHERE pipeline recipes and their usage, it is advisable to begin with Gasgano, because it provides a complete graphic interface for data browsing, classification and association, and offers several other utilities such as easy access to recipes documentation and preferred data display tools. Gasgano can be started from the Command Line Interface in the following way: gasgano & Figure 4.2.1 shows the Gasgano main window.

With the pull-down-menu File->Add/Remove Files directories containing SPHERE data can be added. The data are hierarchically organised as preferred by the user. After each file name are shown the classification, the template id, the original filename, the template exposure number and the number of exposures in the template. More information about a single frame can be obtained by clicking on its name: the corresponding FITS file header will be displayed on the bottom panel, where specific keywords can be opportunely filtered and searched. Images and tables may be easily displayed using the viewers specified in the appropriate Preferences fields.

Frames can be selected from the main window with a <CTRL>-left-click for processing by the appropriate recipe: on Figure 4.2 a set of calibration FITS-files have been selected and after selecting the appropriate recipe, the depicted Gasgano recipe execution window will open, having

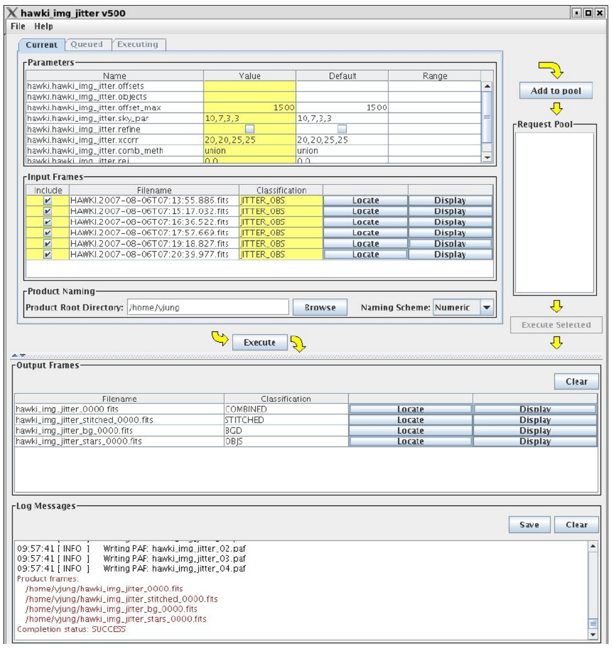


File	OBS.NAME	DPR.TYPE	DPR.CATG	INS.FILT1.NAME	INS.FILT2.NAME	DET.DIT	DET.NDIT	PRO.CATG	TEL.AIRM.STA...
HAWKI_IMG_STD_IlluFrame281_0011.fits	Illumination	STD	TECHNICAL	J	OPEN	1.3000000	2		1.146
HAWKI_IMG_STD_IlluFrame281_0012.fits	Illumination	STD	TECHNICAL	J	OPEN	1.3000000	2		1.145
HAWKI_IMG_STD_IlluFrame281_0013.fits	Illumination	STD	TECHNICAL	J	OPEN	1.3000000	2		1.145
HAWKI_IMG_STD_IlluFrame281_0014.fits	Illumination	STD	TECHNICAL	J	OPEN	1.3000000	2		1.144
HAWKI_IMG_STD_IlluFrame281_0015.fits	Illumination	STD	TECHNICAL	J	OPEN	1.3000000	2		1.144
HAWKI_IMG_STD_IlluFrame281_0016.fits	Illumination	STD	TECHNICAL	J	OPEN	1.3000000	2		1.143
HAWKI_IMG_STD_IlluFrame281_0017.fits	Illumination	STD	TECHNICAL	J	OPEN	1.3000000	2		1.143
HAWKI.2007-10-15T07:33:52.136.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.430
HAWKI.2007-10-15T07:35:17.369.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.428
HAWKI.2007-10-15T07:36:42.602.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.427
HAWKI.2007-10-15T07:38:07.839.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.425
HAWKI.2007-10-15T07:39:33.073.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.423
HAWKI.2007-10-15T07:41:01.165.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.425
HAWKI.2007-10-15T07:42:26.437.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.423
HAWKI.2007-10-15T07:43:51.673.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.422
HAWKI.2007-10-15T07:45:16.909.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.420
HAWKI.2007-10-15T07:46:42.144.fits	JWST_CF_K	OBJECT	SCIENCE	Ks	OPEN	10.0000000	6		1.419

/diska/home/yjung/test/hawki_img_jitter/set10/HAWKI.2007-10-15T07:33:52.136.fits HAWKI_IMG_OBS_GenericOffset288_0051.fits JITTER_OBS

Keyword	Value
SIMPLE	T
BITPIX	8
NAXIS	0
EXTEND	T
ORIGIN	ESO-Paranal
DATE	2007-10-15T07:35:03.2313
TELESCOP	ESO-VLT-U4
INSTRUME	HAWKI
OBJECT	JWST_CF_K
RA	05:22:38.83
DEC	-69:33:22.75
EQUINOX	2000.
RADECSYS	FK5
EXPTIME	10.0000000
MJD-OBS	54388.31518676
DATE-OBS	2007-10-15T07:33:52.1360
UTC	27228.000
LST	15961.503
PI-COI	UNKNOWN
OBSERVER	UNKNOWN

Figure 4.1: The Gasgano main window



Name	Value	Default	Range
hawki.hawki_img_jitter.offsets			
hawki.hawki_img_jitter.objects			
hawki.hawki_img_jitter.offset_max	1500	1500	
hawki.hawki_img_jitter.sky_par	10,7,3,3	10,7,3,3	
hawki.hawki_img_jitter.refine	☐	☐	
hawki.hawki_img_jitter.xcorr	20,20,25,25	20,20,25,25	
hawki.hawki_img_jitter.comb_meth	union	union	
hawki.hawki_img_jitter.rej	0,0	0,0	

Include	Filename	Classification		
✓	HAWKI.2007-08-06T07:13:55.886.fits	JITTER_OBS	Locate	Display
✓	HAWKI.2007-08-06T07:15:17.032.fits	JITTER_OBS	Locate	Display
✓	HAWKI.2007-08-06T07:16:36.522.fits	JITTER_OBS	Locate	Display
✓	HAWKI.2007-08-06T07:17:57.669.fits	JITTER_OBS	Locate	Display
✓	HAWKI.2007-08-06T07:19:18.827.fits	JITTER_OBS	Locate	Display
✓	HAWKI.2007-08-06T07:20:39.977.fits	JITTER_OBS	Locate	Display

Product Root Directory: /home/yjung Naming Scheme: Numeric

Filename	Classification		
hawki_img_jitter_0000.fits	COMBINED	Locate	Display
hawki_img_jitter_stitched_0000.fits	STITCHED	Locate	Display
hawki_img_jitter_bg_0000.fits	BGD	Locate	Display
hawki_img_jitter_stars_0000.fits	OBJS	Locate	Display

```
09:57:41 [ INFO ]    Writing PAF: hawki_img_jitter_02.paf
09:57:41 [ INFO ]    Writing PAF: hawki_img_jitter_03.paf
09:57:41 [ INFO ]    Writing PAF: hawki_img_jitter_04.paf
Product frames:
  /home/yjung/hawki_img_jitter_0000.fits
  /home/yjung/hawki_img_jitter_stitched_0000.fits
  /home/yjung/hawki_img_jitter_bg_0000.fits
  /home/yjung/hawki_img_jitter_stars_0000.fits
Completion status: SUCCESS
```

Figure 4.2: The Gasgano recipe execution window

all the specified files listed in its Input Frames panel. Help about the recipe may be obtained from the Help menu. Before launching the recipe, its parameters may be modified on the Parameters panel (on top). The window contents might be saved for later use by selecting the Save Current Settings entry from the File menu, as shown in figure. At this point the recipe can be launched by pressing the Execute button. Messages from the running recipe will appear on the Log Messages panel at bottom, and in case of successful completion the products will be listed on the Output Frames panel, where they can be easily viewed and located back on the Gasgano main window. Please refer to the Gasgano User's Manual [RD9] for a more complete description of the Gasgano interface.

4.2.2 Using EsoRex

EsoRex is a command line utility for running pipeline recipes. It may be embedded by users into data reduction scripts for the automation of processing tasks. On the other side, EsoRex doesn't offer all the facilities available with Gasgano, and the user must classify and associate the data using the information contained in the FITS header keywords (see Section 6). The user should also take care of defining the input set-of-frames and the appropriate configuration parameters for each recipe run: The set-of-frames: Each pipeline recipe is run on a set of input FITS data files. When using EsoRex the file names must be listed together with their DO category in an ASCII file, the set-of-frames (SOF), that is required when launching a recipe. Here is an example of SOF, valid for the sph_ird_instrument_flat recipe:

```
/data/calib/master_dark.fits                  IRD_MASTER_DARK
/data/2011-03-27/raw_flat_bright_DIT_0.fits   IRD_FLAT_FIELD_RAW
/data/2011-03-27/raw_flat_bright_DIT_1.fits   IRD_FLAT_FIELD_RAW
/data/2011-03-27/raw_flat_bright_DIT_2.fits   IRD_FLAT_FIELD_RAW
```

Note that the SPHERE pipeline recipes do not verify the correctness of the DO category specified by the user in the SOF. The reason of this lack of control is that SPHERE recipes are just one component of the complete pipeline running on Paranal, where the task of data classification and association is carried out by separate applications. Using Gasgano as an interface to the pipeline recipes will however ensure a correct classification of all the data frames, assigning the appropriate DO category to each one of them (see section 4.2.1). A recipe handling an incorrect SOF may stop or display unclear error messages at best. In the worst cases, the recipe would apparently run without any problem, producing results that may look reasonable, but are actually flawed.

EsoRex syntax:

The basic syntax to use ESOREX is the following:

```
esorex [esorex_options] recipe_name [recipe_options] set_of_frames
```

To get more information on how to customise ESOREX (see also [13]) run the command:

```
esorex --help
```

To generate a configuration file esorex.rc in the directory $HOME/.esorex run the command:

```
esorex --create-config
```

A list of all available recipes, each with a one-line description, can be obtained using the command:

```
esorex --recipes
```

All recipe parameters (aliases) and their default values can be displayed by the command

```
esorex --params recipe_name
```

To get a brief description of each parameter meaning execute the command:

```
esorex --help recipe_name
```

To get more details about the given recipe give the command at the shell prompt:

```
esorex --man-page recipe_name
```

4.2.2.1 Recipe configuration:

Each pipeline recipe may be assigned an EsoRex configuration file, containing the default values of the parameters related to that recipe. The configuration files are normally generated in the directory $HOME/.esorex, and have the same name as the recipe to which they are related, with the file name extension .rc. For instance, the recipe sph_ifs_master_dark has its EsoRex generated configuration file named sph_ifs_master_dark.rc, and is generated with the command:

```
esorex --create-config sph_ifs_master_dark
```

The definition of one parameter of a recipe may look like this:

```
# --xcorr
# Cross correlation search and measure sizes.
ifs.master_dark.clean_mean.reject_high=2
```

In this example, the parameter ifs.master_dark.clean_mean.reject_high (controlling the number of outliers at the high end to discard when combining frames) is set to the value 2. In the configuration file generated by EsoRex, one or more comment lines are added containing information about the possible values of the parameter, and an alias that could be used as a command line option. The recipes provided by the SPHERE pipeline are designed to be usable in a cascade of data reduction steps, each controlled by its own parameters. For this reason and to prevent parameter name clashes we specify as parameter prefix not only the instrument name but also the name of the step they refer to. Shorter parameter aliases are made available for use on the command line. The command

```
esorex --create-config recipe_name
```

generates a default configuration file recipe_name.rc in the directory $HOME/.esorex3. A recipe configuration file different from the default one can be specified on the command line:

```
esorex --recipe-config=my_alternative_recipe_config
```

Recipe parameters are provided in Section 9. More than one configuration file may be maintained for the same recipe but, in order to be used, a configuration file not located under $HOME/.esorex, or having a name different from the recipe name, should be explicitly specified when launching a recipe.

4.2.2.2 Recipe execution:

A recipe can be run by specifying its name to EsoRex, together with the name of a set-offrames. For instance, the following command line would be used to run the recipe sph_ifs_master_dark for processing the files specified in the set-of-frames sph_ifs_master_dark.sof:

```
esorex sph_ifs_master_dark sph_ifs_master_dark.sof
```

The recipe parameters can be modified either by editing directly the used configuration file, or by specifying new parameter values on the command line using the command line options defined for this purpose. Such command line options should be inserted after the recipe name and before the SOF name, and they will supersede the system defaults and/or the configuration file settings. For instance, to set the sph_ifs_master_dark reject_high parameter to 4 the following should be typed:

```
esorex sph_ifs_master_dark --ifs.master_dark.clean_mean.reject_high=4
        sph_ifs_master_dark.sof
```

For more information on EsoRex, see [13].

Chapter 5

Mathematical Description

PLEASE NOTE: THIS CHAPTER IS A COPY FROM THE IFS FDR DOCUMENT. IT IS NOT UP TO DATE, HAS LARGE GAPS AND WILL EVENTUALLY BE STRUCTURED BY RECIPE SECTIONS.

Here we describe the mathematical algorithms that are used for data reduction. In addition, this chapter serves as an overview of the general data reduction process.

5.1 Signal propagation through DRH

For a scientific exposure, the most general observation mode for SPHERE, the scientific signal as given by an input flux $S(\alpha, \beta, \lambda)$ results in a detector image, $I(x, y)$ that represents the electrons received by the detector and converted into counts including all instrumental effects. Here the physical (or "sky") coordinates α,β are transformed onto the detector pixels x, y through the dispersive elements, with the mapping

$$S(x, y) = S(x_{\Delta x,\Delta y}(\alpha, \beta, \lambda), y_{\Delta x,\Delta y}(\alpha, \beta, \lambda)),$$

with the pixel to lenslet associations $x_{\Delta x,\Delta y}(\alpha, \beta, \lambda)$ and $y_{\Delta x,\Delta y}(\alpha, \beta, \lambda)$. These pixel to lenslet associations depend on the relative offset between lenslet array and detector, Δx and Δy and are determined during the wavelength calibration procedures. We will henceforth write $S(x, y; \lambda)$ for $S(x_{\Delta x,\Delta y}(\alpha, \beta, \lambda), y_{\Delta x,\Delta y}(\alpha, \beta, \lambda))$, representing the pixelised science image, i.e. a 2-D detector image of the lenslet array that is devoid of instrumental effects. We write the λ dependence here as a reminder that this is a 2D representation of a wavelength cube.

From the entrance into the telescope the scientific signal is affected by several components in an adverse manner, and all these effects have to be removed by the data reduction process in order to achieve maximal scientific output. This is achieved by applying several transformation to the detected image, $I(x, y)$ to reverse the actions of the instrumental and telescope effects. These transformations are in general applied in a sequential manner, reflecting the physical layout of the detecting system, which consists of several components each of which affects the input signal in series. However, it is important to keep this assumption of "sequentially" which underlies most of the principles of astronomical data reduction in mind. In order to allow the removal of the various effects by the instrument/telescope components, one attempts to isolate and measure the effect of each individual component in a calibration procedure which is executed in a separate step to the science observation, either at various times during the observation night, during the preceding day or only at specific times throughout the year.

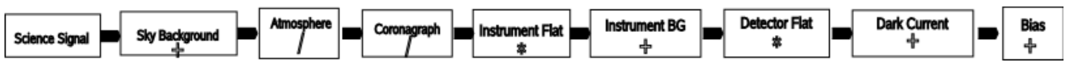


Figure 5.1: Schematic representation of the hardware components for IFS from a DRH point of view. The signal is affected by various hardware components which are calibrated out on the data reduction process. For each component the basic mathematical effect is given either as addition, division or multiplication. The first left most effects are all included in the "science" signal $S(x, y; \lambda)$ below.

The data reduction handling for the IFS subsystem of SPHERE provides calibration procedures to measure and correct for the most important instrumental effects. Realizing that the IFS system can essentially be broken down into three relevant parts: the detector including readout electronics, the instrument, including optical components like lenslets and the telescope, including the SPHERE "common path", the signal propagation can be represented by a series of components that act on the input signal $S(x, y; \lambda)$. We show these components schematically in Fig 3.1. Mathematically the signal propagation can be represented with the following equation:

$$I(x, y) = G \times \{DC(x, y) \times \Delta t + B(x, y) + DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times S(x, y; \lambda)\} + RON, \tag{5.1}$$

where G is the total gain, DC the dark current, B the bias, DF the detector flat response, IF the instrument flat response, RON the readout noise, Δx the detector dither offset in x, Δy the detector offset in y. The exposure time, Δt has the special property (due to the detector technology) that

$$\Delta t = n \times T, \; n \geq 1,$$

where T is a constant exposure time unit, around 1.3sec. Note that $n \geq 1$ and so an exposure time of $\Delta t = 0$ is not possible. This also means that a "bias", defined as the detector response for zero exposure time, can not be measured directly for IFS and IRDIS but has to be inferred.

All the functions for the system components, DC, B, DF, IF and TF are written in detector pixel coordinates, even if the corresponding calibrations may be detector position independent. For example, the instrument flat field is the effect of the lenslet array on the signal, which is a function of lenslet and wavelength but is independent on the detector position. The signal as received in detector pixel coordinates, however, is dependent on the detector offset simply due to a shift in coordinate system. In this sense the functions for the system components defined in the equation above rather represent the detected signal on the detector if all other contributions are zero. The response functions of the detector, the instrument and the telescope are assumed to be linear in the signal S. Linearity of these components, for signals in unsaturated regimes, is part of the SPHERE hardware requirement specification and the linearity assumption is therefore in general justified. An exception are image ghosts (due to optical reflections) and persistence effects.

5.2 Signal propagation reversal

Given the above signal response equation, the inverse can be formulated to infer the original science signal from the detected image:

$$S(x, y; \lambda) = \frac{[I(x, y) - RON]/G - DC(x, y) \times \Delta t - B(x, y)}{DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times TF(x, y, \Delta x, \Delta y, \lambda)}.$$

The statistical mean of the readout noise should be zero (by choice), simplifying the equation slightly to:

$$S(x, y; \lambda) = \frac{I(x, y)/G - DC(x, y) \times \Delta t - B(x, y)}{DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times TF(x, y, \Delta x, \Delta y, \lambda)}. \tag{5.2}$$

Knowledge of the functions DC, B, DF, IF and TF and the gain allows one then to determine the scientific signal from the observed detector image. The various functions are determined in the calibration procedures by isolating the relevant components, using a known input source signal S and processing the detector image.

5.3 Signal propagation for bias and dark calibrations

The signal propagation for bias and dark calibration is very simple since the signal S is zero:

$$I(x, y) = G \times [DC(x, y) \times \Delta t + B(x, y)] + RON.$$

Assuming that the statistical mean of the readout noise is zero, the bias and dark term can simply be obtained as

$$DC(x, y) \times \Delta t + B(x, y) = I(x, y)/G.$$

Thus, taking an exposure with closed shutters and dividing by the gain, directly gives the dark+bias contribution. However, note that this depends on the exposure time. Also, since conversion from electrons to counts in the detector also depends on the readout mode, a bias+dark measurement is required for each exposure time and readout mode used in any observation which is to be processed. This is generally true for all detector effects and will be neglected in the further treatment in this chapter (the only consequence is that every measurement is performed for each possible combination of exposure time and readout mode). In the case that a separate measurement of the components DC and B is required, the following description can be used: repeatedly expose the detector for different times Δt thereby obtaining $I(x, y, \Delta t)$ and perform a linear fit to the observed count, $I(x, y, \Delta t) = k(x, y) \times \Delta t + b(x, y)$. Comparison with the above equation directly yields the dark and bias components. Note that this procedure is necessary because an exposure time of 0s is not possible for the infrared detector and so the bias can not be measured using 0s exposures as for optical CCDs.

5.3.1 A special note about the dark calibration and the use of the word "dark" in this document

Even though the calibration plan foresses a master "dark" calibration, and the calibration as well as the result is referred to as "dark calibration" and "dark" or "master dark" throughout this document, this is not really the correct terminology that should be used for this recipe in the case of IRDIS and IFS. Since the dark current is very low for IR detectors, both IRDIS and IFS, what is actually calibrated in this recipe of the so called Fixed Pattern Noise or FPN which represents the spatial variation of the response of pixels to a zero input stimulus. This is dependent on integration time as well as read out mode and may vary on relatively short timescales. To keep with the terminology of the calibration plan we shall continue to refer to this calibration as the "dark" calibration.

5.4 Signal propagation for the detector flat field

In this case, the detector is illuminated with a uniform lamp of a given wavelength, giving a signal $S(x, y; \lambda) = L(\lambda)$ that is uniform over the detector and depends only on the wavelength, or, more generally, on the spectral energy distribution of the lamp used. Since neither the instrument components (lenslet arrays) or the telescope are involved the detected image is given by:

$$I(x,y) = G \times \{DC(x,y) \times \Delta t + B(x,y) + DF(x,y,\lambda) \times L(\lambda)\} + RON.$$

Knowledge of the bias and dark component from previous measurements, and exploiting the statistical mean of the readout noise of zero gives:

$$DF(x,y,\lambda) = \frac{I(x,y,\lambda)/G - DC(x,y) \times \Delta t - B(x,y)}{L(\lambda)}.$$

This means that the detector flat field response for a given wavelength is measured by taking an exposure of time Δt and subtracting a bias+dark calibration frame with the same exposure time. In general the lamps used for calibration purposes are not perfectly monochromatic, and some detector flats are even taken with a white lamp, giving:

$$DF_L(x,y) = \int [I(x,y,\lambda)/G - DC(x,y) \times \Delta t - B(x,y)]\, L(\lambda)\, d\lambda,$$

where $L(\lambda)$ is the normalized wavelength emission of the calibration lamp L used. Since the actual quantity required in equation 5.2 is $DF(x,y,\lambda)$, it is necessary to extrapolate from a series of $DF_L(x,y)$ for different calibration lamps, L = 1...N. In practice, the calibration lamps used for SPHERE have a small bandwidth and can be assumed to be monochromatic (except for the broad band lamp), giving directly $DF(x,y,\lambda_L)$. Since only a finite number of such calibration lamps are available, it is not possible to determine $DF(x,y,\lambda)$ for every wavelength directly. Rather, determination of $DF_L(x,y)$ for all monochromatic calibration lamps can be used to construct a fit function for every pixel, $f_{x,y}(\lambda)$ which in turn can be used to construct an estimate of $DF(x,y,\lambda)$ for every wavelength. The accuracy of this then depends on the number of monochromatic calibration lamps used, and how well the wavelength dependence of the pixel response can be described by the fitting function chosen.

5.5 Signal propagation for the instrument flat field

For the instrument flat field for the IFS, the set-up is similar to the detector flat field, except that the lenslet array and some related optical components are in the light path. The equation for the signal propagation, eq. 5.1 becomes:

$$I(x,y) = G \times \{DC(x,y) \times \Delta t + B(x,y) + DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times L(\lambda)\},$$

where $L(\lambda)$ is the spectral energy distribution of the calibration lamp and we have assumed that the mean of the readout noise is zero. Now, the calibration measurement is the same as that for the detector flat field, except that we now measure the product $DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda)$ instead of just $DF(x,y,\lambda)$. Since no lenslet array is used in the detector flat exposure, detector flats are independent of the detector position. However, the pixel associated is detector position dependent, and so, in order to measure $IF(x,y,\lambda)$ the pixel positions have to be remapped through the pixel description table before a detector flat is divided out. The main purpose of the sph_ifs_instrument_flat recipe described in section 7 is to create a calibration frame which contains only the $IF(x,y,\lambda)$ part and is detector position independent. These IFU flat calibration frames can simply be obtained in any detector position as long as detector flat fields taken at the same detector position are divided out. Again the limited availability of calibration lamps means that the wavelength dependence will be estimated by functional fits, $f_{x,y}(\lambda)$, to a series of measurements at the different calibration wavelengths, such that

$$DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) = f_{x,y,\Delta x,\Delta y}(\lambda).$$

Recipes that need to correct for the instrument flat field use a set of master input detector flat fields in combination with the detector position independent master IFU flat field to construct the function $f_{x,y,\Delta x,\Delta y}(\lambda)$. Which fitting function to use is decided within the recipe and may be a parameter to the recipe plugin. We should note here, that the λ dependence of the lenslet response is expected to be small – and it may in principle be possible to simplify the data reduction process in this case.

5.6 Finding spectral regions

For the IFU capabilities of the IFS it is necessary to identify all the regions where spectra fall onto the detector in an automatic way. In principle this needs to be done for every possible detector position in a separate calibration step implemented as the sph_ifs_spectra_positions in the data reduction library. However, the creation of detector position dependent PDTs is done purely in the data reduction recipes: only one "master" PDT table for the standard dithering position is created during calibrations. PDTs for other dithering position are calculated from the dithering position and this master PDT. The simplest algorithm for detecting spectral regions proceeds as follows:

1. Create dark subtracted master calibration frames from the input raw frames. Bad pixels must be flagged/set to zero.
2. Divide this frame by a master detector flat field taken with the broad band (white light) lamp.
3. Apply a threshold algorithm to identify regions with pixel values above a certain threshold value.
4. Assign a label for each connected region.
5. Associate these regions from the regions as expected from a model of the lenslet array. Regions that are either associated to two different lenslet IDs or that have no lenslet ID associated from model are counted and marked.
6. Save label information for each pixel in the pixel description table (PDT). Pixels outside spectral regions are given label 0.

The advantage of this procedure is that it is very simple and the spectral regions have been identified using a clear criterion. This procedure is also very robust to "missing" lenslets or gaps. However, this simple minded approach has the disadvantage of requiring a flat spectra response, that is, the signal along a spectrum must be high and the contrast with regions that do not contain spectra must be high. This is not always the case, since the detector is likely to have a strongly wavelength dependent sensitivity the spectra will not be flat on the detector and in some regions the detector sensitivity may be so low that the contrast is not high enough. In addition this procedure does not take account of the fact that the boundary of spectra do not fall exactly in between pixels; some pixels will be illumination partly by a spectrum, further reducing the contrast with un-illuminated regions. All this means that the performance of this procedure depends rather critically on the choice of the threshold parameter.

An alternative approach uses a model function of the spectra locations. This model can be provided simply as an image, $M(x', y')$, where

$$M(x', y') = \begin{cases} 0 & off\ spectra \\ 1 & on\ spectra \end{cases},$$

The determination of spectral regions on the observed detector image, $I(x, y)$then just becomes an optimization problem for finding the offset between $\Delta x = x' - x$ and $\Delta y = y' - y$ such that

$$I(x+\Delta x, y+\Delta y) = M(x', y').$$

This can very easily be done by noting that the correctΔx and Δy maximize the convolution

$$I * M = \int M(x,y) I(x-x', y-y') dxdy.$$

This procedure does assume that the model M used is at least reasonably correct – otherwise determining the maximum of the convolution may not be robust. A further disadvantage of this method is that missing spectra, that are present in the model, may lead to inaccurate associations. An advantage of this method is, however, that the method will perform well even if spectra are not uniform and/or the contrast on the image is low.

Currently only the first method is implemented.

5.7 Signal propagation for wavelength calibrations

For science data reduction purposes of IFU data it is necessary to perform a series of wavelength calibration procedures. Regions on the detector have to be identified where the spectra fall on and every pixel has to be corrected for the wavelength dependent effects. In general, any IFU data at spatial coordinates α, β and at wavelength λ will be constructed from a detector image $I(x,y)$ (obtained with the lenslet array in the optical path) in the following way:

$$IFU(\alpha,\beta,\lambda) = I(x_{\Delta x,\Delta y}(\alpha,\beta,\lambda), y_{\Delta x,\Delta y}(\alpha,\beta,\lambda)),$$

with the pixel to lenslet associations $x_{\Delta x,\Delta y}(\alpha,\beta,\lambda)$ and $y_{\Delta x,\Delta y}(\alpha,\beta,\lambda)$. These pixel to lenslet associations depend on the relative offset between lenslet array and detector and are determined during the spectra positions procedures described in section 3.6. In order to associate the detector pixels with the correct wavelength, a known line spectrum is used to illuminate the detector. Spectral regions are identified and a fit is performed to determine the pixel coordinates of the known line centers of the spectrum. For every lenslet spectrum a table associating the line wavelengths with pixels information is constructed and a fitting/interpolation procedure is used to associate wavelengths for all pixels in between. In this way every pixel will be associated with a lenslet (i.e. α and β coordinates) and a wavelength. Since the procedure makes use of the same instrument set-up as used for the instrument flat procedures the resulting detector image is:

$$I_{\Delta x,\Delta y}(x,y) = G \times \{DC(x,y) \times \Delta t + B(x,y) + DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times S(x,y;\lambda)\}.$$

However, contrary to the instrument flat field procedure we are not interested in obtaining $I(x,y,\lambda)$ but rather, we wish to obtain $S(x,y;\lambda)$, the "true" input signal, i.e. the idealized projection of the spectra after having gone through the lenslet array. Using the reversed propagation equation 5.2, we can write

$$S_{\Delta x,\Delta y}(x,y;\lambda) = \frac{I(x,y)/G - DC(x,y) \times \Delta t - B(x,y)}{DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda)}.$$

Thus, a reconstruction of $S(x,y;\lambda)$ can be achieved if the bias and dark current $DC(x,y)\Delta t + B(x,y)$ as well as the instrument flat $DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda)$ are measured accurately. Since at this stage, the information of the pixel to wavelength associations is not yet available (that is rather the result or purpose of the wavelength calibration) the flat fielding has to be performed here using a IFS flat field frame which has not been divided by the detector flat, but is a measure of both detector and IFU flat.

The flat fielded signal, $S_{\Delta x,\Delta y}(x,y;\lambda)$ is then analysed to detect the spectral lines of the wavelength calibration lamp. The calibration lamp produces very sharp, monochromatic lines. The line profile as observed on the detector are a convolution of the intrinsic line profile, negligible for the calibration lamp lines used, and the instrumental profile (spectrograph resolution). Therefore, the expected line width for these new calibration hardware will be entirely given by the spectrograph resolution (about 2 pixels). Since the calibration lines are sharp, there is a possibility of some additional faint lines due to fringing, and so the positions on the spectra lines have to be determined by pre-selecting the regions close to the expected positions of the lines to avoid the fitting to be performed on some of these fringes (present as local maxima). Alternatively, the data extracted from the spectra region is first passed through a low pass filter to smooth out fringes before fitting is performed – before de-convolving again to assure that the measured FWHM is not affected. The line fringing is not expected to be an important effect for IFS, where spectra resolution is low, but for IRDIS MRS spectroscopy line fringing has to be taken into account. Only the first, simplest method is currently implemented. The fitting itself is performed using a Gaussian fitting procedure.

Once the line positions have been identified, a polynomial of degree $P > 0$ is used to fit the curve of known line wavelengths to measured pixel coordinates. This fit is then used to fill all pixels not covered by lines with wavelength information. If $P > 1$ the second derivative is used to estimate the dispersion. [1]

5.8 Spectral and flux calibrations

The IFS uses IFU capabilities to create a wavelength data cube as the main science product of every observation. This wavelength data cube needs to be, as much as possible, free of instrumental effects and measure as accurately as feasible the true spectrum of the source. However, the observed spectrum is, just like the detector image, affected by several unwanted effects: the atmosphere as well as telescope subsystem introduce wavelength dependent effects. The observed spectrum is given by:

$$F_{obs}(\lambda; z, r, \theta) = F_{real}(\lambda) \times A_{atm}(\lambda; z) \times A_{corono}(\lambda; r, \theta) \times A_{tel}(\lambda; r, \theta), \quad (5.3)$$

where $A_{atm}(\lambda; z)$,$A_{chorono}(\lambda; r, \theta)$ and $A_{tel}(\lambda; r, \theta)$ are the attenuation effects of the atmosphere, the coronagraph and the telescope, respectively and $F_{real}(\lambda)$ is the true scientific signal to detect, which needs to be reconstructed in the calibration procedure. In order to be able to do this, without the need to obtain calibration frames for every science exposure, it is necessary to model the various instrumental and atmospheric effects individually and measure the relative contributions and model parameters at regular intervals. To this end, the various effects can be disentangles making use of the different dependencies: the atmospheric effects depend on airmass, z but are independent of source location on the detector, whereas telescope and coronagraph affect the signal dependent on the source position within the frame but are independent of air mass. The various components are modeled and calibrated as follows.

5.8.1 Atmospheric effects

THIS SECTION DESCRIBES A FEATURE NOT YET IMPLEMENTED !

The atmosphere leads to absorption at certain wavelengths and its effect on the spectrum can be modeled as:

[1]Alternatively, a dispersion model which gives the dispersion as a function of wavelength, $D(\lambda)$ is used to create a "guess" pattern of line positions for each lenslet, and this pattern is matched to the observed line pattern (allowing for positional shifts). The "goodness" of fit of the pattern is calculated for each lenslet and can be used to monitor the dispersion stability of the lenslets. The dispersion $D(\lambda)$ model can also be created as an output of the wavelength calibration when the polynomial fit is used to obtain line positions; the wavelength calibration can in fact be regarded as a measurement of the dispersion for each lenslet.

$$A_{atm}(\lambda; z) = 1 - f(\lambda)exp(-kz),$$

where k is a constant and $f(\lambda)$is the absorption function of the atmosphere. To measure A_{atm} one needs to measure $F_{obs}(\lambda; z, r, \theta)$ at various airmasses, $z_i, i = 0..N$ by observing a blank piece of sky, so that $F_{real}(\lambda) = const$. Then,

$$F_{obs}(\lambda; z, r, \theta) = const \times [1 - f(\lambda)exp(-kz)] \times A_{corono}(\lambda; r, \theta) \times A_{tel}(\lambda; r, \theta),$$

and normalization of measurements by dividing by the measurement at z_0 gives,

$$\frac{F_{obs}(\lambda; z_{i>0}, r, \theta)}{F_{obs}(\lambda; z_0, r, \theta)} = \frac{const \times [1 - f(\lambda)exp(-kz_{i>0})] \times A_{corono}(\lambda; r, \theta) \times A_{tel}(\lambda; r, \theta),}{const \times [1 - f(\lambda)exp(-kz_0)] \times A_{corono}(\lambda; r, \theta) \times A_{tel}(\lambda; r, \theta),},$$

which simplifies to

$$f(\lambda) = \frac{F_{obs}(\lambda; z_{i>0}, r, \theta) - F_{obs}(\lambda; z_0, r, \theta)}{F_{obs}(\lambda; z_{i>0}, r, \theta)exp(-kz_0) - F_{obs}(\lambda; z_0, r, \theta)exp(-kz_{i>0})}.$$

If the constant k is known it is in principle enough to measure the spectrum at two different airmasses to obtain the atmospheric effect. Measurements at more than 2 airmasses allow a determination also of the constant k.

5.8.2 Coronagraph effects

THIS SECTION DESCRIBES A FEATURE NOT YET IMPLEMENTED !

In general, the contribution of the coronagraph is not removed. Frames are processed without division by the coronagraph attenuation to reduce the impact that the incorrect removal of its effect may have on the data quality.

In the rare cases where a removal of the coronagraph effect is explicitly required, exposures can be taken with and without coronagraph and the ratio of the resulting spectra gives directly the coronagraphic effect (except for an unknown normalization constant). However, the fact that measurements are necessary at different points in the field makes this rather problematic if no good model of the coronagraphic effect with few parameters is found. It should be possible to determine a functional model, $A_{corono}(\lambda, r, \theta; \alpha_0, \alpha_1, ..., \alpha_N)$ with N parameters $\alpha_0, ..., \alpha_N$, and N being a small number. The parameters of the model and verification will have to be determined in the first weeks after or during commissioning.

5.8.3 Instrumental background and sky background

THIS SECTION DESCRIBES A FEATURE NOT YET IMPLEMENTED !

In 5.1 no additive effects are included except those arising from the detector. However, there are signal contributions from both the sky and the instrument itself which need to be removed in order to obtain the true science signal. Since these are additive effects, they need to be subtracted out after the reverse propagation equation above, 5.2 has been applied. The total signal is

$$S_{tot}(x, y; \lambda) = S_{sky}(x, y; \lambda) + S_{ins}(x, y; \lambda) + S_{sci}(x, y; \lambda),$$

where S_{sky} is the sky background, S_{ins} is the instrument background and S_{sci} is the actual science signal. Note that these signals are additive – and so the calibration/removal of the sky and instrument background follows a similar procedure to the dark calibration, in the sense that they are subtracted from the input frames. Also note that all contributions have a wavelength dependence.

For IFS the contributions of the sky and instrument background are expected to be small and unimportant for the main science objective: the detection and imaging of planets. It is only in special cases, for example when extended source are observed, that the sky and instrument background may significantly affect the science goal of the observations. Therefore, even though recipes are included in the pipeline to calibrate these effects, their applicability will be limited.

5.8.4 Flux normalization calibration

Since the above calibrations are all performed using ratios, it is not possible to determine in this way the absolute flux. In order to do this, one needs to observe a known source and compare total received flux (i.e. detector counts) with the known flux of the source. This requires a separate calibration procedure: sph_ifs_standard_photometry. For more details, see the description of the recipe.

5.9 Time dependency impact of systems

In the above treatment of the signal propagation for the individual calibrations, we have not discussed the effect of time variation in the various detector, instrument and telescope systems. None of these systems are perfectly stable in time, meaning that it is necessary to repeat calibration procedures at time intervals which are less than the stability time-scale for the required accuracy. For example, the detector flat field is stable to within 0.1% only for about one hour. This means that the recipes that need to correct for effects that involve the $DF(x, y, \lambda)$ term need to use calibration measurements of this quantity that are maximally one hour old. An alternative in such cases it to use monitoring measurements to construct a model of the time behaviour of the relevant subsystems. In the following table, we list approximate dependencies of the calibration terms:

Term	Variable dependence	Time-scale	Variation
$DC(x, y)$	x, y	1 day	1%
$B(x, y)$	x, y	1 day	1%
$DF(x, y, \lambda)$	x, y	30 mins	0.1%
$DF(x, y, \lambda)$	λ	1 week	0.1%
$IF(x, y, \lambda)$	x, y	1 day	0.1%
$IF(x, y, \lambda)$	λ	1 month	0.1%
$TF(x, y, \lambda)$	x, y	1 week	1%
$TF(x, y, \lambda)$	λ	1 month	1%

It is the responsibility of the observer to make sure that the frames used for the calibrations have the required "freshness" – the pipelines will make no checks for that. Also note that some calibration procedures measure quantities that are combinations of terms with different time variability. In these cases, the acceptable time-scale is given by the smallest acceptable time-scale of the subsystems involved. For example, the instrument flat field procedure measures the term $DF(x, y, \lambda) \times IF(x.y, \lambda)$ which has an acceptable time scale for stability of about 30 mins - 1 hour. In some cases, it is also possible to model the time variability in such a way that only part of the procedure has to be performed in frequent intervals. For example, to perform accurate removal of the detector flat field in wavelength calibration it is in principle necessary to have detector flat frames for all 4 calibration lamps that are all newer than 1 hour. However, modeling the behaviour of the detector flat field as

$$DF(x, y, \lambda) = DF(x, y) \times f(\lambda),$$

and noting that the detector response is very stable in terms of the wavelength dependence, $f(\lambda)$ is almost constant over time, it is possible to only perform measurements of $D(x, y)$ frequently, which requires taking calibration data with only a single lamp.

5.10 Clean Mean Algorithm: Basic Frame Combination with outlier rejection

The clean mean algorithm is used to average frames taking into account the possibility of bad pixels and outliers in individual frames. The quality of the detector linearity will therefore be an important contributor to the quality of the data reduction process in SPHERE. We describe in section 8 how we will implement tests for this. The goal is to achieve an optimal mean frame that is not affected by individual bad or outlying pixels at the same time as keeping the maximum amount of information.

We use iterative clean mean with sigma computation (following that described in ESO's SINFONI Pipeline User Manual) as our baseline frame combination method. For this process, the user sets minimum and maximum allowable intensity values. For a stack of frames, values inside this range are then used to determine an intensity mean and standard deviation, for each pixel position. Pixels with values differing from the mean by $k * std$ are removed. This process is re-iterated n times to generate a final mean-combined image.

We wrote an alternative frame combination script with the aim of achieving superior outlier rejection, compared to the clean mean method. This alternative procedure (presented in [RD2]) uses an iterative median/mean combination outlier rejection strategy that takes advantage of the spatial derivative of an image to better deal with variations from a changing PSF shape or small (sub-pixel) pointing errors. This is particularly important in regions where the PSF slope is steepest: there, a small change in pointing or PSF shape could lead to pixel values being wrongly interpreted as outlier pixels. The spatial derivative method should be effective at dealing with such phenomena, as demonstrated in data reduction for Spitzer IRAC (see [RD3]) and HST ([RD4]).

The code operates by first conducting a biased-median-combination of input frames to create a *best estimate image*. "Biased median" refers to taking the value b positions below the median value, to deal with non-symmetrical noise sources like cosmic ray effects. From this *best estimate image*, *BEI,* a *spatial derivative array, SDA*, may be calculated using the following equation.

$$SDA(x,y) = maxabs(BEI(x,y)\check{\ }[BEI(x-1,y), BEI(x+1,y), BEI(x,y+1), BEI(x,y-1)]) \tag{5.4}$$

Going back to the original input frames, we remove any pixel that differs in value from the corresponding *best estimate image* pixel by more than k times the corresponding pixel value in the SDA. In other words, we reject original input frame pixels that meet the following condition.

$$|original - frame(x,y)\check{\ }BEI(x,y)| > kSDA(x,y) \tag{5.5}$$

The now-corrected input images are then mean-combined to generate the final image.

We are in the process of testing these algorithms on simulated images that include effects such as hot pixels, residual cosmic rays, flat-field uncertainties, sub-pixel pointing errors, sub-pixel PSF shape variations, dynamic bad pixels, and other non-ideal phenomena. There are also a number of alternative procedures we may consider for frame combination purposes – this will be decided based on the test results in the development stage.

5.11 Detector pixel linearity

This algorithm is used for example in the sph_ifs_master_detector_flat recipe to determine the detector linearity for each pixel. See section 9.4 for a description of the recipe.

The linearity measurement is used in all recipes where the detector response is assumed to be linear as part of the algorithm. This is the case for all recipes that divide out the detector flat field for example, since the exact detector response is a function of input signal, and extrapolation to the actual input signal from the available detector flat calibration frames is needed (the value $DF(x, y, \lambda)$ in the equation 5.1 is just this linear detector coefficient).

Recipes that need to correct for the detector flat field actually use the detector pixel linearity for the flat fielding, since this gives the correction as a function of detector mean, rather than exposure time. The detector flat to correct for is a function of signal rather than integration time.

As part of this algorithm pixels are identified that do not conform with linearity requirements. Such pixels can also be regarded as "bad pixels" in the sense that their behaviour does not follow the expected behaviour – depending on the required accuracy such pixels may need to be excluded from a frame combination procedure. The identification of bad pixels using this method is possible both for dynamic and static bad pixel identification – but its main use is to identify static bad pixels. The algorithm itself only calculates the reduced χ-square of linear fits to each pixels response and the linear coefficients. Recipes can then use this information to flag certain pixels as bad.

This method to identify bad pixels can also be used to check that the dynamic bad pixel identification works as required and that the dynamic bad pixel identification routine are not affecting further data calibration and reductions adversely.

To determine the detector pixel linearity for monitoring is responsibility part of the sph_ifs_gain recipe, where the response of every pixel to different signal levels is measured and a fit performed. The resulting output is, aside from a map of the gain (the linear fitting coefficient) a map of the reduced χ^2 of the fit as well as a map of pixels that were used in the reduction (dynamic bad pixels having been removed). A correlation analysis between the $\chi^2$and the presence of dynamic bad pixels can then be used to estimate the quality of dynamic bad pixel removal and monitor the performance of the dynamic bad pixel identification algorithm.

5.12 Bad Pixel Identification

Bad pixel identification in SPHERE happens on several levels and in several ways. First, there is a distinction between dynamic and static bad pixels. Static bad pixels are due to a property of the detector and are, as the name implies, unlikely to change with time. It may happen that a pixel changes its status from "good" to "bad" in terms of static bad pixel detection (i.e. the pixel breaks), but the converse should in general not occur. Dynamic bad pixels however vary from exposure to exposure. Identifying these is therefore responsibility for all algorithms that combine a set of raw frames into a smaller set of output frames, e.g. the CLEAN MEAN algorithm described in 5.10.

5.12.1 Static Bad Pixel Identification

The identification of static bad (dead or hot) pixels in the data reduction occurs usually during the master dark calibration recipe. When the master dark frames are created, a static bad pixel map is created in the following way:

1. The master dark calibration frames are created from the input frames. As described in section 7, a master dark is created for every exposure time and readout and for each of these a badpixel map is identified by identifying pixels that are further than a threshold value away from the mean of the frame.
2. A single (empty) static bad pixel map is created
3. For every pixel on the static bad pixel map, all badpixel values of the master dark calibration frames at the same pixel position are read.

4. The static bad pixel is set if and only if all individual master dark calibration frames have a badpixel at the same position

This routine ensures that static bad pixels are truly static and are not random chance events. However, note that a reliable identification of truly static pixels requires that the sigma threshold in identifying the bad pixels in each master calibration frame is chosen adequately and that there is a reasonable (i.e. more than about 3) number of exposure time set-ups in the input frames.

A second procedure to detect static bad pixels uses the detector linearity behaviour to determine bad pixels. During the sph_ifs_detector_flat fielding recipe, a detector linearity map is created (see 5.11). This map gives, for every pixel, a measure of the linearity (goodness of fit for a linear fit) and the linearity coefficient. This map can then be used to flag pixels as bad. Many recipes allow a parameter to control the threshold on the linearity to accept/reject pixels. The detector flat recipe itself creates a static bad pixel map in this way, which is the standard bad pixel map input for other recipes.

The static (or “hot”) bad pixel map identified using the first method, in the sph_ifs_master_dark recipe, is used primarily for monitoring purposes and to validate the static bad pixels identified in the sph_ifs_master_detector_flat field recipe. For that purpose the detector flat field recipe outputs a quality control parameter that measures the number of pixels that have been identified in one static bad pixel map, but not the other. A large number here usually means that the detector linearity performance has degraded.

5.12.2 Dynamic Bad Pixel Identification

Apart from static bad pixels due to faults in the detector, there are also dynamic bad pixels that are created by transient effects: most notably by cosmic rays. These are identified whenever raw frames are combined. Each frame combination routine, like the clean mean algorithm above, needs to reject pixels that are deemed as bad – in the case of the clean mean algorithm due to their value away from the mean value. Since bad pixels due to cosmic rays can affect neighboring pixels as well as the same pixel at a subsequent readout, all pixels around a bad pixel (in a cross shape) should also be flagged as bad as well in the subsequent frame. This happens, for pixels that are sufficiently outlying, automatically in the clean mean algorithm.

5.13 Field Center

THIS SECTION DESCRIBES A FEATURE NOT YET IMPLEMENTED !

For all SPHERE instruments accurate determination of the field center is important. The required accuracy currently is 3mas with a goal of 1mas. This is true in particular for all pupil stabilized (or fixed de-rotator) modes. Here frame combination as described in 3.14 requires de-rotation of raw frames and for this the rotation center needs to be determined accurately. The situation as given from the hardware is as follows:

- the DTTS loop and reference slopes calibration ensures that:
    - this center of rotation is also the photo center and
    - this is also the location of coronagraph (all of this with satisfactory accuracy <0.5 mas)
- the DTTS calibration (CPI-TEC-01) outputs the position on the IRDIS detector of the coronagraph (for its internal use)

Currently there are several different field center calibration strategies considered:

1. Calibration within science data reduction recipes: here one uses the science raw frames themselves and the fact that the coronagraph center can be easily determined by determining the center of the region masked by the coronagraph. The AO (DTTS loop) then ensures that this coronagraph center is also the rotation center. For frames taken without coronagraph the star center itself, which is easily determined by finding the peak and Gaussian fitting.

2. Calibration of the center in a dedicated recipe but using the input raw scieThe RON is computed using dark raw frames with different DITs. A linear fit to the signal as a function DIT is performed and a map of the values at 0s integration time (obtained using the linear fit) is determined. This is repeated for for all input frames and the RMS on the result is the readout noise.nce observation frames. This would be done in analogy with IRDIS star center calibration which uses 4 secondary diffraction peaks to extrapolate the center of the star and hence the rotation center.

3. Dedicated calibration recipe using an artificial source. An artificial illumination source is used to perform a dedicated calibration which determines the center of the coronagraph for all possible instrument set-ups. This would be a daytime calibration with a as yet to be determined frequency. A dedicated recipe would reduce these calibration images and return a reference field center to use for frame combination in science data reduction recipes.

Which of these will be implemented will be decided in the next weeks, using extended analysis.

5.14 Frame combination: de-shifting and de-rotating

Frame combination is one of the most crucial steps in the SPHERE data reduction since accurate frame combination including de-shifting and de-rotation allows use of ADI, SDI and other more advanced planet finding algorithms.

In SPHERE the frame combination including de-rotation, scaling (for SDI) and de-shifting is intended to be an integral part of the pipeline on the other hand and to be flexible and modular on the other. This is realised by allowing the choice of framecombination to be given as an input parameter to recipes, along with all relevant parameters needed for the framecombination algorithm. Currently only a simple ADI is implemented.

For IRDIS the algorithm of choice to de-shift, de-rotate and de-scale is the FFT. The exact FFT implementation is a separate module and de-coupled from the actual framecombination code so it can be replaced with different implementation easily. Currently the FFT provided by GSL is used.

There FFT rotation routine is augmented by a filter to remove high requency noise. This filter is a simple top-hat filter which removes all frequencies that have a k-value in the fourier domain that is above a percantage F of the maximum k-value.

5.15 Creating wavelength cubes

One of the last reduction steps in producing calibrated science frames is to construct a wavelength data cube, $S(\alpha, \beta, \lambda)$. As described before, this is achieved using pixel-to-lenslet association tables. The interpretation of the resulting cubic structure is, however, not straightforward unless some additional geometric transformations are used. The reason is that the lenslet array has a hexagonal rather than rectangular structure. This means that every spatial x, y position in the wavelength cube has a non-trivial associated sky position α,β. How this geometric transformation is performed depends on the level of science reduction. For the basic reduction there is two possible outputs: a result on a hexagonal grid as a FITS table, representing a hexagonal “cube” and a result on a

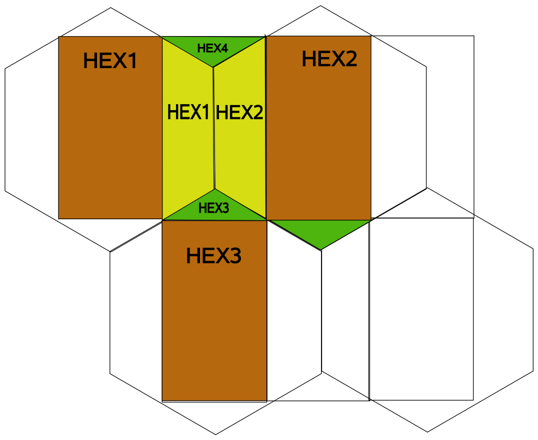


Figure 5.2: Geometry for interpolating the hexagonal lenslet grid onto a rectangular gird. Every second rectangular pixel, R_{odd} is set only from a single hexagon. The other rectangular pixel, $R_{even}$is set using contributions from four hexagons, with larger and equal contributions from two neighboring hexagons (yellow area) and smaller equal contributions for the second pair of hexagons (green regions).

(square) pixel cube. In each case, the way that the interpolation onto the output grid is performed depends on the observing mode: field or pupil-stabilized. We describe the general algorithm for creation of wavelength cube below.

The starting point for cube creation is in any case a series of detector images. Spectra have been identified using the sph_ifs_spectra_locations and sph_ifs_wave_calib routines, which have created a master pixel description table, describing pixel wavelength associations for the standard zero-point dithering position. For each dithering position used in the observations to combine, a new PDT has to be constructed. This is done purely in a software manner – calculating a new PDT for the offset position from the master PDT using the sph_pixel_description_table_new_shift function. These PDTs can then be used to extract a spectrum for every raw input frame – thus providing dither independent information.

Once the dither dependence is removed in this way, the spectra from the raw frames need to be combined. In field stabilized mode, this can simple be done by adding the spectra together. Monochromatic images (in a hexagonal geometry) are then created from the final spectra. In the pupil stabilized mode, this is not possible, since each frame contains an image that is rotated to a different angle. For this case, the raw spectra are used first to create monochromatic images (in a hexagonal geometry) and are then de-rotated before combination.

After monochromatic images in the hexagonal geometry have been created, a cube with a square geometry can be created if Euro3D output is requested. The hexagonal grid is interpolated onto a rectangular grid using the following geometry:

The pixels of the resulting rectangular grid have a width of r_w and a height of r_h, which are related to the length of the side-length s of the hexagons by $r_h = 3s/2$ and $r_w = \sqrt{3}s/2$. The distance between the hexagon centers is also $2s$. The lengths of the sides of the green triangles is r_w and $s/2$ for the long and short sides respectively. The total area of the hexagons is $3/2s^2\sqrt{3}$ and so the weighted interpolation value P_i for the rectangular pixel i is:

$$P_i = \frac{2}{3s^2\sqrt{3}} \begin{cases} 3\sqrt{3}s^2 H_{i/2}/4 & i\ even \\ s^2\sqrt{3}/16(H_{(down)} + H_{(up)}) + \frac{5s^2\sqrt{3}}{16}(H_{left} + H_{right}) & i\ odd \end{cases},$$

where the $H_{up}, H_{down}, H_{left}$, and H_{right} are the values in the Hexagons at the marked location with respect to the square pixel. The wavelength cube will have a plane for every pixel along the wavelength direction of the spectra. In case that not all spectra have the same length on the detector, the number of wavelength planes in the cube is given by the longest spectrum.

5.16 Astrometry and plate scale solution

THIS SECTION DESCRIBES A FEATURE NOT YET IMPLEMENTED !

Chapter 6

Instrument Data Description

In this section we describe the raw data, including the for DRH relevant DPR keywords for each data type. All valid combinations of FITS DPR keywords are identified, listed and identified with corresponding recipes which need to use them as input.

For each of these data structures the basic data type is a FITS file with an image in the HDU corresponding to the full 2048x2048 pixel region of the detector and no extensions. The keywords in the header of the FITS file depend strongly on the data structure represented. The following table lists the keywords for each of the data structures for technical, science and monitoring calibrations.

6.1 General Data Layout

A raw SPHERE file always has the images stored in the primary FITS data unit. The table on the next pages lists the raw data types, the corresponding calibrations (names as in the calibration plan)

Data Type	Source	Description	Calibration Recipe	Identifying FITS Keywords	Other FITS Keywords[1]
DARK_RAW	IFS- TEC- 01		sph_ifs_master_dark	DPR CATG = CALIB DPR TYPE = DARK DPR TECH = IMAGE	INS.SHUT.ST = F (closed)
DFF_RAW	IFS- TEC- 02 or IFS- TEC- 03	Detector flat frames	sph_ifs_ master_ detector_ flat	DPR CATG = CALIB DPR TYPE = FLAT, LAMP DPR TECH = IMAGE	INS.LAMP1.ST = T/F INS.LAMP2.ST = F/T INS.LAMP3.ST = T/F INS.LAMP4.ST = F/T
IFS_FLAT_ RAW	IFS- TEC -04	Instrument flat frames	sph_ifs_instrument_flat	DPR CATG = CALIB DPR TYPE = FLAT, LAMP DPR TECH = IFU	INS.LAMP1.ST = T/F INS.LAMP2.ST = F/T INS.LAMP3.ST = T/F INS.LAMP4.ST = F/T INS.DITHER.X <standard dither pos> INS.DITHER.Y <standard dither pos>
	IFS- TEC -04		sph_ifs_spectra _positions	DPR CATG = CALIB DPR TYPE = FLAT, LAMP DPR TECH = IFU	INS.LAMP1.ST = T/F INS.LAMP2.ST = F/T INS.LAMP3.ST = T/F INS.LAMP4.ST = F/T INS.DITHER.X <standard dither pos> INS.DITHER.Y <standard dither pos>
WAVELENGTH_ CALIB_RAW	IFS- SCI- 10	Wavelength calibration frames	sph_ifs_wave_ calib	DPR CATG = CALIB DPR TYPE = LAMP, WAVE DPR TECH = IFU	CPI.LAMP3.ST = T/F CPI.LAMP4.ST = F/T INS.DITHER.X <standard dither pos> INS.DITHER.Y <standard dither pos>

[1] All recipes require special settings for the EXPTIME, DIT and DET READNO Keywords. See Appendix B.

SCIENCE_ RAW	OBS	Science observation	sph_ifs_science_dr	DPR CATG = SCIENCE DPR TYPE = OBJECT DPR TECH = IFU	ESO TEL AIRM START <airmass> ESO TEL AIRM END <airmass> INS.OPTI10.NAME <coron setting>
DARK_RAW	IRDIS- TEC-01	Raw dark exposures	sph_ird_master_dark	DPR CATG = CALIB DPR TYPE = DARK DPR TECH = IMAGE	
IFF_RAW	IRDIS- TEC-02	Instrument flat frames	sph_ird_instrument_flat	DPR CATG = CALIB DPR TYPE = FLAT, LAMP DPR TECH = IMAGE	INS.LAMP5.ST = On INS.FILTER.NAME = <common filter> INS.OPTI2.NAME = <dual filter>
SCIENCE_CI_ RAW	OBS	Science observation	sph_ird_science_imaging	DPR CATG = SCIENCE DPR TYPE = OBS DPR TECH = IMAGE	INS.FILTER.NAME = <common filter> INS.OPTI2.NAME = Empty INS.OPTI10.NAME <coron setting>
SCIENCE_DBI_ RAW	OBS	Science observation	sph_ird_science_dbi	DPR CATG = SCIENCE DPR TYPE = OBS DPR TECH = IMAGE	INS.FILTER.NAME = <common filter> INS.OPTI2.NAME = <dual filter> INS.OPTI10.NAME <coron setting>

6.2 Imaging Frames

The raw imaging frames for IFS and IRDIS all contain in total 2048x2048 pixels. For IRDIS all raw frames that are obtained with calibrations that illuminate the detector through the IRDIS optical path, only an area of 2048x1024 pixels is used. This again is split in two parts for classical imaging and DBI modes.

Chapter 7

Static Calibration Data

7.1 Valid DIT Lists

THIS SECTION DESCRIBES A FEATURE IMPLEMENTED ONLY IN PART !

As described throughout this document, in normal circumstances (e.g. all standard science observations and all standard calibration observations) the list of valid DIT is restricted to certain values to allow only a limited and manageable list of combinations between number of readouts and integration time. A preliminary list of valid combinations together with the recipe requirements is given in the table below. The final list will be compiled before PAE.

DIT ID	Exposure time/ readout	number Readouts	Total Exp. time
1	1.3 sec	1	1.3 sec
2	1.3 sec	2	2.6 sec
3	1.3 sec	3	3.9 sec
4	1.3 sec	4	5.2 sec
5	2.6 sec	1	2.6 sec
6	2.6 sec	2	5.2 sec
7	2.6 sec	3	7.8 sec
8	5.2 sec	1	5.2 sec
9	5.2 sec	2	10.4 sec
10	10.4 sec	1	10.4 sec
11	10.4 sec	2	20.8 sec
12	10.4 sec	10	104.0 sec
13	10.4 sec	20	208.0 sec

7.2 IFS lenslet model

Several recipes, in particular the wavelength calibration, spectra positions and IFU flat recipes require a model of the lenslet. This model describes how the lenslets are projected onto the detector in some standard dithering position (the “zero” position). In future version this may be extended to include other relevant IFS instrument model parameters, like filter parameters, etc. In the current version, the lenslet model can be provdied most easily as a simple ASCII text file which is written in the “ini” style format of “KEY = VALUE” pairs on each line. The following is a template that can be used to create custom lenslet model files (remove comments in file):

```
[LENSLET MODEL]
DETECTOR PIXEL SIZE = 2048                  # Size of the detector in pixels
LENSLETS ON SIDE = 145                      # Number of lenslets on each side of hexagon
PIXEL SIZE = 18.000000                      # Size of pixels in microns
LENSLETS SIZE = 161.500000                  # Size of lenslets (side of hexagon) in microns
SPECTRA PIXEL LENGTH = 38.100000            # length of spectra in pixels
SPECTRA PIXEL WIDTH = 4.930000              # width of spectra in pixels
SHEAR IN Y = 1.155000                       # Shear factor (this shears the position)
ZERO OFFSET X = 2.000000                    # Offset in zero dither position
ZERO OFFSET Y = 8.000000                    # Offset in zero dither position
SHRINK X = 0.700000                         # Shrinkage factor in X (affects spec positions only)
SHRINK Y = 1.000000                         # Shrinkage factor in Y (affects spec positions only)
MAX LAMBDA = 1.350000                       # Maximum of wavelength range (microns)
MIN LAMBDA = 0.950000                       # Minimum of wavelength range (microns)
```

Please note that the ASCII format of the irdis model is provided for ease of use during AIT but will not be supported after PAE due to ESO requirements. Also note that some parameters controlled in the instrument model file will in future be determined by keywords in the raw frames.

7.3 IRDIS Instrument Model

As for IFS some recipes for IRDIS require a model of the instrument. This model is more important for IRDIS since it defines the regions that are illuminated in the standard dithering position and is used by almost all recipes (exception are dark type recipes where the shutter is closed, or recipes where the IRDIS path is not used).

Similarly to IFS the IRDIS instrument model is currently passed as an input "frame" as an ASCII type file in a "ini" format. Below is the standard model that can be used as a template for own models:

```
[ IRDIS MODEL ]
DETECTOR PIXEL SIZE = 2048                        # Size of the detector in pixels
PIXEL SIZE = 18.000000                            # Size of pixels in microns
ZERO OFFSET X = 2.000000                          # Offset in zero dither position
ZERO OFFSET Y = 8.000000                          # Offset in zero dither position
WINDOW SIZE Y = 1024                              # Size of the detector read out window
ZERO WINDOW START Y = 512                         # The y position of the window
SPLIT PIXEL X = 1024                              # The pixel in x that divides the two fields
```

Please note that as for IFS the ASCII format of the irdis model is provided for ease of use during AIT but will not be supported after PAE due to ESO requirements. Also note that some parameters controlled in the instrument model file will in future be determined by keywords in the raw frames.

Chapter 8

Data Reduction

8.1 Data Reduction Overview

8.2 Calibration Products Data Representation

For both IRDIS and IFS, the data reduction pipeline creates calibration products in a variety of formats. The two most general formats are described in this section, but see the description of the individual recipes for more detailed information and information on other data product formats.

8.2.1 The SPHERE "master frame"

The most simple data product produced by the SPHERE pipeline consists of a FITS file with 4 extensions, all containing a single plane (NAXIS = 2) and all having the same number of pixels in both x and y (for IFS and IRDIS this will be NAXIS1 = NAXIS2 = 2048 in most cases). Ths extensions have the following meaning:

Extension Number	Type	BITPIX	Meaning
1	FLOAT	-32	Image / Main values
2	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad)
3	FLOAT	-32	Weightmap (e.g. number of pixels that went into result)
4	FLOAT	-32	RMS Error / Other Error Info

8.2.2 The SPHERE "cube"

THIS SECTION DESCRIBES AN ADVANCED FEATURE.

In addition to the master frame described above, a commonly used data format is the "cube". This is not a wavelength cube or similar object, but rather a collection of images that are organised in one file. These "cubes" contain calibration products. In general recipe produce data in this format only when a more "user opaque" mode of execution is requested by setting the internal recipe logic switch to "on" (e.g. this is the case when the pipeline runs fully automatically within the consortium). Since this data is difficult to intpperpret simply by "looking at it" it is imperative to refer to the detailed output format information for the corresponding recipes in chapter 9 of this document.

Data written in a SPHERE cube is contained in a single FITS file with 4 extensions, but any number of N $>$ 0 planes. It can be thought of as a stacked SPHERE master frame. In addition to the four image extensions (which have the same meaning as for the SPHERE master frame descriped above) there is also a table extension. This table describes in each row the particular instrument settings of one plane. Currently, the table only has 2 columns, one with a number corresponding to the plane (for 1 to N) and the second wtih a number giving a "z-value" which is interpreted in all present cases as being a DIT ID, corresponding to the ID as given in the DIT table described in section 7.1. In future versions the SPHERE cube will have as a last extension a more extensive table that has a number of columns, each corresponding to one FITS keyword that represents a particular property of that plane. For eample, a master dark represented in a SPHERE cube would consist of 4 image extensions (dark current, badpixels, weightmap and rms error) with each N planes. The table would have N rows, with the first column giving the plane number, the second column giving the value of the exposure time keyword, and the third column giving the value of the read out mode keyword. The columns will be headed by the corresponding keyword name.

Chapter 9

IRDIS Pipeline Recipe Interfaces

9.1 sph_ird_master_dark

Purpose:

to create the master dark frame.

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
DarkFrameList	IRD_ DARK_RAW	Instrument	No	The list of raw dark frames, obtained with different number of readouts
ValidDitListFrame	IRD_MASTER_ DARK_SETTINGS	special	Yes	A FITS table with the valid DIT list to use. If this frame is not specified the default list is used.

parameter list:

Name	Data Type	Default Value	Description
ird.master_dark.coll_alg	INT	0 : CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN (value=0) or MEAN (value = 1) possible
ird.master_dark. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
irdmaster_dark. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ird.master_dark. outfilename	STRING	"master_dark.fits"	The output file the resulting master dark will be written to. Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.
ird.master_dark. make_hotpix	BOOL	True	Specifies if hot pixel output is requested (1 = yes)
ird.master_dark. hotpixfilename	STRING	"hotpixel_mask.fits"	The output file the resulting hot pixel mask will be written to. If not set (the default) no hot pixel mask is written.Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.

Description/Algorithm:

Using the input exposures, which have to conform to the Raw Dark Frame format described in section 6, a SPHERE cube structure is created with the output average dark frames. Each plane in the resulting data cube corresponds to a combined frame for a specific DIT ID, as present in the list of valid DITS (see section 7).

As a first step, the input list of raw frames is sorted into groups with the same DIT setup (i.e. identical DIT ID) and for each group a combined master frame is created using the algorithm specified by the CollapseAlgorithm parameter. For the collapse algorithm a normal mean (set the ifs.master_dark.coll_alg to 1) or a clean mean is possible (ifs.master_dark.coll_alg = 0). The clean mean algorithm sorts all the pixels of all frames with the same position according to value and removes the lowest reject_low and highest reject_high values. For example, if at a pixel 1234, 321 the dark in 10 frames has values 12, -6, -1, -3, 10, 232, 2, 5, -21, 1 and reject_low is set to 2 and reject_high set to 3, the values -21, -6, 232, 12, 10 would be removed before combining with a normal mean.

After combination static bad pixels are flagged as those pixels that lie 5 sigma away from the mean of all the pixels of the result.

The collapsed frame is stored in the cube structure, along with statistical information maps: number of combined input frames for each pixel, RMS map and bad pixel map. If the Hotpixel output is set, a map of the static bad pixels is written out, see section 5.12.1.

Recipe Logic:

The master dark recipe is implemented currently only for the logic switch "on" mode: master dark frames for all allowed detector settings present in the complete set of input frames are stored in a single cube.

Output:

Name	Multiplicity	Data Type	Further used by	Description
MasterDark	1	IRD_MASTER_DARK	almost all recipes	The output master dark in a cube with every plane corresponding to a combined dark frame of a given valid DIT setup, identified by the DIT ID.
Hot PixelMask	1 (optional)	IRD_STATIC_BADPIX-ELMAP	instrument monitoring	The static bad pixels flagged as bad in all input frames by the recipe are written into a separate FITS file for instrument monitoring. All flagged bad pixels are also present in the MasterDark output in the bad pixel extension.

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_DARK_MEAN	double[1..N]	0.0	-	The mean value of the resulting dark frames: one value per plane in cube
QC_DARK_STDEV	double[1..N]	0.0	-	The standard deviation value of the resulting dark frames: one value per plane in cube

9.2 sph_ird_instrument_flat

Purpose:

to create the master flat frame

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
FlatFrameList	IRD_ FLAT_ FIELD_RAW	Instrument	No	The list of raw detector flat field frames, obtained with different number of readouts
MasterDarkFrame	IRD_MASTER_DARK	sph_ird_ master_dark	No	The master dark frame
Hot Pixel Mask	IRD_STATIC_ BADPIXEL_MAP	sph_ird_ master_dark	Yes	The hot pixel mask identified in the dark calibration

parameter list:

Name	Data Type	Default Value	Description
ird.instrument_flat. coll_alg	INT	0: CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN possible
ird.instrument:_flat. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ird_instrument_flat. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ird.instrument_flat. outfilename	STRING	"master_flat.fits"	The output file the resulting master frame will be written to. In case more than one output is written, the name without the .fits extension is used as a base to construct the filenames from.Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.
ird.instrument_flat. make_fittab	BOOL	False	Whether pixel fit table should be written or not
ird.instrument_flat. fittab_outname	STRING	"fittab"	The output file base (without the *.fits extension) of the fit parameters to the pixels. Only used if ifs.master_ detector_flat. make_badpix is yes.Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.
ird.instrument_flat. badpix_chisqtolerance	DOUBLE	50.0	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.master_ detector_flat. make_badpix is yes.
ird.instrument_flat. badpix_uptolerance	DOUBLE	50.0	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.master_ detector_flat. make_badpix is yes.
ird.instrument_flat. badpix_lowtolerance	DOUBLE	50.0	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.master_ detector_flat. make_badpix is yes.
ird.instrument_flat. mean_tolerance	DOUBLE	1.0	The mean count level resolution. Frames within this mean count are deemed to have the same illumination level and will be combined before the fit is performed.

Description/Algorithm:

The recipe to create the flat fields for IRDIS uses input exposures with different illumination levels, taken with various sets of filters. the broad lamp. The set of these filter and DIT dependent master calibration frames is used in all subsequent recipes that need to remove the pixel to pixel variation in signal response of the detector. The flat field recipe creates master calibration frames, using the input exposures, which have to conform to the FLAT_RAW format described in section 6. The master calibration frames are created as multi-plane, multi-extension FITS files for each filter.

Each plane in the resulting data cubes corresponds to a combined frame for a specific DIT ID (similar to the procedure used for the sph_ird_master_dark recipe).

As a first step the list of input frames is sorted into groups of frames with the same DIT setup (i.e. identical DIT ID). Frames are sorted within each of these sets according to the mean count in these frames. Pixels marked as bad in the static badpixel mask from the master dark are not included in the calculation of the mean. All frames that have a mean count that is within ird.instrument_flat.mean_tolerance are combined using the algorithm specified by the CollapseAlgorithm parameter. A linear fit is then performed to obtain the flat field value for each pixel. The algorithm for this is described in more detail in section 5.11. During this the bad pixel map extension and the RMSMAP extension of the master calibration detector flat are set. Finally the results are written out as FITS files conforming to the standards laid out in section 7. If the hot pixel mask from the dark calibration is provided, the bad pixels identified in the detector flat is compared with the input hot pixel map.

Recipe Logic:

The master flat recipe is implemented currently only for the logic switch "on" mode: master dark frames for all allowed detector settings present in the complete set of input frames are stored in a single cube. Use this recipe with frames with identical instrument setu-ups only to obtain the same beahviour as for the logic "off" mode.

Output:

Name	Multiplicity	Data Type	Further used by	Description
MasterFlat	1	IRD_MASTER_ FLAT_FIELD	all science calibration	The output master detector flat in a cube with every plane corresponding to a combined dark frame of a given valid DIT setup, identified by the DIT ID. See section 5.3 and 5.4 for more info.
Pixel Linearity Map	1	PIXEL_ LINEARITY_ MAP	some science calibrations.	The polynomial fit parameters for all pixels listed in a table along with the reduced chi squared value of the fit.

QC1 Parameters:

QC Parameters have not been implemented in this version. This section describes foreseen requirementes.

Name	Data Type	Exit value on error	Further used by	Description
QC_DFF_ MEAN	double[1..N_{DIT},1...N_λ]	0.0	-	The mean value of the resulting detector flat frames: one value per plane in cube and per calibration wavelength
QC_DFF_ STDEV	double[1..N_{DIT},1..N_λ]	0.0	-	The standard deviation value of the resulting detector flat frames: one value per plane in cube and per calibration wavelength
QC_DFF_ UNMATCH-ING_ BADPIX_ COUNT	int	-1	-	The difference between the number of bad pixels identified either only in the linearity method or only in the hot pixel map. Large numbers indicate a possible problem with the pixel linearity. The value is only written if the hot pixel mask is given as input.

9.3 sph_ird_science_imaging

Purpose:

to reduce the science observations from IRDIS in classical maging mode.

type:

science calibration

input data:

Name	Data Type	Source	Optional	Description
RawFrameList	IRD_ SCIENCE_IMAGING_RAW	Instrument	No	The list of raw detector flat field frames, obtained with different number of readouts
MasterDarkFrame	IRD_MASTER_DARK	sph_ird_ master_dark	No	The master dark frame
MasteFlatFrame	IRD_MASTER_FLAT	sph_ird_ instrument_flat	No	The master flat frame

parameter list:

Name	Data Type	Default Value	Description
ird.science_imaging. coll_alg	INT	0: CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN possible
ird.science_imaging. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ird.science_imaging. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ird.science_imaging. use_adi	BOOL	TRUE	Swwitch to set if ADI is supposed to be used. ADI will only work correctly on frames taken with pupil stabilised mode (an output is produced in any case though).
ird.science_imaging. filter_radius	DOUBLE	0	A value to control a top hat filter used when de-rotating frames. See below for more details.

Description/Algorithm:

This is the main science data reduction recipe for IRDIS in classical imaging mode. The input frames should be a set of raw science frames taken in either field or pupil stabilised mode, along with the corresponding set of calibration data: a master dark and a master flat frame. The dark is subtracted from all frames before each frame is divided by the flat frame provided. The two part of the IRDIS detector image are then extracted into a left and right part. The following framecombination steps are then executed for each part independently:

1. Reading the dither position keywords ESO SPH PC DITHER X and ESO SPH PC DITHER Y from each frame and shift each frame by a corresponding dithering amount to align all frames to the same dithering position.
2. If ADI mode is requested by the user, the frames are now combined to create a speckle image
3. Reading the ESO SPH PC POSANG keyword to determine the rotation angle of the frame in question. The frame is de-rotated using a FFT approach. In case that the filter_radius parameter is larger than 0, a radial top hat filter is applied in the fourier domain during the rotation step. This masks out all regions in k-space where $k_x^2 + k_y^2 \geq filter_{radius}^2 \times k_{max}^2$, where k_{max}^2 is the maximul k value for the frame. In other words, all frequencies above a fraction of filter_radius of the maximum frequency present are removed.
4. If ADI is requested, the speckle pattern is subtracted from each de-rotated frame.
5. The resulting frames are combined using the requested framecombination algorithm with corresponding parameters. In order to remove unwanted negative signals, a clean_mean should be chosen and a reject_high should be set to 1 or higher.

When frames have been combined in this way and a resulting left and right frame has been created they are written out as FITS files with the format described in 8.2.1.

Output:

Name	Multiplicity	Data Type	Further used by	Description
Reduced Science	2	IRD_SCIENCE_IMAGING	none	The output science frame. Two frames are produced: one for the left and one for the right field.

QC1 Parameters:

QC Parameters have not been implemented in this version. This section describes foreseen requirementes.

Name	Data Type	Exit value on error	Further used by	Description
QC_DFF_MEAN	double[1..N_{DIT},1...N_{λ}]	0.0	-	The mean value of the resulting detector flat frames.
QC_DFF_STDEV	double[1..N_{DIT},1..N_{λ}]	0.0	-	The standard deviation value of the resulting detector frames.

Chapter 10

IFS Pipeline Recipe Interfaces

10.1 sph_ifs_master_dark

Purpose:

to create the master dark frame.

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
DarkFrameList	IFS_ DARK_RAW	Instrument	No	The list of raw dark frames, obtained with different number of readouts

parameter list:

Name	Data Type	Default Value	Description
ifs.master_dark. coll_alg	INT	2 : CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN possible
ifs.master_dark. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.master_dark. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.master_dark. sigma_clip	DOUBLE	5.0	The sigma value to use for sigma clipping to determine the static badpixels.
ifs.master_dark. outfilename	STRING	“master_dark.fits”	The output file the resulting master dark will be written to.Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.
ifs.master_dark. make_badpix	BOOL	True	Specifies if hot pixel output is requested (1 = yes)
ifs.master_dark. badpixfilename	STRING	“static_badpixels.fits”	The output file the resulting hot pixel mask will be written to. If not set (the default) no hot pixel mask is written.Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.

Description/Algorithm:

Using the input exposures, which have to conform to the Raw Dark Frame format described in section 6, a SPHERE cube structure is created with the output average dark frames. Each plane in the resulting data cube corresponds to a combined frame for a specific DIT ID, as present in the list of valid DITS (see section 7).

As a first step, a combined master frame is created using the algorithm specified by the CollapseAlgorithm parameter. For the collapse algorithm a normal mean (set the ifs.master_dark.coll_alg to 1) or a clean mean is possible (ifs.master_dark.coll_alg = 0). The clean mean algorithm sorts all the pixels of all frames with the same position according to value and removes the lowest reject_low and highest reject_high values. For example, if at a pixel 1234, 321 the dark in 10 frames has values 12, -6, -1, -3, 10, 232, 2, 5, -21, 1 and reject_low is set to 2 and reject_high set to 3, the values -21, -6, 232, 12, 10 would be removed before combining with a normal mean.

After combination static bad pixels are flagged as those pixels that lie 5 sigma away from the mean of all the pixels of the result.

The collapsed frame is stored in the cube structure, along with statistical information maps: number of combined input frames for each pixel, RMS map and bad pixel map. If the Hotpixel output is set, a map of the static bad pixels is written out, see section 5.12.1.

Recipe Logic:

The master dark recipe is implemented currently only for the logic switch "on" mode: master dark frames for all allowed detector settings present in the complete set of input frames are stored in a single cube.

Output:

Name	Multiplicity	Data Type	Further used by	Description
MasterDark	1	MASTER_ DARK	almost all recipes	The output master dark in a master_frame corresponding to a combined dark frame of a given valid DIT setup, identified by the DIT ID.
Hot PixelMask	1 (optional)	HOTPIXEL_ MASK	instrument monitoring	The static bad pixels flagged as bad in all input frames by the recipe are written into a separate FITS file for instrument monitoring. All flagged bad pixels are also present in the MasterDark output in the bad pixel extension.

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_DARK_MEAN	double[1..N]	0.0	-	The mean value of the resulting dark frames: one value per plane in cube
QC_DARK_STDEV	double[1..N]	0.0	-	The standard deviation value of the resulting dark frames: one value per plane in cube

10.2 sph_ifs_master_detector_flat

Purpose:

to create the master detector flat frame

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
DetectorFlatFrameList	IFS_DETECTOR_ FLAT_FIELD_RAW	Instrument	No	The list of raw detector flat field frames, obtained with different number of readouts
MasterDarkFrame	IFS_MASTER_DARK	sph_ifs_ master_dark	No	The master dark frame
Hot Pixel Mask	IFS_STATIC_ BADPIXEL_MAP	sph_ifs_ master_dark	Yes	The hot pixel mask identified in the dark calibration

parameter list:

Name	Data Type	Default Value	Description
ifs.master_detector_flat. coll_alg	INT	0: CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN possible
ifs.master_detector_flat. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.master_detector_flat. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.master_ dectector_flat. outfilename	STRING	"master_dff.fits"	The output file the resulting master frame will be written to. In case more than one output is written, the name without the .fits extension is used as a base to construct the filenames from. Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.
ifs.master_ detector_flat. badpixfilename	STRING	"badpixel_map.fits"	Name of bad pixel map to be written or not, only used if make_badpix set to yes. Note that this parameter is ignored when Esorex / Gasgano is used to execute the recipe. In that case setting the filename is done by Esorex/ Gasgano.
ifs.master_ detector_flat. make_badpix	BOOL	False	Whether bad pixel map should be written or not
ifs.master_ detector_flat. badpix_chisqtolerance	DOUBLE	50.0	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.master_ detector_flat. make_badpix is yes.
ifs.master_ detector_flat. badpix_uptolerance	DOUBLE	10.0	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.master_ detector_flat. make_badpix is yes.
ifs.master_ detector_flat. badpix_lowtolerance	DOUBLE	0.10	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.master_ detector_flat. make_badpix is yes.

Description/Algorithm:

Calibration using detector flat fields is foreseen to occur in two modes in the SPHERE calibration plan. The first calibration procedure uses detector flat frames obtained using the full set of calibration lamps with different wavelengths. The second, faster, calibration procedure only uses detector calibration frames obtained with a broad band lamp. To correct any exposure for the pixel response variations (i.e. perform "flat fielding") these broad band detector flat frames are used to normalize the wavelength dependent response function that has been measured in the first detector flat calibration procedure. In this way, the time-variability of the the detector flat can be taken into account (see also section 5.9). The recipe sph_ifs_master_detector_flat as described here can be used to create both types of detector flat frames.

The recipe to create the detector flat field for IFS uses input exposures with different illumination levels, taken with the a coloured calibration lamp or, alternatively, the broad lamp, to create master calibration detector flat fields for each of the calibration wavelengths of the input raw frames. The set of these wavelength dependent master calibration frames is used in all subsequent recipes that need to remove the pixel to pixel variation in signal response of the detector. The detector flat field recipe creates master calibration frames, using the input exposures, which have to conform to the DFF_RAW format described in section 6. The master calibration frame is created as multi-extension FITS file.

As a first step, the input list of raw frames are ordered in their mean count levels. A linear fit is then performed to obtain the flat field value for each pixel. The algorithm for this is described in more detail in section 5.11. During this the bad pixel map extension and the RMSMAP extension of the master calibration detector flat are set. Finally the results are written out as FITS files conforming to the standards laid out in section 7. If the hot pixel mask from the dark calibration is provided, the bad pixels identified in the detector flat is compared with the input hot pixel map.

Output:

Name	Multiplicity	Data Type	Further used by	Description
MasterDetector Flat	1	MASTER_DFF	all science calibration	The output master detector flat in master_frame See section 5.3 and 5.4 for more info.
Linearity Bad Pixel Map	1	STATIC_ BADPIXEL _MAP		The static bad pixels flagged as bad in all input frames by the recipe are written into a separate FITS file for instrument monitoring. All flagged bad pixels are also present in the MasterDFF output in the bad pixel extension.

QC1 Parameters:

QC Parameters have not been implemented in this version. This section describes foreseen requirementes.

Name	Data Type	Exit value on error	Further used by	Description
QC_DFF_ MEAN	double[1..N_{DIT},1...N_λ]	0.0	-	The mean value of the resulting detector flat frames: one value per plane in cube and per calibration wavelength
QC_DFF_ STDEV	double[1..N_{DIT},1..N_λ]	0.0	-	The standard deviation value of the resulting detector flat frames: one value per plane in cube and per calibration wavelength
QC_DFF_ UNMATCH-ING_ BADPIX_ COUNT	int	-1	-	The difference between the number of bad pixels identified either only in the linearity method or only in the hot pixel map. Large numbers indicate a possible problem with the pixel linearity. The value is only written if the hot pixel mask is given as input.

10.3 sph_ifs_spectra_positions

Purpose:

to find the positions of the IFU spectra on the detector, defining regions belonging to each spectra and creating the pixel description table (PDT).

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
IFSFlatFrameList	IFS_ SPECPOS_ RAW	Instrument	No	The list of raw instrument flat field frames, obtained with different number of readouts. Frames are assumed to be taken with the broad band lamp.
MasterDarkFrame	IFS_MASTER_ DARK	sph_ifs_ master_dark	Yes	The master dark frame
MasterDarkFrame	IFS_MASTER_ DETECTOR_FLAT_FIELD	sph_ifs_ master_detector_flat	Yes	The broad band DFF frame
LensletModel	IFS_LENSLET_MODEL	special	Yes	A model of the lenslet array (see secrion 7)

parameter list:

Name	Data Type	Default Value	Description
ifs.spectra_positions. outfilename	INT	0: CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN possible
lfs.spectra_positions. clean_mean.reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
lfs.spectra_positions. clean_mean.reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
lfs.spectra_positions. threshold	Double	5.0	The signal/noise threshold for spectra areas. This is used if method is set to "SIMPLE"
lfs.spectra_positions. method	INT	0: "SIMPLE"	cane be either "SIMPLE" or "CORRELATE" corresponding to the thresholding method or the correlation methods described in section 5.6. Cureently only "SIMPLE" implemented.
OutputFilename	STRING	"spectra_ positions.fits"	The output file the resulting pixel description table will be written to.

Description/Algorithm:

This recipe finds the regions on the detector image in which the spectra from the lenslet array fall.

The recipe is only valid for the standard dithering position and there will be one output frame corresponds to each of the settings of detector dithering position, as identified by the INS:DITHER keywords in the IFS instrument flat raw frames (see the definition of Raw data structure in section 6.1). In addition, the raw input frames must all have DIT settings that are consistent with the DIT settings in the input calibration frames given (master dark and master detector flats).

After creating a master frame from the input raw IFS flat frames, similar to the ifs_instrument_flat recipe below, the master dark calibration frame is subtracted, if given. If provided, a broand band detector flat is divided out. The resulting, calibrated flat frames are then used to detect regions of spectra by applying the algorithm in section 5.6. Currently only the “SIMPLE” method is implemented which does a simply thresholding above the value given in the threshold user parameter. The identified regions are then stored and for each connected region, its lowest x and lowest y coordinates are stored. A model of the lenslet array is then used to predict the expected locations of these points. The mean offset in x and y between the actual and expected points is calculated. This is then stored as FITS keywords of a copy of the model pixel description table which is written out.

Note: future versions will also support a overall scale change in the sprecta positions, besides a shift in x and y.

Output:

Name	Multiplicity	Data Type	Further used by	Description
Pixel Description Table	1	PIXEL_ DESCRIPTION_ TABLE	all science calibration	The output pixel description table as a FITS table.

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_NSPECPIXELS	INT	0	-	The number of pixels on the detector identified as inside a spectrum
QC_SPEC_ TILTANLGE_MAX	double	-1	-	The maximal angle between spectra (assuming rectangular regions) and the y-pixel axis.
QC_SPEC_ TILTANGLE_RMS	double	-1	-	The rms angle between spectra (assuming rectangular regions) and the y-pixel axis.

10.4 sph_ifs_instrument_flat

Purpose:

to create the master ifs flat frame. This recipe creates a IFS flat frame, which is not itself divided by detector flat field as well as an IFU flat frame, which is divided by the given master detector flat.

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
IFSFlatFrameList	IFS_FLAT_RAW	Instrument	No	The list of raw instrument flat field frames
MasterDarkFrame	IFS_MASTER_ DARK	sph_ifs_ master_dark	Yes	The master dark frame
MasterDFFFrameList	IFS_MASTER_ DFF_ LONG1, IFS_MASTER_ DFF_ LONG2, IFS_MASTER_ DFF_ LONG3, IFS_MASTER_ DFF_ LONG4, IFS_ MASTER_ DFF_ LONGBB	sph_ifs_ mater_detector _flat	Yes*[1]	The list of master detector frames, one for every calibration wavelength.A must when IFU flat needs to be created, otherwise ignored.
MasterDFFFrameShort	IFS_MASTER_ DFF_ SHORT	sph_ifs_ mater_detector _flat	Yes*	The short broad band flat.A must when IFU flat needs to be created, otherwise ignored.
Calibrated PDT	PIXEL_ DESCRIPTION_ TABLE	sph_ifs_ wave_ calib	Yes*	The standard pixel description table. This must be the pixel description table for the standard dithering position obtained during wavelength calibration. A must when IFU flat needs to be created, otherwise optional.
Uncalibrated PDT	PIXEL_ DESCRIPTION_ TABLE	sph_ifs_ spectra_ positions	Yes*	The standard pixel description table. This must be the pixel description table for the standard dithering position obtained during spectra positions. Ignored when IFU flat needs to be created.

parameter list:

Name	Data Type	Default Value	Description
ifs.instrument_flat. coll_alg	INT	2: CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN possible
ifs.instrument_flat. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.instrument_flat. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.instrument_flat. clean_mean. sigma	DOUBLE	5.0	Sigma clipping value. (DEPRECATED)
ifs.instrument_flat. badpixfilename	STRING	“iff_badpixels.fits”	Name of bad pixel map to be written or not, only used if make_badpix set to yes.
ifs.instrument_flat. make_badpix	BOOL	False	Whether bad pixel map should be written or not
ifs.instrument_flat. badpix_chisqtolerance	DOUBLE	50.0	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.instrument. make_badpix is yes.
ifs.instrument_flat. badpix_uptolerance	DOUBLE	10.0	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.instrument_flat. make_badpix is yes.
ifs.instrument_flat. badpix_lowtolerance	DOUBLE	0.10	The maximal reduced chi square to accept for bad pixel determination. Only used if ifs.instrument_flat. make_badpix is yes.
ifs.instrument_flat. outfilename	STRING	“ifs_instrument _flat.fits”	The output file the resulting master frame will be written to. In case more than one output is written, the name without the .fits extension is used as a base to construct the filenames from.

Description/Algorithm:

This recipe is very similar to the sph_ifs_master_detector_flat recipe. From the input raw IFS Flat Field Frames a set of different sph_master_flat structures is created and saved.

There are two calibration settings that this recipe is used in: in the first case, only a dark subtracted IFS Master Flat is created. In the second case, a flat fielded IFU flat is also created. For the second case, a standard calibrated PDT corresponding to the standard dithering position has to be given in the input. The mode is chosen automatically depending on whether a calibrated or non calibratied PDT is provided.

The recipe is only valid for the standard dithering position and the output frame corresponds to that standard setting of detector dithering position, as identified by the INS.DITHER keywords in the IFS instrument raw frames (see the definition of the data structure in section 6). In addition, the raw input frames must all have DIT settings that are consistent with the DIT settings in the input calibration frames given (master dark and master detector flats).

From the input frames,the recipe creates a master calibration frame. The input master dark frame is subtracted and the standard uncalibrated PDT is used to mask spectra. The flat fielding itself is performed in the following way: the instrument flats at different illumination levels are used to construct a 2nd order polynomial fitting function giving the full (instrument+detector) flat response as a function of mean counts.

Finally, the resulting linear coefficients are saved as a FITS file with the format described in section 6.

In case that the recipe has been used in the IFU flat field mode, the input long flat fields at different wavelength and the short braod band flat field are then used together wtih the calibrated PDT from the wavelength calibration recipe to construct a "super detector flat" which contains the valid flat field value for each pixel at the given dithering position. The instrument flat field found previously is then divided by this flat field. The IFU flat field is saved as a detector image.

Output:

Name	Multiplicity	Data Type	Further used by	Description
MasterIFS Flat	1	MASTER_ IFS_ FLAT	sph_ifs_ wave_ calib	The output master IFS flat as a sph_master_frame format (similar to DFF).
MasterIFU Flat	1	MASTER_ IFU_ FLAT	all science calibration	The output master IFS flat, with the DFF divided out, in a sph_master_frame format.
Linearity Bad Pixel Map	1	STATIC_ BADPIXEL _MAP		The static bad pixels flagged as bad in all input frames by the recipe are written into a separate FITS file for instrument monitoring. All flagged bad pixels are also present in the MasterDFF output in the bad pixel extension.

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_IFSFLAT_MEAN	double[1...N$_{pos}$]	0.0	-	The mean value of the resulting IFS flat frames: one value per plane in cube and per calibration wavelength
QC_IFSFLAT_STDEV	double[1..N$_{pos}$]	0.0	-	The standard deviation value of the resulting IFS flat frames: one value per plane in cube and per calibration wavelength

10.5 sph_ifs_wave_calib

Purpose: To associate different pixels on the detector with the corresponding wavelength of lenslet spectra and fill the pixel description table (PDT) with relevant calibrated pixel information.

type: technical calibration

input data:

Name	Data Type	Source	Optional	Description
WavelengthCalibrationFrameList	IFS_WAVECALIB_ RAW	Instrument	No	The list of raw wavelength calibration field frames, obtained with the wavelength calibration lamp. Has to have standard dithering position.
Standard Pixel Description Tables	IFS_SPECPOS	sph_ifs_ spectra_positions	No	The standard pixel description table for the standard dithering position
MasterDarkFrame	IFS_MASTER_ DARK	sph_ifs_ master_dark	No	The master dark frame
MasterIFSFlatFrame	IFS_INSTRUMENT_ FLAT_FIELD	sph_ifs_ instrument _flat	No	The master IFS flat frame to use.

parameter list:

Name	Data Type	Default Value	Description
ifs.wave_calib.coll_alg	INT	0: CLEAN_MEAN	The algorithm to use when combining the frames – currently only CLEAN_MEAN possible
ifs.wave_calib. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.wave_calib. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.wave_calib.number_lines	INT	4	The number of lines in the calibration spectrum
ifs.wave_calib. wavelength_line1	DOUBLE	1.0	calibration wavelength 1
ifs.wave_calib. wavelength_line2	DOUBLE	1.1	calibration wavelength 2
ifs.wave_calib. wavelength_line3	DOUBLE	1.2	calibration wavelength 3
ifs.wave_calib. wavelength_line4	DOUBLE	1.3	calibration wavelength 4
ifs.wave_calib.wavelength_line45	DOUBLE	1.4	calibration wavelength 5
ifs.wave_calib. line_threshold	DOUBLE	90	count threshold for lines on raw images (DEPRECATED)
ifs.wave_calib. polyfit_order	INT	3	polynomial fitting order for pixel-wavelength fit (DEPRECATED)

Description/Algorithm: This recipe is responsible for the main wavelength calibration. It takes input frames consisting of images of IFS exposures of the wavelength calibration lamp and uses these to associate every pixel within a spectral region with a central wavelength and the wavelength-width covered by the pixel.

The recipe is only valid for the standard dithering position and the output frame corresponds to that standard setting of detector dithering position, as identified by the INS.DITHER keywords in the IFS Raw Flat Frames (see the definition of Raw data structure in section 6.2). In addition, the raw input frames must all have DIT settings that are consistent with the DIT settings in the input calibration frames given (master dark and master instrument flats).

The recipe furthermore does assume that more than 2 emission lines are present in the wavelength calibration spectrum. If less are found the recipe exits with an error message.

In preparatory steps, the input list of raw frames is collapsed using the given collapse algorithm. From this the input master dark is subtracted. The flat fielding is performed by simply dividing by the IFS master flat frame (taken with a white calibration lamp) which calibrates out the detector response and instrument response, $DF(x, y)$and $IF(x, y)$in equation 5.2 at the same time. This can only be accurate if the IFS master flat has been created from exposures that are less than 1 hour old.

After the flat fielding is performed, the spectral description table for the standard detector offset position is used to extract a spectrum (as a pixel image) for each lenslet. The second, short, dimension of the spectra is projected out, giving a set of one dimensional, pixelised, spectra. The identification of pixel wavelength is now performed for each spectrum in two possible ways:

1. If the dispersion and spectra line models are provided, they are used to construct, for ev-

ery lenslet, a model of the spectrum on a pixel grid. Since the dispersion model is mode dependent, a check will be performed to assure the correct dispersion model is used. The dispersion pixel grid is then matched with the pixelised spectrum under consideration. Along the major spectrum axis, pixel information is filled from the model by simply copying the values. Along the short axis, all pixels are assigned the same pixel descriptor information: wavelength, bandwidth and dispersion (or second derivative). [NOT CURRENTLY IMPLEMENTED]

2. If the dispersion model is not provided, the number of local maxima along the major axis of each spectrum is determined. This determination is done within a window of 8 pixels centred on each line given as user paramerter. On continuation, a Gaussian fit is performed to each spectral line thus found, determining the peak and FWHM. The peak position (in pixel coordinates) is then entered into a an array containing in total as many elements as number of lines as provided by the user. A second array of the same size is created from the given peak wavelength of each line. The two arrays then give wavelength-pixel value pairs which are used to perform a polynome fit, $\lambda(x_{pixel}) = \sum c_n x_{pix}^n$. This resulting fit function, with different polynom coefficients c_n for each spectrum detected on the detector is then used to fill the pixel description table with wavelength information. For each pixel, the wavelength is set directly from the wavelength to pixel function,$\lambda(x_{pixel})$. The wavelength width for each pixel is the slope b. Second derivatives are also filled into the table as c.The new, filled, pixel description table for each detector offset is then saved.

Output: A new pixel description table with the new fitted wavelenghts.

Name	Multiplicity	Data Type	Further used by	Description
Calibrated Pixel Description Table	1	PIXEL_ DESCRIPTION_ TABLE	all science calibration	The output pixel description table as a FITS table. Instead of creating a new table, the input tables are modified.

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_WAVECALIB_ FITGOODNESS	double	0	-	The average of the reduced chi squared for the linear fits performed. This measures how well the linear approximation used in the algorithm for this recipe holds.

10.6 sph_ifs_science_dr

This recipe is currently not working correctly. Please do not use.

10.7 sph_ifs_ron

Purpose:

to measure the read out noise.

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
RawFrameList	IFS_ RON_RAW	Instrument	No	The list of raw frames, obtained with different levels of illumination

parameter list:

Name	Data Type	Default Value	Description
ifs.ron. coll_alg	INT	2 : CLEAN_MEAN	The algorithm to use when combining the frames
ifs.ron. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.ron. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.ron. sigma_clip	DOUBLE	5.0	The sigma value to use for sigma clipping to determine the static badpixels.
ifs.ron. outfilename	STRING	"ifs_ron_map.fits"	The output file the resulting file will be written to.

Description/Algorithm:

This recipe is similar to the master_dark recipe, except that the only the rms on the resulting frame is written. The RON is computed using raw frames with different DITs. The input raw dark frame must contain N frames with the same DIT setting and frames with in total at least 2 different DITs settings. A linear fit to the signal as a function DIT is performed and a map of the values at 0s integration time (obtained using the linear fit) is determined. This is repeated for for all N sets of frames and the RMS over the N 0s maps is the readout noise.

Output:

Name	Multiplicity	Data Type	Further used by	Description
RONFrame	1	IFS_RON	none	The output in a master_frame

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_RON_MEAN	double[1..N]	0.0	-	The mean value of the resulting dark frames: one value per plane in cube
QC_RON_STDEV	double[1..N]	0.0	-	The standard deviation value of the resulting dark frames: one value per plane in cube

10.8 sph_ifs_gain

Purpose:

to measure the gain for all pixels.

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
RawFrameList	IFS_ GAIN_RAW	Instrument	No	The list of raw frames, obtained with different levels of illumination

parameter list:

Name	Data Type	Default Value	Description
ifs.gain. coll_alg	INT	2 : CLEAN_MEAN	The algorithm to use when combining the frames
ifs.gain. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.gain. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.gain. sigma_clip	DOUBLE	5.0	The sigma value to use for sigma clipping to determine the static badpixels.
ifs.gain. mean_ binning_ width	DOUBLE	0.0	he bin width to use when binning frames according to their mean counts. If set to 0, no binning is performed.
ifs.gain. outfilename	STRING	"ifs_gain_map.fits"	The output file the resulting file will be written to.

Description/Algorithm:

This recipe determines the gain. All input frames must carry the IFS_GAIN_RAW tag. The input frames are binned according to their mean counts (if mean_binning_width $>$ 0) and then combined using the chosen combination algorithm, rejecting transient badpixels. Sigma clipping is used to detect static bad pixels, just like for the sph_ifs_master_dark recipe. Using the binned frames, a photon transfer curve is derived and the gain is read off from this curve and saved as a SPHERE master frame (FITS file with 4 extensions). The overall gain is stored as a keyword in the header. Please note that this determination of the gain does not take inter-pixel-capacitance (IPC) into account.

Output:

Name	Multiplicity	Data Type	Further used by	Description
GainFrame	1	IFS_GAIN	none	The output in a master_frame

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_GAIN_MEAN	double[1..N]	0.0	-	The mean value of the resulting dark frames: one value per plane in cube
QC_GAIN_STDEV	double[1..N]	0.0	-	The standard deviation value of the resulting dark frames: one value per plane in cube

10.9 sph_ifs_detector_persistence

Purpose:

to measure the persistence of the detector.

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
RawFrameList	IFS_ DETECTOR_ PERSISTENCE_ RAW	Instrument	No	The list of raw frames, obtained with different levels of illumination

parameter list:

Name	Data Type	Default Value	Description
ifs.detector_ persistence. coll_alg	INT	2 : CLEAN_MEAN	The algorithm to use when combining the frames
ifs.detector_ persistence. clean_mean. reject_low	INT	0	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.detector_ persistence. clean_mean. reject_high	INT	0	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.detector_ persistence. sigma_clip	DOUBLE	5.0	The sigma value to use for sigma clipping to determine the static badpixels.
ifs.detector_ persistence. fitorder	INT	2	The fitting order to use when fitting the fall-off.
ifs.detector_ persistence. outfilename	STRING	"ifs_detector_persistence_map.fits"	The output file the resulting file will be written to.

Description/Algorithm:

When given one illuminated and saturated frame and a series of at least 5 frames without illumination the recipe produces a map of the linear fall-off coefficient for the signal. The frames must contain valid time stamps in "year-month-dayThour:min:sec" and the ESO SPH LAMP ON keyword which should be on only for the saturated frame. For all other frames it should be set to off.

Output:

Name	Multiplicity	Data Type	Further used by	Description
PersistenceFrame	1	IFS_ DETECTOR_ PERSISTENCE	none	The output in a master_frame

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_MEAN	double[1..N]	0.0	-	The mean value of the resulting frames.
QC_STDEV	double[1..N]	0.0	-	The standard deviation value of the resulting frames.

10.10 sph_ifs_cal_background

Purpose:

to measure the instrument background.

type:

technical calibration

input data:

Name	Data Type	Source	Optional	Description
RawFrameList	IFS_ CAL_ BACKGROUND_ RAW	Instrument	No	The list of raw frames obtained in the same way as darks. At least 2 must be given.
MasterDark	IFS_ MASTER_DARK	Instrument	Yes	An optional master dakr

parameter list:

Name	Data Type	Default Value	Description
ifs.cal_ background. coll_alg	INT	2 : CLEAN_MEAN	The algorithm to use when combining the frames
ifs.cal_ background. clean_mean. reject_low	INT	1	The number of pixels rejected at the low end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.cal_ background. clean_mean. reject_high	INT	1	The number of pixels rejected at the high end. Parameter for the clean mean algorithm. Only used if CollapseAlgorithm set to CLEAN_MEAN.
ifs.cal_ background. sigma_clip	DOUBLE	5.0	The sigma value to use for sigma clipping to determine the static badpixels.
ifs.cal_ background. fitorder	INT	2	The fitting order to use when fitting the fall-off.
ifs.cal_ background. outfilename	STRING	"ifs_detector_persistence_map.fits"	The output file the resulting file will be written to.

Description/Algorithm:

This recipe is responsible for measuring any unwanted background instrument signal so it can be subtracted from science frames if needed. Since this functionality is required only for very specific observations mainly of extended sources, the result of this recipe are currently not used as input into any other recipe – observers will have to manually use these frames to subtract the unwanted background. The recipe is similar to the sph_ifs_master_dark recipe, except that the image is taken with very long exposure time. However, before the resulting frame is collapsed from the raw frames an optional dark frame is removed. This frame should be taken with the same read out mode, but short DIT.

Output:

Name	Multiplicity	Data Type	Further used by	Description
BackgroundFrame	1	IFS_ CAL_ BACK-GROUND	none	The output in a master_frame

QC1 Parameters:

Name	Data Type	Exit value on error	Further used by	Description
QC_MEAN	double[1..N]	0.0	-	The mean value of the resulting frames.
QC_STDEV	double[1..N]	0.0	-	The standard deviation value of the resulting frames.

10.11 sph_ifs_distortion_map

This recipe is currently not working correctly. Please do not use.

Chapter 11

Installation Procedure and Examples

11.1 Install Pipeline

To install the pipeline, please follow the following instructions (also on http://www.mpia-hd.mpg.de/~moeller/sphere_install.html):

Once you have downloaded (from http://www.mpia-hd.mpg.de/~moeller/sphere-kit-0.5.1.tar.gz) or gotten otherwise a tarball unpack it and cd to the main directory

```
tar -xvzf sphere-kit-0.5.1.tar.gz
cd sphere-kit-0.5.1
```

In the directory you will find a script called install_pipeline. Run it with

```
./install_pipeline
```

It will prompt you for a directory to install the pipeline in. We recommend choosing as name /home/myself/sphereP where home/myself is your home directory. All software should now be installed automatically. This will take a while.

A special note for mac users: if you have a mac, execute the command “make install-mac” in sphere-kit-0.5.1/sphere-0.5.1 after the install_pipeline command has finished.

When everything is done, and you are alright with waiting for a while, yao may do the following to make sure it all worked: in the sphere-kit directory (from the unpacked tarball) type

```
cd sphere-0.5.1
make check
```

and hopefully it should end with a statement "All 23 tests passed" or so. But be warned that this may take quite a long time ! If it fails, please do contact us (moeller@mpia-hd.mpg.de or pavlov@mpia-hg.mpg.de).

Setting everyting up to run recipes with esorex

To run recipes with esorex you first need to set some environment variables. How you do this in detail depends on your shell. In the description here we assume you have bash. To set the enviroment variables, add this to your .bashrc file:

```
export LD_LIBRARY_PATH=/home/me/sphereP/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/home/me/sphereP/lib/sphere/plugins/sphere-0.5.1:
```

```
                                    $LD_LIBRARY_PATH
export PATH=/home/me/sphereP/bin:$PATH
export ESOREX_PLUGIN_DIR=/home/me/sphereP/lib/sphere/plugins:
                                    $ESOREX_PLUGIN_DIR
```

Here you should substitute /home/me/sphereP with the directory where you installed everyting (the name you gave when you ran the install_pipeline script). If you have a apple Mac, change LD_LIBRAY_PATH to DYLD_LIBRARY_PATH. To test it, open a new xterm and type

```
esorex --recipes
```

and it should give

```
***** ESO Recipe Execution Tool, version 3.7.1 *****
List of Available Recipes :
sph_ifs_master_detector_flat  : Creates the master detector flat
                                from a list of input frames
sph_ifs_master_dark  : Creates the master dark from a list of
                       input dark frames
```

For more info on ESOREX, please see ESOrex's homepage (http://www.eso.org/sci/data-processing/software/cpl/esorex.html) or just type

```
esorex --help
```

11.2 Getting the example data

To execute the example you also need to obtain the example data. This can be obtained from http://www.mpia-hd.mpg.de/~moeller/example_easter2010.tar.gz

Unpack it in an empty directory with

```
tar -xvzf example_easter2010.tar.gz
```

11.3 Creating the master dark

The first step is to create the master dark frame from the input frames. To that end, in the directory with the example data, create a text file with the name mydark.sof with the following content:

```
raw_dark_DIT_0.fits   IRD_DARK_RAW
raw_dark_DIT_1.fits   IRD_DARK_RAW
raw_dark_DIT_2.fits   IRD_DARK_RAW
raw_dark_DIT_3.fits   IRD_DARK_RAW
raw_dark_DIT_4.fits   IRD_DARK_RAW
raw_dark_DIT_5.fits   IRD_DARK_RAW
raw_dark_DIT_6.fits   IRD_DARK_RAW
raw_dark_DIT_7.fits   IRD_DARK_RAW
raw_dark_DIT_8.fits   IRD_DARK_RAW
raw_dark_DIT_9.fits   IRD_DARK_RAW
```

Now run

```
esorex sph_ird_master_dark mydark.sof
```

to execute the recipe. By the way: A call with

```
esorex --help sph_ird_master_dark
```

provides you with a help page.

The recipe will run for a less than a minute. It will then output:

```
[ INFO  ] esorex: Created product /home/geminitest/example_easter2010/out_0000.fits
[ INFO  ] esorex: Created product /home/geminitest/example_easter2010/out_0001.fits
```

The first of these is the master dark, the second the badpixels in a seperate file. Rename out_0000.fits to master_dark.fits and view it with your favourite FITS viewer and compare with the input files to verify that it is indeed the mean of the inputs.

11.4 Creating the master flat

The procedure to create the master flat is similar to that for creating the master dark. First, create a myflat.sof file with the following content

```
master_dark.fits            IRD_MASTER_DARK
raw_flat_bright_DIT_0.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_1.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_2.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_3.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_4.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_5.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_6.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_7.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_8.fits  IRD_FLAT_FIELD_RAW
raw_flat_bright_DIT_9.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_0.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_1.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_2.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_3.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_4.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_5.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_6.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_7.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_8.fits  IRD_FLAT_FIELD_RAW
raw_flat_faint_DIT_9.fits  IRD_FLAT_FIELD_RAW
```

Note that there are now both “faint” and “bright” frames that are instrument flat frames with different illumination levels. The flat recipe needs at least 2 different illumination levels to work. Also note that the first line places the result from the master dark run as input to the flat recipe.

Now run

```
esorex sph_ird_instrument_flat myflat.sof
```

to create a new out_0000.fits, containing the flat field. Again, verify the result with your favourite FITS viewer and rename it to master_flat.fits.

11.5 Executing the science recipe

Last but not least, to execute the science recipe, another .sof file is needed. To begin with, only use a part of the available raw frames. Create mydbi.sof and set as content:

```
raw_science_dbi_DIT_0.fits   IRD_SCIENCE_DBI_RAW
raw_science_dbi_DIT_1.fits   IRD_SCIENCE_DBI_RAW
raw_science_dbi_DIT_2.fits   IRD_SCIENCE_DBI_RAW
raw_science_dbi_DIT_3.fits   IRD_SCIENCE_DBI_RAW
raw_science_dbi_DIT_4.fits   IRD_SCIENCE_DBI_RAW
master_flat.fits   IRD_FLAT_FIELD
master_dark.fits   IRD_MASTER_DARK
```

Note the last two lines. Here the outputs of the two previous recipes is used as input to the science recipe.

Execute the recipe as before with

```
esorex sph_ifs_science_dbi mydbi.sof
```

The recipe does take quite a bit longer to run than the previous ones. This is because the recipe is de-rotating the frames and uses ADI to subtract out the "speckle" pattern. Both de-rotation and ADI can be switched off using the parameters. See the manual for the parameters and the esorex documentation on how to set them.

When the recipe has finished, two outputs are created: "out_0000.fits" and "out_0001.fits" corresponding to the left and right fields repsectively. The two should be identical since the input raw frames were simulated with identical filters for both the left and right optical path.

We encourage you to now try to use different options for the science dr recipe: provide some reject_high value that is larger than 1 for example (this should remove the "dark planets") or provide a filter_radius.