Observatoire de Genève
51, Ch. des Maillettes, CH-1290 Sauverny
Phone: +41 22 755 26 11 * Fax: +41 22 755 39 83

GIRAFFE

Functional Specification for BLDR Software

Doc. No. VLT-SPE-OGL-13730-0030
Issue 1.1
November 9, 1999

Approved at PDR
compiled on November 17, 1999 at 17:33

	Name	Date	Signature
Prepared	A.Blecha, P.North, F.Royer, G.Simond, V.Cayatte	9 November 1999	
Approved	A.Blecha	9 November 1999	
Released	A.Blecha	9 November 1999	

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Remarks
1.0	Oct 4, 1999	All	Result of internal releases 0.1-0.3; sent to ESO October 6, for the PDR October, 28 1999. Available at http://obswww.unige.ch/Instruments/GIRAFFE/Files/doc/ PDR/fs100.ps
1.1	Nov 9, 1999	9.5, 10.17, 14	Modifications considering the RIDs 2.1 and 2.7 from the SW-PDR

Table of Contents

1 Introduction (A. Blecha)

1.1 Organization of this document, level of treatment and completeness

Section 1 gives the reference information and some introductory and working remarks. Section 2 is a brief description of the BLDRS functions and the *boundaries* and interactions with other parts of the GIRAFFE and VLT software. Sections 3 to 5 give the description of functions in, some details including functions that will not be subject to the development (available from VLT general or public-domain SW) but are necessary for the BLDRS. The standard pipelines recipes are given in section 9. The following detailed description (section 10) for the *core* functions is generated from the BLDRS SW database and is completed by the data tables and index.

Only for *core* functions the algorithms are described and almost-complete lists of Input/Output are given for each function.

The modules that require the interfacing to the VLT SW are not analyzed at the same level of detail as the *core* functions partly because this task will be done directly under ESO control, partly because the final definitions of these interfaces are not yet available.

Since the document is also heavily used for the internal discussions of the GIRAFFE and FLAMES consortia, the authors of sections are identified within the text. If several names are given, the first name is to be contacted in priority. If a name is given at the subsection or paragraph, it applies only to this specific part of the text.

1.2 Purpose

The present document (FS-BLDRS) specifies the functions provided by the Base-line Data Reduction Software (BLDRS), part of the Data Reduction Software (DRS), for the Giraffe Spectrograph which is a part of the VLT Multi Object Fibre Facility (called FLAMES) at UT2 telescope.

It provides the basis for the detailed design and implementation of the BLDRS.

1.3 Scope

The FLAMES facility is made up of the fibre unit pick-up feeding the high resolution (10000-40000) spectrographs. The fibres may be fed by the light of individual objects (MEDUSA mode - 132 fibres with one fibre per object and 5 simultaneous calibration fibres), one single resolved object (ARGUS - 300 image elements on one object, 15 sky fibres and 5 simultaneous calibration fibres) or 15 small resolved objects (IFU - 20 image elements, 1 sky per object + 5 simultaneous calibration fibres). In MEDUSA mode the spectrograph could be Giraffe or UVES (8 objects - no simultaneous calibration fibres). The present specifications do not cover the UVES DRS.

The BLDRS should account for all instrumental effects and produce the flux and wavelength calibrated spectra in all standard observing modes.

The BLDRS will be complemented (not covered by this document) by the Ancillary Data Analysis Software (ADAS) which will provide the Data Analysis tools specific to the instrument such as High-precision Radial velocity package and image reconstruction in a given passband (ARGUS and IFU modes). However, at least crude image reconstruction will be provided by the BLDRS and by the Quick-Look Software for Argus and IFU modes.

The BLDRS will be used in two different ways.

1. In the VLT DFS pipeline as a set of SW modules (to be integrated to the DFS under the responsibility of ESO)

2. In the MIDAS pipeline as a data-reduction package callable from MIDAS and fitted with a graphical user interface (GUI).

In both cases the functions are identical. We assume that the DFS pipeline will not offer any interactivity and therefore some functions will not be used there.

1.4 Applicable and Reference Documents

1.4.1 Explanatory remarks

There are no appropriate tools to handle efficiently the VLT documentation and some reference or applicable documents are incompletely identified (version, dates missing). No list exists where documents could be traced and their status checked in order to know when old documents were superseded by new ones, when not yet available but already referenced documents are due to be released and what is the access level to documents.

1.4.2 Applicable Documents (G. Simond)

The Applicable Documents give the information describing the work to be carried out, the available tools and the formal conditions of the work.

AD1 `INS-SPE-ESO-13730-1657,1.0` – GIRAFFE Technical Specifications
AD2 `VLT-SPE-ESO-10000-0004,1.0` – VLT Specification Environmental Conditions
AD3 `VLT-TRE-ESO-13740-1681,1.0 06/10/98` – GIRAFFE SPECTROGRAPH Optical Final Design
AD4 `VLT-PLA-ESO-13700-1788,1.0` – FLAMES IOCP - Instrument Operation and Calibration Plan
AD5 `INS-PLA-AUS-13721-0056 1.0 29/09/99` – FLAMES Fibre Positionner OpCalPlan
AD6 `GEN-SPE-ESO-19400-0794,1.1 25/11/97` – ESO DICD - Data Interface Control Document
AD7 `ARC-SPE-ESO-00000-0001,1.4 10/03/94` – ESO Archive - Data Interface Requirements
AD8 `VLT-SPE-ESO-19600-1233,0.3 15/10/96` – VLT DFS Specifications for Pipeline and Quality Control
AD9 `VLT-PLA-ESO-19000-1183,1.0 09/12/96` – VLT DFS Operations Model for VLT/VLTI Instrumentation
AD10 `VLT-SPE-OGL-13730-0020,2.0 02/10/99` – GIRAFFE User's Requirements for BLDR Software
AD11 `VLT-MAN-ESO-17240-0866,2.5 28/07/97` – Real Time Display - User Manual
AD12 ESO MIDAS November 1998 – Data Reduction Manual
AD13 `VLT-SPE-ESO-19000-1618,1.0 21/04/99` – Data Flow for VLT Instruments Requirements Specification
AD14 `VLT-SPE-ESO-19000-0749,1.0 ????????` – VLT On-line Data Flow Requirements Specification
AD15 `VLT-MAN-ESO-19500-1619 1.0 02/26/99` – Data Flow Pipeline and Quality Control - User's Manual
AD16 `VLT-MAN-ESO-19400-1785,1.0 23/02/99` – OLAS User's Guide
AD17 `VLT-SPE-ESO-19600-1217,1.0 ????????` – Data Flow System Quality Control Equipment Model API
AD18 `VLT-MAN-ESO-19500-1771,1.2 18/06/99` – FORS Pipeline & Quality Control - User's Manual
AD19 `VLT-MAN-ESO-19500-1772,1.0 25/02/99` – ISAAC Pipeline & Quality Control - User's Manual
AD20 `VLT-PLA-ESO-10000-0441,1.0 05/12/97` – Science Operation Plan

1.4.3 Reference Documents

The Reference Documents are *bibliographic references* in the classical sense of term. They supply the information complementary to the text.

RD1 `VLT-PLA-ESO-00000-0006,2.0 21/05/92` – VLT Software Management Plan
RD2 `VLT-PRO-ESO-10000-0228,1.0 10/03/93` – VLT Software Programming Standard
RD3 2dF User Manual, J. Bailay, K. Glazebrook, May 29, 1998
RD4 `VLT-TRE-VIRG-14616-0058,1.0 16/06/98` – VIMOS Final Design Review
RD5 Australis Concept Study Report, The Australis Consortium), K. Taylor, M. Colless et al., 27 Sept. 1996
RD6 `INS-TRE-OP-137400-0009,1.0 expected for 28/10/1999` – FLAMES simulator Design and Architecture Report, V. Cayatte
RD7 Horne K., 1986, PASP 98, 609 Optimal extraction
RD8 `GIR-TRE-OPM-XXXXX,1.0` – Qualité d'image en bord de ferrule, D. Jocou (OP), 27 July 1999
RD9 Sky PSF modeling – Viton & Milliard (LAS, Marseille), 1999, private communication

1.5 Abbreviations, Acronyms and Glossary

ADAS	GIRAFFE Ancillary Data Analysis Software - Data Analysis tools specific to GIRAFFE instrument
ADU	CCD Analog-to-digital Unit - Units used to quantify the CCD signal intensity

ARGUS	GIRAFFE Integral Field Spectroscopy mode
AUSTRALIS	Australian project for fibre coupled spectrograph
BLDR	GIRAFFE Base Line Data Reduction
BLDRS	GIRAFFE Base Line Data Reduction Software
CRH	Cosmic Ray Hit (general term)
DAL	Data Access Layer - part of the GIRAFFE DRS
DFS	Data Flow Software - VLT software controlling the on-line data processing
DICB	Data Interface Control Board - VLT software
DMD	Data Management Division of ESO
DRS	Data Reduction Software (general term)
FF	Flat-Field (general term)
FITS	Flexible Image Transport System - file format for images and tables (general term)
FLAMES	VLT Multi Object Fibre Facility - UT2-VLT Instrument; GIRAFFE is a part of FLAMES
FORS	Focal Reducer Spectrograph - UT1-VLT instrument
FOV	Field of View (general term) used here in sense of Nasmyth FOV
FP	Fibre positionner - part of FLAMES instrument
FS-BLDR	Functional Specifications for GIRAFFE Base Line Data Reduction
FSFF	*Full-Slit* Flat Field - FF taken with entrance slit uniformly illuminated
GIRAFFE	Spectrograph - this instrument, part of FLAMES
GUI	Graphical User Interface - set of command windows controlling the DRS (general term)
IFU	Multi-Integral Field Unit mode of GIRAFFE (15 IFUs of 20 image element + sky each)
IOCP	Instrument Operation and Calibration Plan (general term) but referenced here for FLAMES
ISAAC	Infrared Spectrometer And Array Camera - UT1-VLT instrument
KW	Keyword (usually FITS KW) - part of the FITS format
LSQ	Least Square Fit (general term)
MEDUSA	Multi-Object Spectroscopy mode of GIRAFFE (one fibre per object)
MIDAS	Munich Image Data Analysis System - Standard ESO Image Processing System
MOS	VLT Multi-object slit spectrograph (general term)
NFF	*Narrow* Flat Field - FF taken with slit illuminated through fibres
OGL	Observatoire de Genève et l'Université de Lausanne
OP	Observatoire de Paris
PAF	Parameter File Format - VLT DFS standard format
PC	Process Control - part of the BLDRS, set of tools that control the BLDRS execution
PE	Processing Engine - core set of functions of BLDRS
PHFF	Photometric Flat Field - FF taken with CCD illuminated directly (without passing through spectrograph)
PSF	Point Spread Function - image resulting from monochromatic point-like illumination of the entrance-slit
QC0	Quality Control Level 0 - results form the comparison of current parameters with nominal values
QC1	Quality Control Level 1 - results form the comparison of trends within the Calibration Database with expected fluctuations
QLS	GIRAFFE Quick-Look Software - simplified reduction with purpose to asses the data quality
RON	Read-Out Noise - CCD noise measured on 0 time exposure or on the image overscan
RTD	Real-Time Display - VLT tool for display and graphical handling of images
RR	Reduction Recipe - DFS standard *scripts* linking the DRS function into a procedure executable by the DFS
SEWC	Separate Wavelength Calibration - Calibration made with all fibres illuminated by calibration lamp
SIWC	Simultaneous Wavelength Calibration - Calibration made with 5 dedicated fibres illuminated by calibration lamp
SL	Scattered Light - parasite light with unknown spectral distribution falling on the detector
SLM	Scattered Light Model - continuous model built from discontinuous SL measurements
SNR	Signal to Noise Ratio - (general term)
SOS	Flames Super Observing Software - SW controlling the FLAMES through requests to sub-systems

SPU	Spectrograph unit - GIRAFFE spectrograph
SW	SoftWare
TBC	To Be Confirmed
TBD	To Be Defined
UR	GIRAFFE User's Requirements - generic terms for UR-BLDRS, UR-ADAS
UT1-4	VLT Unit Telescope 1 to 4
UVES	U-V Echelle Spectrograph - Third UT2-VLT instrument coupled to Fibre Positionner of FLAMES (8 fibres)
VCS	VLT Control Software
VI(R)MOS	Visual (Infrared) Multi Object Spectrograph - UT3-VLT multi-slit deep low-resolution spectrograph due for 1 Quarter 2000 and 3 Quarter 2001 (IR part)
WLC	Wavelength Calibration - (general term)

1.6 Typing conventions

The Input/Output data are described using typing conventions

typing/symbol	explanation
imageTest(X,Y)	2D object (light distribution) in the focal plane
imageTest(x,y)	2D image from the detector (with $x \parallel$ dispersion direction)
imageTest(x,n)	2D Set of $n \times 1D$ extracted uncalibrated spectra.
ImageTest(x,n)	2D Set of $n \times 1D$ flux calibrated spectra
imageTest(λ,n)	2D Set of $n \times 1D$ wavelength calibrated spectra
ImageTest(λ,n)	2D Set of $n \times 1D$ wavelength and flux calibrated spectra
$\langle image(x,y) \rangle$	Optional term within an expression
$\overline{image(x,y)}$	Average value (a number)
$\overline{image}(x,y)$	Average image
$A \cup B$	Logical "or" element-by-element of common part of matrix A and B

Whenever possible the dimensions of object are given using the general numbers

symbol	expected value	explanation
Nx	4096	Number of pixels of the detector in direction parallel to the dispersion
Ny0	2248	Number of pixels of the detector perpendicular to the dispersion with overscan lane
Ny	2048	Number of pixels of the detector perpendicular to the dispersion
Ns	137/320/320	Number of spectra Medusa/Argus and IFU

1.7 Preliminary and working remarks

In theory the specifications in this document should be based on written documents (applicable and reference documents) and on traceable and certified information issued by VLT division or third parties. In practice, unfortunately, the pieces of information are of unequal quality and completeness and some of them are missing or exist only in preliminary or obviously out-of-date versions.

We identify 3 interfaces that should be properly addressed:

- BLDRS/IOCP
- BLDRS/Calibration Data Base
- BLDRS/Instrument Optical Final Design

The purpose of the following subsections is to list the critical elements of information not available at the desirable level and sort them in two categories; those which we expect will be completed and those which will remain poorly defined and therefore present a real challenge for the Data Reduction.

1.7.1 The information that will be completed

- The GIRAFFE Technical Specifications (AD1) do not contain detailed enough specifications. We expect that the GIRAFFE SPECTROGRAPH Optical Final Design (AD3) will clarify all raised questions:
  - The overall instrument PSF (fibre, spectrograph) and its spatial variation and stability
  - The accuracy of the reference spectra and stability of the calibration lamps
  - The concept and the specifications of the Nasmyth screen
- The calibration plan Issue 1.0 FLAMES Instrument Operation and Calibration Plan (IOCP - AD4) does not contain enough of details concerning the calibration procedures. It may be useful to produce a release 2 of the AD4 unless the functional specifications for the FLAMES and GIRAFFE OSes address the following points:
  - Detailed description of the calibration with fibres positioned on the plate (Nasmyth Flats)
  - The variation of the fibre transmission between the close-packed (standard transmission calibration) position and the field position as well as the variation during the exposure (twist due to the rotation)
  - Fibre transmission: the detailed descriptions of the calibration procedures are missing. The accuracy of the relative fibre transmission should be assessed; within the document, 0.5% or 1% are mentioned which is better that what was required in the GIRAFFE Technical Specifications (2%) and much better than the stability announced with the existing instruments (10%). If 0.5% could be guaranteed, the calibration would be substantially simplified.
  - Sky subtraction: the detailed descriptions of the calibration procedures are missing. The problem of the PSF variation is not addressed.
  - Calibration of the Nasmyth screen (illumination map)
  - The 5 fibres used for the simultaneous calibration. Calibration of 5 fibres (both FF and wavelength) used for the simultaneous wavelength calibration is not described. Only the calibration using the FP lamps is discussed in the Calibration Plan.
  - Laboratory (initial) calibrations
  - A global view of the Calibration Procedures should be elaborated where at least the following are described and addressed:
    * What are the calibration data needed to the data reduction in all instrumental modes. This is partly supplied by the present document, but should be cross-checked by a detailed calibration plan.
    * Level 0 data - What data will be supplied at the once-for-all basis from the laboratory measurements ?
    * Level 1 data - What calibration will be done during the maintenance runs, at what frequency, and what will be monitored parameters ?
    * Level 2 data - What calibration will be done routinely as a part of observation day-time or twilight preparation and how are they related to the level 1 calibrations and how will the database of average calibration data be built and maintained ?
    * Level 3 data - Calibrations done during the night.
    * Evaluate the data volume and acquisition time necessary for calibration in all modes taking into account the time necessary to reach maximum S/N
    * What data will be kept on-line, archived, discarded ?
    * Detailed description of the calibration procedures with quantifiable parameters (avoiding the *several, good, various*)

1.7.2 The information that will remain unavailable

- Data archive

  The existing documentation (AD6,7) does not permit to include the adequately interfacing to the archive into our design. We assume, that there will not be soon the possibility to use the archive on-line through standard calls, not subject to possible changes during the design of the GIRAFFE DRS. This is the reason why we foresee a generic data-access layer that will insure this role and could be adapted when the archive access will become stable enough.

- Atmospheric Transmission Monitor

  The standard transmission curve taking into account the measured meteo parameters will be used for the uncalibrated spectro-photometry.

2 General overview (A. Blecha, P. North)

Traditionally, the DRS is an off-line package and though it will be used within the framework of the VLT acquisition process, the on-line aspect has only little impact on the DRS requirements. The configurable level of interactivity going from unattended automatic mode during the standard acquisition to full interactivity during the instrument maintenance runs and robustness required for a fully automatic operation are the main features that distinguish the Giraffe BLDRS from the classical off-line approach.

2.1 Interfaces

2.1.1 Inputs

On the input, the BLDRS receives the following data:

- The raw images from the VLT acquisition pipeline.

  The acquisition pipeline is a part of the VLT DFS and is running under the control of the FLAMES Super Observing Software (SOS). The BLDRS communicates with the SOS through data only. All the necessary *registration data* (Objects ID, positions, magnitudes) are supplied via FITS keywords on raw images.

- The calibration data.

  The necessary calibration data and standard control parameters are extracted from the instrument database. Two sorts of calibration data are used:

  - The current calibrations.

    Current calibration files are the data obtained from the calibration pipeline. Most recent, standard or average data may be used. An example of these are the flat-fields, localization or wavelength solutions.

  - The calibration constants.

    These are the data which are not obtained from the normal calibration pipeline. They are not strictly speaking constant, but could only be changed during the maintenance run through a special calibration process or imported by an ad-hoc command. The calibration constants are set before any standard Data Reduction could proceed. An example of such constants are the table of the spectrograph optical parameters, the reference absolute spectrum of the calibration lamps, the tables of the spectral lines for the wavelength calibration and initial parameters for the processing algorithms.

- The control parameters.

  In order to make the automated processing possible and efficient, all options and controls are fully parameterized. A subset of comprehensible and widely accepted *classical* main parameters is put in evidence separately in order to facilitate the handling by technically inexpert user (with necessary astronomical background). An example is the level of the sigma-clipping, the method of the rebinning or the accepted level of the scattered light contamination.

There is no difference, except for the raw images and the control parameters access, between the use of the BLDRS within the VLT or MIDAS pipeline.

2.1.2 Outputs

The BLDRS produces the following data:

- Flux and wavelength calibrated spectra with optionally error and number of pixels in each bin
- Standard set of numerical values that characterize the results and are used for the long term performance monitoring and quality control
- Set of report/history files and graphics

Optionally all necessary calibration and control data can be exported for further off-line use.

2.2 Main components

- Data access layer (DAL)

  This layer is included in order to make the BLDRS independent of the environment and of the access to the instrument data-base. The BLDRS works only with standard and simple data structures (FITS for images and ASCII for tables and configuration files).

- The Processing engine (PE)

  This SW provides all computation tools necessary to carry out the full data reduction. It could be divided in:

  - The Preprocessing
  - The Localization
  - The Extraction
  - The Wavelength calibration and rebinning
  - The global Flat-Fielding
  - The Sky modeling and subtraction
  - The Flux spectrophotometric calibration
  - The Image reconstruction

  Note that PE includes the tools necessary to the quality control and performance assessment.

- The graphics and image displays (GID)

  Beside the standard components such as the classical graphical tools and the image displays which are available in the VLT Common Software (AD10) and in the MIDAS (AD12) as well as in many public-domain environments, there are components specific to the intrinsic 3-D nature of the data to be displayed which are:

  - The Graphical field navigator
    This tool displays the field according to the information provided by the input data (object description in MEDUSA and IFU mode, the integrated image in ARGUS mode with optionally in both cases the field image in the background) and enables display functions on part or totality of data selected graphically.
  - The Full graphical reporter
    This tool displays all or part of data in a specified mode.

- The graphical user interface (GUI) both for scientific and maintenance mode.

- The External Components

  This includes all public-domain or VLT standard SW used within the BLDRS.

2.3 Quality control

Virtually all high level functions return parameters that are used for the quality control. They are described in the sections giving the detailed functional specifications.

The quality control level 0 (QC0) is a part of the normal data reduction processing. This is done for all processed frames including the science frames in the DFS pipeline. With the exception of the FF calibration exposures, any GIRAFFE frame includes 5 optimally exposed spectra used for the simultaneous wavelength calibration. This is the opportunity for a full check of the actual instrument performances compared to the nominal ones:

1. read-out noise, CCD offset and average dark
2. number of cosmic hits and detector defects

3. proportion of the scattered light
4. spectra localization stability
5. instrument throughput (integrated light per spectrum/calibration illumination)
6. wavelength solution stability and spectral resolution

The quality control level 1 (QC1) detecting trends and producing the average certified calibration data is not a part of the DRS.

2.4 The Back-up solutions

As mentioned in section 1.7 some details in the technical specifications, the calibration plan or in the instrument design are not yet fully clarified and may remain so till the end of the DRS design phase. This is the reason, why we mention specifically within this document, whether a back-up solution exists, when some requirement is not yet guaranteed. The support of back-up solution is a part of the functional specifications, being agreed that they may be removed during the design phase if the point is clarified soon enough. It is worth noticing that in some cases **no back-up solutions** exist.

3 Data access layer - DAL (G. Simond)

The BLDRS works with standard and simple data structures. The VLT FITS format with ESO HIERARCH keywords is used for observation, calibration frames, data tables, and short-FITS/VLT Parameter File Format (PAF) for configuration and parameters files. As described by ESO DICB and DMD, all files generated by the BLDRS modules will adopt the VLT FITS, PAF or plain ASCII format and the VLT file naming conventions [AD6,AD7].

At the telescope, the DFS framework ensures that the necessary input data: science and calibration frames, parameters and configuration files, are available, retrieved from the VLT On-line Archive System (OLAS) for each reduction step and that the data products are stored and archived using the VLT OLAS. All the reduction steps applied and results obtained as well as reduction status and errors are stored with time-stamps in log files using the VLT logging system.

For the off-line reduction process, all the necessary data products (observation and calibration frames, observation parameters, instrument data configurations, conditions and log files) will be extracted from the VLT OLAS or VLT science archive and will form the data-set delivered to the astronomer.
For maintenance or tests at the telescope, the required data will be available from disk or from VLT OLAS.

The BLDRS Data Access Layer (BLDRS-DAL) will provide all data access functions required by the BLDRS to read, write and manipulate FITS formatted data coming from the DFS or from the astronomer data-set disk files, thus all reduction modules should be able to run within or outside the VLT DFS pipelines.

Though the large number of observing modes (high and medium resolutions in all spectral ranges in MEDUSA, IFU and ARGUS modes and for all fibre assemblies) will require a large number of calibration files, the organization of the data-set should provide information on how data products are related to each other. This can be achieved using a standard directory structure, file naming convention and/or data catalogs.

4 The Processing engine - PE (A. Blecha, P. North)

This SW includes all necessary computational functions. The specifications follow closely the document User's Requirement for BLDRS. **In case of discrepancies between the UR and FS (this document), the FS is to be considered as the up-to-date reference.**

The Baseline Data Reduction (BLDR) consists in *removing the instrument signature from the data and conversion to the physical units.* The output of BLDR is a set of flux and wavelength calibrated sky-subtracted spectra. The following general remarks apply:

- The processing will be the same in both low and high resolution modes.
- With a few exceptions the MEDUSA, ARGUS and IFU data will be reduced using the same SW functions. Only the control parameters and calibration data will be different.
- Associated to the processed image, the numerical mask badpixel(x,y) of the *bad pixels* is initialized and progressively updated through the reduction process. The analysis software supports *flagged* pixels all through the analysis. The flagging is implemented in a way to identify the reason of point rejection by value of the mask, while the mask itself will be used as a logical operand (points with value greater than 0 are rejected). Alternatively, the possibility to replace flagged-pixel at any stage of the data reduction will be implemented.

4.1 The Preprocessing

All raw images will go through the same steps of pre-processing. The order of steps is under the control of the Reduction Recipes (RR) and some steps may be merged together for the sake of efficiency. The following steps of pre-processing will be carried out:

- Get and subtract bias
- Initialize the bad pixel mask
- Check and subtract the dark current
- Cosmic ray detection
- Adjust the scattered light
- Remove the scattered light
- Spectroscopic pixel-to-pixel flat-fielding - (back-up option)

4.2 The Localization

The localization strategy is based on the following technical assumptions (AD1):

- The long-term set-up repeatability is within 3 pixels in the direction perpendicular to dispersion. This assumption is not critical and may be relaxed if necessary.
- The relative mechanical stability of the slit mechanical assembly (the random differential displacement between sub-slits) is better than 0.1 microns in the slit plane, therefore the relative displacements of spectra, each one with respect to others, on the detector are negligible for the purpose of the localization. This assumption is not critical for the purpose of the localization, but **is critical** for the simultaneous wavelength calibration. **There is no back-up solution**.
- The analytical model with adjustable parameters linking the position on the entrance slit to the position on the CCD detector is available. Such model gives both the position and the width of the projected image of the entrance fibre on the CCD. Provisionally we adopt the model proposed by V. Cayatte (RD6). As a **back-up solution the polynomial model could be used**.
- The analytical model with adjustable parameters describing the PSF in the direction perpendicular to the dispersion (y) as a function of the position in the dispersion direction is available for each spectrum. As a **back-up solution the Gaussian, another ad-hoc or tabulated profile** could be used.

The localization starts from a set of preliminary parameters which are already close to the final values. The simple optical model (calibration constants) is used for the preliminary solution. The localization operates on a complete set or a subset of the data. It is used to derive a complete localization solution as well as for the adjustment of the current solution using 5 simultaneous wavelength calibration spectra only. The localization proceeds according the following steps:

- Using the current localization, compute centroid and optionally the width for each spectral bin of each spectrum.
- Select bins which will be used for the fit on the basis of the signal level, number of valid pixels in the bin and estimated error of the bin localization.
- Fit parameters of the localization correction model on selected bins
- Update the current localization

4.3 The Extraction

The extraction process rests on the same technical assumption as the localization. It makes use of the parameters of the localization with no further correction allowed in that respect. The extraction proceeds as follows:

- By default, the pixels are weighted proportionally to the inverse of their estimated variances. Other weighting schemes are possible.

- For each spectral bin an analytical model is fitted on the multiple spectra profile (including all spectra). Optionally, if the cross-talk between spectra can or is wanted to be neglected, this step could be done separately for each spectrum by Horne's method (RD7).
- The intensities, the sigmas and the local backgrounds of the extracted spectra are set to the amplitudes, errors and background terms of the fitted model.

Remarks on the extraction process:

- Flagged points:

  During the extraction, the flagged points are not used for the fit. Since in the extraction step all data are used in a global model, this is the optimal opportunity to replace flagged points in the input image by the modeled points and have at disposal a *cleaned* input image. This is normally not necessary since all further processing will be done on the extracted spectra, but it could be useful if the user wants to iterate the extraction process.

  Note that some flagged points could remain on the extracted spectra. The values of the flagged points of the extracted spectra are set to the sum of all points of the considered bin. The extraction will produce also the *image* of the number of points used for each extracted bin.

- Error estimate:

  The standard error σ of the flux at each pixel of the extracted spectrum is estimated during the extraction process and saved. This provides an independent estimate of the overall error for each spectral element which could be compared to the shot-noise limit. This will be used at the level of the *Ancillary Data Analysis Software* (ADAS).

- Flat-fielding:

  The relative pixel sensitivity is assumed to be uniform enough to make possible the localization and extraction processes before any flat-fielding. In this case, the high spatial frequency component of the pixel sensitivity acting as a small additional noise in these processes is neglected. Nevertheless, the possibility to correct for the high spatial frequency component of the detector non-uniformity *before* the extraction is offered as an option, for the off-line reduction. In general, both high and low spatial frequencies will be corrected for *after* extraction using the NFF extracted frames.

4.4 The Wavelength calibration and rebinning

The wavelength calibration rests on similar technical assumption as the localization. It uses an analytical model and fits physical parameters of the model to the position of localized spectral lines. It operates in two steps which are not necessarily carried out both:

- Get Wavelength Solution which associates a wavelength vector to each raw extracted spectrum in a way that gives the wavelength for the x-th bin of the n-th spectrum.

  The analytical expression for the Wavelength Solution is used. An initial solution, if no valid current solution exists, is obtained from the optical design, the localization and optionally (technical set-up) the residual initial offset by interactively pointing a few lines. A list of lines is associated to each source of the calibration spectra (thorium/argon/neon source, sky). Each line of each spectrum is accurately localized through line profile fit (provisionally quadratic) on the upper part of the data. The sigma clipping is implemented to eliminate spurious detections (according to preliminary tests on real spectra, sigma clipping at 10 sigma level is efficient enough). The analytical Wavelength Solution is fitted either to the positions or to the residuals of the detected lines.

- Carry out rebinning of extracted spectra so that all rebinned spectra are equally spaced in λ and cover exactly the same spectral domain.

  The final Wavelength Solution is checked by computing the cross-correlation between the wavelength-calibrated spectrum and the standard calibration spectrum built from the list of lines

The same Wavelength calibration is used during the *separate calibration through all fibres* as well for *simultaneous calibration on 5 fibres.*

4.5 Global FF correction - correction for the blaze function, individual fibre transmission and CCD spectral response

In this step, the extracted spectra are corrected for spectrograph and detector response, as well as for individual fibre transmission. The high-frequency correction is applied at this step. Both scientific spectra and calibration Narrow Flat-Fields have been pre-processed and extracted in a same way and therefore both are sets of uncalibrated extracted spectra.

The Global FF correction is based on the following technical assumptions:

- Since the fibre illumination during the calibration by the FP is time-multiplexed, the variation of the relative spectrum of the FF calibration lamp (located at FP) should be negligible during the calibration run and the optics of the calibration system must insure that the illumination of all fibres is within 0.1% (AD4) or at least varies from fibre to fibre in a reproducible way.
- The transmission of the Nasmyth focal plane optical system (including the fibres, and the Nasmyth corrector) varies in a simple and reproducible way with the position within the focal plane. Provisionally we assume that 2-D radial/axial polynomial could be fitted to the measured variations.

The resulting flux-calibrated spectra should be considered as delivered in *units* of the reference spectrum of the calibration lamp (calibration constant). There is **no back-up solution if the reference spectrum at the fibre entrance is poorly known**. The variation of the calibration lamps will be monitored using the total light collected on the detector per unit of time and per spectrum. The comparison of the collected flux when using the wavelength calibration lamp and FF calibration lamp should reveal the ageing of any of them (uncorrelated part), while the correlated part of variation will indicate the modification of the fibre/spectrograph transmission.

This is the first step of the BLDRS where the fibre transmission is used. It is worth mentioning that each slit fibre assembly will have its own calibration data and with respect to the wavelength dependence, these will also be different for each observing mode. The remaining sources of the variations of the transmission with the position at the Nasmyth focal plane (corrector transmission, vignetting) will be fibre-assembly independent and could be disentangled from the fibre transmission.

4.6 The Sky modeling and subtraction

The sky subtraction is carried out on the extracted flux-calibrated spectra. Several critical issues must be mentioned:

- The accuracy of the subtraction is fully dependent on the previous step and more specifically on the accuracy of the measurement and stability of the fibre transmission. There is **no back-up solution other than the repeated fibre transmission measurements** and monitoring of all possible sources of error.
- In order to be able to directly subtract extracted spectra, the PSFs must be identical. The critical step is therefore to *model* the sky spectrum with the PSF of the target spectra. This is done by fitting the sky model using the independent variables related to the PSF variation. The choice of independent variables is still open. It could be the λ or more simply a spectral bin number together with the width of the PSF. As a back-up solution the convolution to the worst-case PSF offers the possibilities at the expense of resolution.
- It is assumed that the sky could vary across the field only in intensity, not in spectral distribution. The back-up solution is the piecewise (in such a case the spectral interval is divided in several segments) normalization.

The Sky modeling is made as follows:

- Normalize the sky spectra in order to be able to compare the line-profiles
- Build the model of the normalized sky as function of the wavelength and detector *y-position* or the PSF width
- Optionally compute field-position dependent model of the total sky intensity
- Multiply local sky model by the local total sky intensity (from the model or from the data)
- Subtract sky model

4.7 The Flux spectrophotometric calibration (F. Royer)

The flux calibration is carried out on wavelength calibrated and flat-fielded IFU spectra and is done either using spectrophotometric standard star(s) measured simultaneously, or using a model of transmission (atmosphere and telescope).

The first step is to correct fluxes from the loss of light due to the PSF. To do so the actual loss factor for each object is obtained and the flux-and-wavelength calibrated spectra of each IFU are multiplied by the loss factor. Then the PSF corrected fluxes should be calibrated.

- Use of spectrophotometric standard stars: the spectra of the standard star(s) are processed in the same way as the other spectra, flat-field corrected and wavelength calibrated. All the observed spectra are normalized for airmass equal to unity. The ratio of absolute fluxes over the observed ones is computed for one (or possibly several) spectrophotometric standard star(s). Then all observed fluxes are multiplied by this ratio, which should essentially reflect the atmospheric and telescope transmission.
- Use of transmission model: the calibration lamp then plays the role of the standard star. Since the absolute spectrum of the calibration source is already taken into account during the flat-field correction step, only the atmosphere and telescope transmission models should be included so that observed fluxes are divided by these quantities. Atmospheric transmission, updated by the extinction monitor (TBC ESO), as well as the telescope transmission as a function of wavelength, are needed to achieve this step.

5 The External Components / Software interfaces (G. Simond)

Depending on the running mode: maintenance, test and off-line re-reduction the BLDRS will rely on different external components.

- Image display utility such as ESO Real Time Display (RTD) [RD6],
- Graphics 2-D/3-D plotting utility/library,
- ESO MIDAS data reduction package,
- VLT On-line Archive & Science Archive,
- Software library to read/write work on FITS formatted files,
- Software library to manipulate datacubes such as the Eclipse image processing library

This is the currently foreseen list, it could be reduced or increased during the BLDRS design phase.

6 The graphical user interface - GUI (G. Simond)

Within the DFS framework, all BLDRS modules are running off-line, without any user interaction and therefore no GUI at all. However the BLDRS shall have a user interface to be used during development, test and integration phases, as well as for the re-reduction process outside the VLT environment.

The BLDRS-GUI is a top layer over the BLDRS-PC and the following GUI panels are foreseen:

- BLDRS-PC for configuration/status and control the execution of reduction steps,
- BLDRS-GFN configuration and display panel,
- BLDRS-FGR configuration and display panel,
- ESO Real Time Display (RTD)

7 The graphics and image displays - GID (A. Blecha)

A MOS instrument such as FLAMES/GIRAFFE produces intrinsically 3-D data with two spatial dimensions X,Y in the focal plane and wavelength dimension in λ direction. The spatial sampling is continuous in ARGUS mode, discontinuous in MEDUSA mode, and mixture of both in IFU mode, while the λ sampling is continuous in any case.

In order to display the data in a synthetic and comprehensible way the true 3-D graphic is not appropriate for at least two reasons: the discontinuous nature of the spatial sampling, and a very large difference in number of samples between the λ direction with ~ 4000 pixels and spatial directions with $\sqrt{130-320} \sim 10-20$ pixels each. Most of information is present in 1-D vectors of individual spectra and it is assumed that the standard graphical tools will be used to visualize them. On the other hand, and even in the case of MEDUSA mode (130 spectra) such a high number of curves is definitely too much and could not be displayed directly. The tools that give the possibility to *navigate* in 3-D space and link the spatial and spectral informations are therefore necessary. This is the task of the Graphical Field Navigator (GFN) described below.

7.1 Graphical Field Navigator - GFN

The GFN fulfills several functions:

1. Field Display (GFN-FD)

   The GFN displays the field covered by 3-D data. It uses the data contained in FITS KWs. In its simplest form, the objects are displayed as symbols located in the $[\alpha, \delta]$ plane with a reduced number of physical properties displayed as graphical attributes. The magnitudes-scaled circles in MEDUSA mode and oriented

rectangles representing the covered field in IFU or ARGUS modes could be an example. Display of all parameters available in the FITS KWs is possible via the configurable graphical attributes (not necessarily simultaneously). It includes also the display of parameters related to the observation such as the fibre attribution and the spectra numbers.

2. Background image (GFN-BI)

   If a field reference image to scale is supplied, it could be optionally displayed in the background with GFN-FD superimposed

3. Observed image display (GFN-OID)

   Observed intensities integrated over selected spectral interval could be displayed instead or superimposed to GFN-FD.

4. Object selection/tagging (GFN-SEL)

   GFN enables the graphical object selection individually, by region and globally (all). Both selection and exclusion are implemented and several selections could be activated simultaneously.

5. Interface to Graphical Functions on selected objects (GFN-INT)

   GFN provides the communication and control of the standard graphical tools. The control direction could be inverted in a way that the GFN-OID is controlled from a specific graphical window. In that case the user *navigates* in the spectral domain while integrated image or otherwise projected 3D image is being updated continuously unlike the *standard* mode when navigation/selection takes place in the spatial domain and the spectra graphics are updated accordingly.

   - Enables the selection of graphical functions and sets necessary parameters (displays the selected spectra in specified spectral range and associated functions)
   - Initializes and configures the graphical tools
   - Transmits the requests to graphical tools and controls their execution

7.2 Full Graphical Reporter - FGR

Though it is (even with the progress of the computer machinery) out of question to display all the information concerning a single exposure in a synthetic way on one or several computer displays, the color printing just starts to open the possibility to produce such a display on a single page or few pages. This is the reason to propose a parameterized tool that could adjust to the possibility of the available printing machine and offers the possibility to concentrate the information in way to quickly perceive the data content.

- Standard parameterized graphical pages layout

  A selection of standard graphical pages, each of them also parameterized, is proposed. The full default and general parameters (such as *all, auto-scale, ...*) are available.

  - Calibrated Spectra, applicable to MEDUSA and IFU mode.
    The printed equivalent of the GFN-FD display with all options including GFN-BI and GFN-OID and graphically linked to selected spectra in the selected spectral domain.
  - Calibrated Images, applicable to ARGUS and IFU
    The sequence of images reconstructed in the specified spectral domains and graphically linked to the specific or average spectrum (also displayed)
  - Data Reduction Reports (FGR-DRR)
    The 2-D images with localization mask and badPixelMask superimposed together with graphics of various corrections relevant to the specific Recipe. At least one FGR-DRR/Standard Recipe is foreseen.

- Fast preview

  Rapid coarse preview on the computer display is available in order to adjust parameters in case of interactive use.

- Possibility to split report in several pages when repeating the *navigation* part on all pages.

8 The process control PC (G. Simond)

All BLDRS reduction steps are implemented as stand-alone SW modules in such a way that it is possible to run them independently, chained in a step-by-step execution or fully automatically, using scripts or procedures.

At the telescope, data reduction is running in automated mode within the VLT DataFlow System. BLDRS modules are linked together using Reduction Rules, Reduction Recipes and Recipes Signatures handled and scheduled by the DFS pipelines as defined in VLT DFS Specifications [AD8,AD9].

The interactive modes will be used during the maintenance, integration, tests or off-line data reduction processing. In these modes each BLDRS module can be run separately with data, configuration and parameters specified as arguments on the command line or as/in input files.

The process control should be able to:

- Set or modify general control parameters (step-by-step), source of control parameters ...
- Start, pause or abort the execution of the reduction process
- Set input parameters of any reduction step
- Modify configuration parameters of any reduction step
- Get and display each reduction step output data and parameters

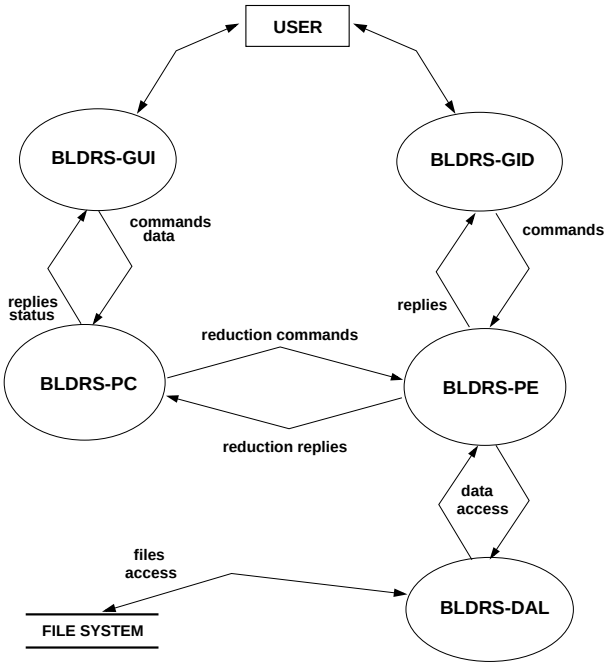

Figure 1: General control of the BLDRS

9 Standard pipeline recipes (G. Simond, F. Royer)

9.1 Introduction

The VLT Control System (VCS) produces FITS files which are manipulated within the DFS in the form of frames. The DFS reduction pipeline applies Reduction Rules on the incoming frames in order to create Reduction Blocks which identifies the set of frames to be processed and a reference to the applicable Reduction Recipe. Reduction Recipes are the data analysis procedures executed by the DFS to carry on the data reduction steps [RD3,RD4]. BLDRS Reduction Rules and Reduction Recipes will be used

- in the Reduction Pipeline, for standard instrument modes and standard data reduction steps;
- in the Calibration Pipeline, for calibration of all type of data received from the instrument;
- in the Quality Control Pipeline, for the assessment of the technical/physical quality and for trend analysis.

The following sections describes basic reductions recipes available in the BLDRS. These Recipes will be updated/expanded and new Recipes will be added during the BLDRS design and test phases.
The access to the input data, control parameters, the graphical functions/displays and the error handling are not included in this description.
In the same way, data types as well as parameters types, range and values are not given here, they will be fully described in the recipe's Data Definition Files and Signature File.

9.2 Recipes description

GirRecipeName

Description:
GirRecipeName – recipe description. In order to differentiate BLDRS recipes from BLDRS functions, recipe names always begin with the prefix `"Gir"` whereas function names are prefixed with `"gi"`.

Control parameters:
Recipes control parameters.

Input (basic) data: Specifies input data required by this recipe, for example:
`inFrame, <darkFrame>`
This recipe requires a mandatory input frame `inFrame` and an optional dark frame `darkFrame`.
Input data can be derived data from another recipe, this is the case for calibration data, or raw data coming from the instrument.

Output (derived) data: Specifies output data produced by this recipe, for example:
`outFrame, outValue`
Derived data are always produced unless an error occurs during the recipe execution.

Processing:
The pseudo-code describing the recipe processing to compute derived data from basic data in a matlab-like syntax. For example to describe a recipe named *GirRecipeName*, we used the following notation:

```
recipe [outFrame,outValue] = GirRecipeName(inFrame,paramsList)
{
   # Anything after a '#' is treated as a comment (ignored!)

   # This recipe just returns inFrame and 1.
   LongNameFrame = giFunctionWithAVeryLongName(inFrame,longNameFrame,veryLongNameFrame, ...
                                               evenLongerNameFrame,longNameArgument);
```

```
   outValue = 1;
}
```

Where "recipe" is the keyword specifying the beginning of a recipe definition, "[xxxx]" represents the list notation and "[outFrame,outValue]" stands for a list containing the two elements: outFrame and outValue.
Here outFrame and outValue are the derived data (a frame and a value) output from *GirRecipeName* and inFrame and paramsList are the elements of the list of needed input data (Basic data) and control parameters.
The trailing dots "..." on a line stand for "continued on next line".

9.3 Calibration Recipes

GirCcdDarkCal

Description:
GirCcdDarkCal – create CCD (normalized) dark frame

Control parameters:
```
biasParameters = [bias-areas-params,bias-sigma-clip-params]
darkParameters = [dark-sigma-clip-params]
```

Input (basic) data:
```
inFrameList    = [inFrame0,inFrame1,...]       : List of one or more dark frames
badPixMaskList = [badPixMask0,badPixMask1,...]: List of corresponding bad pixel masks
```

Output (derived) data:
```
stdDarkFrame
```

Processing:
```
recipe [stdDarkFrame] = GirCcdDarkCal(inFrameList,badPixMaskList)
{
     for (i = 0; i < length(inFrameList); i += 1) do {
        # Remove bias and trim overscan areas
        [darkFrameList[i],biasValue,biasSigma] = giGetRemoveBias(inFrameList[i], ...
                                                                 badPixMaskList[i], ...
                                                                 biasParameters);
    }

    # Remove cosmic ray hits on darkFrameList, build an average dark frame
    [stdDarkFrame,stdDarkSigmaFrame,crhBadPixMask,crhCount] =
                          giDetectCosmicM(darkFrameList,badPixMaskList,crdetParameters);

}
```

GirNarrowFfCal

Description:
GirNarrowFfCal – creates CCD (normalized) narrow flat-field frame, computes localization data

Control parameters:
```
biasParameters  = [bias-areas-params,bias-sigma-clip-params]
crdetParameters = [crh-sigma-clip-params]
```

```
estimParameters = [badPixTreshold,maxEstimRadius,estimMetric]
sclParameters   = [interspectra-areas-params,scattered-light-model]
locParameters   = [psfParams,loc-sigma-clip-params]
extParameters   = [psfParams,ext-sigma-clip-params]
```

Input (basic) data:

`inFrameList = [inFrame0,inFrame1,...]` : List of flat-field frames
`badPixMaskList = [badPixMask0,badPixMask1,...]`: List of corresponding bad pixel masks
`stdLocY,stdLocWy` (initial) localization data derived from *`GirNarrowFfCal`*
`stdDarkFrame`: standard dark frame derived from *`GirCcdDarkCal`*
`stdPhffFrame`: standard photometric FF frame derived from CCD laboratory calibration

Output (derived) data:

`nffFrame`: (normalized) narrow flat-field frame
`nffBadPixMask`: updated bad pixels mask
`extNarrowFF`: extracted narrow flat-field spectra
`nffLocData`: localization data

Processing:

```
recipe [nffFrame,nffBadPixMask,extNarrowFF,nffLocData] = ...
                              GirNarrowFfCal(inFrameList,badPixMaskList, ...
                                             stdLocY,stdLocWy,stdDarkFrame,stdPhffFrame)
{
    for (i = 0; i < length(inFrameList); i += 1) do {
        # Remove bias and trim overscan areas
        [biasFrame,biasValue,biasSigma] = giGetRemoveBias(inFrameList[i], ...
                                                  badPixMaskList[i],biasParameters);

        # Remove dark
        prepFrameList[i] = giSubtractDark(biasFrame,badPixMaskList[i],stdDarkFrame);
    }

    # Remove cosmic ray hits on inFrameList, build an average flat-field frame
    [avgFrame,sigmaFrame,crhBadPixMask,crhCount] =
                          giDetectCosmicM(prepFrameList,badPixMaskList,crdetParameters);

    # Optionally, remove scattered light
    sclmFrame = giAdjustSL(avgFrame,crhBadPixMask,stdPhffFrame,sclParameters);
    nffFrame  = giArithm(avgFrame,sclmFrame, '-');

    # Compute localization data
    [nffLocY,nffLocWy,nffLocCenter,nffLocWidth] = ...
                               giLocalSpectra(nffFrame,stdLocY,stdLocWy,locParameters);
    nffLocData = [nffLocY,nffLocWy,nffLocCenter,nffLocWidth];

    # Extract spectra
    [extNarrowFF,extNarrowFfError,extNarrowFfPixels,extBackground,nffBadPixMask] = ...
                                        giExtractSpectra(nffFrame,crhBadPixMask, ...
                                                         nffLocY,nffLocWy, ...
                                                         stdDarkFrame,extParameters);
}
```

GirWlengthCal

Description:

GirWlengthCal – Wavelength calibration (SEWC)

Control parameters:

```
biasParameters  = [bias-areas-params,bias-sigma-clip-params]
crdetParameters = [crh-sigma-clip-params]
estimParameters = [badPixTreshold,maxEstimRadius,estimMetric]
locParameters   = [psfParams,loc-sigma-clip-params]
extParameters   = [psfParams,ext-sigma-clip-params]
wlenParameters  = [wlen-sigma-clip-params]
```

Input (basic) data:

`inFrameList` = `[inFrame0,inFrame1,...]` : List of one or more ThArNe frames
`badPixMaskList` = `[badPixMask0,badPixMask1,...]`: List of corresponding bad pixel masks
`stdLocY,stdLocWy`: standard localization data derived from *GirNarrowFfCal*
`stdWlenSolution`: (initial) wavelength solution derived from *GirWlengthCal*
`stdLineTable` derived from physical constants and laboratory calibration
`stdDarkFrame`: standard dark frame derived from *GirCcdDarkCal*

Output (derived) data:

`wlenSolution`: adjusted analytical model of wavelength solution
`extSpectraLambda`: table of wavelength λ for each $x - bin$ of each spectra

Processing:

```
recipe [wlenSolution,extSpectraLambda] = ...
                        GirWlengthCal(inFrameList,badPixMaskList,stdLocY,stdLocWy, ...
                                      stdWlenSolution,stdLineTable,stdDarkFrame)
{
    for (i = 0; i < length(inFrameList); i += 1) do {
        # Remove bias and trim overscan areas
        [biasFrame,biasValue,biasSigma] = giGetRemoveBias(inFrameList[i], ...
                                                  badPixMaskList[i],biasParameters);

        # Remove dark
        prepFrameList[i] = giSubtractDark(biasFrame,badPixMaskList[i],stdDarkFrame);
    }

    # Remove cosmic ray hits
    #  and compute average of inFrameList
    [avgFrame,sigmaFrame,crhBadPixMask,crhCount] =
                        giDetectCosmicM(prepFrameList,badPixMaskList,crdetParameters);

    # Optionally, compute localization data and compare with standard localization
    [locY,locWy,locCenter,locWidth] = ...
                             giLocalSpectra(avgFrame,stdLocY,stdLocWy,locParameters);

    # Extract spectra
    [extSpectra,extSpectraError,extSpectraPixels,extBackground,extBadPixMask] = ...
                                        giExtractSpectra(avgFrame,crhBadPixMask, ...
                                                         stdLocY,stdLocWy, ...
                                                         stdDarkFrame,extParameters);
```

```
    # Compute wavelength solution
    [wlenSolution,extSpectraLambda] = giGetWaveSolution(extSpectra,stdWlenSolution, ...
                                                        stdLineTable,wlenParameters);
}
```

9.4 Observation Reduction Recipes

GirSingleExpObs

Description:

GirSingleExpObs – single observation reduction

Control parameters:

```
biasParameters  = [bias-areas-params,bias-sigma-clip-params]
estimParameters = [badPixTreshold,maxEstimRadius,estimMetric]
crdetParameters = [crh-sigma-clip-params]
sclParameters   = [interspectra-areas-params,scattered-light-model]
locParameters   = [psfParams,loc-sigma-clip-params]
extParameters   = [psfParams,ext-sigma-clip-params]
wlenParameters  = [wlen-sigma-clip-params]
rebinParameters = [deltaLambda,rebinMethod]
nskyParameters  = [lambdaMin,lambdaMax]
mskyParameters  = [sky-sigma-clip-params]
sskyParameters  = [skySubOption]
```

Input (basic) data:

`inFrame`: raw frame
`badPixMask`: bad pixel mask
`stdLocY,stdLocWy`: standard localization data derived from *GirNarrowFfCal*
`stdExtNarrowFF`: standard narrow FF extracted spectra derived from *GirNarrowFfCal*
`stdWlenSolution`: standard wavelength solution derived from *GirWlengthCal*
`focalXY`: X and Y co-ordinates of fibres on the focal plane `stdLineTable` derived from labo calibration
`stdPhffFrame`: standard photometric FF frame derived from CCD labo calibration
`stdNffFrame`: standard narrow FF frame derived from *GirNarrowFfCal*
`stdDarkFrame`: standard dark frame derived from *GirCcdDarkCal*

Output (derived) data:

```
skyFlxWlenCalSpectra
skyFlxWlenCalSpectraError
```

Processing:

```
recipe [skyFlxWlenCalSpectra] = GirSingleExpObs(inFrame,badPixMask,stdLocY,stdLocWy, ...
                                                stdExtNarrowFF,stdWlenSolution,focalXY, ...
                                                stdLineTable,stdPhffFrame,stdDarkFrame)
{
    # Remove bias and trim overscan areas
    [biasFrame,biasValue,biasSigma] = giGetRemoveBias(inFrame,biasParameters);

    # Remove dark
    drkcFrame = giSubtractDark(biasFrame,badPixMask,stdDarkFrame);

    # Remove cosmic ray hits
      # Optionally the 3 following steps could be skipped
        #1# Compute estimate Frame
        [estFrame,estSigma,estNpixels] = ...
                    giEstimatePix(drkcFrame,badPixMask,stdLocY,stdLocWy,estimParameters);

        #2# Compute CRH mask
        [crhBadPixMask,crhCount] = ...
```

```
                giDetectCosmicS(drkcFrame,estFrame,estSigma,badPixMask,crdetParameters);

        #3# prepFrame = CRH cleaned input frame
        prepFrame = giReplaceFlagPix(drkcFrame,crhBadPixMask,estFrame);

    # Optionally, remove scattered light
    sclmFrame = giAdjustSL(prepFrame,crhBadPixMask,stdPhffFrame,sclParameters);
    sobsFrame = giArithm(prepFrame,sclmFrame, '-');

    # Compute localization data and compare with standard localization
    [locY,locWy,locCenter,locWidth] = ...
                            giLocalSpectra(sobsFrame,stdLocY,stdLocWy,locParameters);

    # Extract spectra
    [extSpectra,extSpectraError,extSpectraPixels,extBadPixMask,extBackground] = ...
                               giExtractSpectra(sobsFrame,crhBadPixMask,stdLocY, ...
                                                stdLocWy,stdDarkFrame,extParameters);

    # Compute wavelength solution (SIWC)
    [wlenSolution,extSpectraLambda] = giGetWaveSolution(extSpectra,stdWlenSolution, ...
                                                        stdLineTable,wlenParameters);

    # Flatfield correction in X-space
    [fltCalSpectra,fltCalSpectraError] = giFlatSpectra(extSpectra,stdExtNarrowFF);

    # fltCalSpectra contains fltCalObjects and fltCalSkies
    [fltCalObjects,fltCalSkies] = fltCalSpectra;

    # Rebin in wavelength space
    [fltWlenCalSpectra,fltWlenCalSpectraError] = ...
                                  giRebinSpectra(fltCalSpectra, ...
                                                 extSpectraError,extSpectraPixels, ...
                                                 extSpectraLambda,rebinParameters);

    # Flat and wavelength calibrated object and skies spectra
    [fltWlenCalObjects,fltWlenCalSkies] = fltWlenCalSpectra;

    # Optionally multiply by FF calibration lamp
    flxWlenCalSpectra  = giArithm(fltWlenCalSpectra,fltWlenCalLamp, '*');

    # Sky subtraction in wavelength space
      # Optionally the 3 following steps could be skipped
        #1# Normalize sky spectra
        [intSky,normSky] = giNormalizeSky(flxWlenCalSkies,normSkyList,nskyParameters);

        #2# modelize sky spectra
        [sInt,sNorm,skyModel] = giModelSky(intSky,normSky,focalXY,mskyParameters);

        #3# subtract sky model from each spectrum
        skyFlxWlenCalSpectra = giSubtractSky(flxWlenCalSpectra,skyModel);

    [skyFlxWlenCalObjects,skyFlxWlenCalSkies] = skyFlxWlenCalSpectra;
}
```

9.5 Miscellaneous Reduction Recipes

GirSubBiasDark

Description:

GirSubBiasDark – Bias and Dark correction, trims overscan areas

Control Parameters:

`biasParameters = [bias-areas-params,bias-sigma-clip-params]`

Input (basic) data:

`rawFrame`: raw frame
`badPixMask`: bad pixel mask
`stdDarkFrame`: standard dark frame derived from *GirCcdDarkCal*

Output (derived) data:

`outFrame`: dark, bias and overscan corrected frame
`biasValue,biasSigma`

Processing:

```
recipe [outFrame,biasValue,biasSigma] = ...
                GirSubBiasDark(rawFrame,badPixMask,stdDarkFrame)
{
    # Remove bias and trim overscan areas
    [biasFrame,biasValue,biasSigma] = giGetRemoveBias(rawFrame,biasParameters);

    # Remove dark
    outFrame = giSubtractDark(biasFrame,badPixMask,stdDarkFrame);
}
```

GirRemoveCrh

Description:

GirRemoveCrh – Detects and removes Cosmic Ray Hits from one or a list of frames.
The input can be one image or a series of images. This will be used mainly for flat-field images.

Control parameters:

```
estimParameters = [badPixTreshold,maxEstimRadius,estimMetric]
crdetParameters = [crh-sigma-clip-params]
```

Input (basic) data:

`inFrameList = [inFrame0,inFrame1,...]` : List of one or more input frames
`badPixMaskList = [badPixMask0,badPixMask1,...]`: List of corresponding bad pixel masks
`stdLocY,stdLocWy`: standard localization data derived from *GirNarrowFfCal*

Output (derived) data:

`outFrame`: CRH-cleaned frame
`crhBadPixMask`: CRH-updated badPixMask
`crhCount`: CRH attributed count

Processing:

```
recipe [outFrame,crhBadPixMask] = ...
                                GirRemoveCrh(inFrameList,badPixMaskList,stdLocY,stdLocWy)
{
    if (length(inFrameList) == 1) then {
```

```
        # Compute estimate Frame
        [estFrame,estSigma,estNpixels] = ...
                    giEstimatePix(inFrame,badPixMask,stdLocY,stdLocWy,estimParameters);
        # Compute CRH mask
        [crhBadPixMask,crhCount] = ...
                  giDetectCosmicS(inFrame,estFrame,estSigma,badPixMask,crdetParameters);

        # outFrame = CRH cleaned input frame
        outFrame = giReplaceFlagPix(inFrame,crhBadPixMask,estFrame);
    } else {
        # Compute average frame and CRH mask from inFrameList
        [avgFrame,sigmaFrame,crhBadPixMask,crhCount] =
                           giDetectCosmicM(inFrameList,badPixMaskList,crdetParameters);

        # The following steps are needed only if CRH pixels remains in crhBadPixMask
# otherwise outFrame is the avgFrame returned by giDetectCosmicM unchanged
        # Compute estimate frame for average frame
        [estFrame,estSigma,estNpixels] = ...
                  giEstimatePix(avgFrame,crhBadPixMask,stdLocY,stdLocWy,estimParameters);

        # outFrame = CRH cleaned average frame
        outFrame = giReplaceFlagPix(avgFrame,crhBadPixMask,estFrame);
    }
}
```

GirRemoveScatteredLight

Description:

GirRemoveScatteredLight – Removes scattered light.

Control parameters:

`sclParameters = [interspectra-areas-params,scattered-light-model]`

Input (basic) data:

`inFrame`: preprocessed (dark, bias and overscan)
`badPixMask`: bad pixel mask
`stdPhffFrame`: standard photometric FF frame derived from CCD labo calibration

Output (derived) data:

`outFrame`: SL-cleaned frame
`sclmFrame`: computed scattered light frame

Processing:

```
recipe [outFrame,sclmFrame] = GirRemoveScatteredLight(inFrame,badPixMask,stdPhffFrame)
{
    sclmFrame = giAdjustSL(inFrame,BadPixMask,stdPhffFrame,sclParameters);
    outFrame = giArithm(inFrame,sclmFrame, '-');
}
```

GirExtractSpectra

Description:

GirExtractSpectra – Spectra extraction

Control parameters:

`locParameters = [psfParams,loc-sigma-clip-params]`
`extParameters = [psfParams,ext-sigma-clip-params]`

Input (basic) data:

`inFrame`: preprocessed (dark, bias and overscan)
`badPixMask`: bad pixel mask
`stdLocY,stdLocWy`: standard localization data derived from *GirNarrowFfCal*
`stdNffFrame`: standard narrow FF frame derived from *GirNarrowFfCal*
`stdDarkFrame`: standard dark frame derived from *GirCcdDarkCal*

Output (derived) data:

`extLocData`: localization data
`extSpectraData`: extracted spectra, spectra errors, spectra npoints
`extBadPixMask`: updated bad pixels mask
`extBackground`: background models

Processing:

```
recipe [extLocData,extSpectraData,extBadPixMask,extBackground] = ...
                      GirExtractSpectra(inFrame,badPixMask,stdLocY,stdLocWy,stdNffFrame)
{
    # Compute localization data
    [locY,locWy,locCenter,locWidth] = ...
                             giLocalSpectra(stdNffFrame,stdLocY,stdLocWy,locParameters);

    # Extract spectra
    [extSpectra,extSpectraError,extSpectraPixels,extBackground,extBadPixMask] = ...
                                giExtractSpectra(inFrame,badPixMask, ...
                                                 locY,locWy,stdDarkFrame,extParameters);

    extLocData     = [locY,locWy,locCenter,locWidth];
    extSpectraData = [extSpectra,extSpectraError,extSpectraPixels];
}
```

GirWlenSolution

Description:

GirWlenSolution – Compute wavelength solution

Control parameters:

`wlenParameters = [wlen-sigma-clip-params]`

Input (basic) data:

`extSpectra`: extracted spectra derived from *GirExtractSpectra*
`stdWlenSolution`: standard wavelength solution derived from *GirWlengthCal*
`stdLineTable` derived from physical constants and labo calibration

Output (derived) data:

`wlenSolution`
`extSpectraLambda`: $\lambda - x$ table

Processing:

```
recipe [wlenSolution,extSpectraLambda] = ...
                           GirWlenSolution(extSpectra,stdWlenSolution,stdLineTable)
{
```

```
    [wlenSolution,extSpectraLambda] = giGetWaveSolution(extSpectra, ...
                                          stdWlenSolution,stdLineTable,wlenParameters);
}
```

GirFlatCalSpectra

Description:
GirFlatCalSpectra – blaze function, fibre transmission and CCD response correction

Control parameters:
`fltCalParameters = TBD`

Input (basic) data:
`extSpectra`: extracted spectra derived from *GirExtractSpectra*
`stdExtNarrowFF`: standard narrow FF extracted spectra derived from *GirNarrowFfCal*

Output (derived) data:
`fltCalSpectra`: flat-fielded spectra (skies and objects) modulo reference spectrum
`fltCalSpectraError`: standard error of the flat-fielded spectra

Processing:

```
recipe [fltCalSpectra,fltCalSpectraError] = ...
                      GirFlatCalSpectra(extSpectra,stdExtNarrowFF)
{
   [fltCalSpectra,fltCalSpectraError]  = giFlatSpectra(extSpectra,stdExtNarrowFF);

   [fltCalObjects,fltCalSkies] = fltCalSpectra;
}
```

GirWlenCalSpectra

Description:
GirWlenCalSpectra – rebinning of extracted spectra in wavelength space

Control parameters:
`rebinParameters = [deltaLambda,rebinMethod]`

Input (basic) data:
`inSpectraData`: extracted spectra data derived from *GirExtractSpectra*
`extSpectraLambda`: $\lambda - x$ table derived from *GirWlenSolution*

Output (derived) data:
`wlenCalSpectra`: spectra (skies and objects) rebinned in wavelength space
`wlenCalSpectraError`: standard error of the rebinned spectra

Processing:

```
recipe [wlenCalSpectra,wlenCalSpectraError] = ...
         GirWlenCalSpectra(inSpectraData,extSpectraLambda)
{
   [inSpectra,inSpectraError,inSpectraPixels] = inSpectraData;

   [wlenCalSpectra,wlenCalSpectraError] = ...
                           giRebinSpectra(inSpectra,inSpectraError,inSpectraPixels, ...
                                          extSpectraLambda,rebinParameters);

   [wlenCalObjects,wlenCalSkies] = wlenCalSpectra;
}
```

GirSkyCalSpectra

Description:

GirSkyCalSpectra – sky subtraction

Control parameters:

```
nskyParameters  = [lambdaMin,lambdaMax,normSkyList]
mskyParameters  = [psfParams,sky-sigma-clip-params]
```

Input (basic) data:

`fltWlenCalSpectra`: extracted spectra data derived from *GirWlenCalSpectra*
`stdLocY,stdLocWy`: standard localization data derived from *GirNarrowFfCal*
`focalXY`: X and Y co-ordinates of fibres on the focal plane

Output (derived) data:

`skyCalSpectra`

Processing:

```
recipe [skyCalSpectra] = ...
                  GirSkyCalSpectra(fltWlenCalSpectra,stdLocY,stdLocWy,focalXY)
{
    [fltWlenCalObjects,fltWlenCalSkies] = fltWlenCalSpectra;

    # Normalize sky spectra
    [intSky,normSky] = giNormalizeSky(fltWlenCalSkies,normSkyList,nskyParameters);

    # modelize sky spectra
    [sInt,sNorm,skyModel] = giModelSky(intSky,normSky,stdLocWy,focalXY,mskyParameters);

    # subtract sky spectra
    skyCalSpectra = giSubtractSky(fltWlenCalSpectra,skyModel);

    # sky calibrated objects and skies spectra
    [skyCalObjects,skyCalSkies] = skyCalSpectra;
}
```

GirFlxCalSpectra

Description:

GirFlxCalSpectra – Flux spectrophotometric calibration

Control parameters:

`TBD`

Input (basic) data:

`stdflxCalSpectra`: standard stars extracted spectra data derived from *GirWlenCalSpectra*
`fltWlenCalSpectra`: extracted spectra data derived from *GirWlenCalSpectra*
`absflxCalSpectra`: absolute fluxes for the corresponding standard stars

Output (derived) data:

`flxCalSpectra`: Nifu flux and wavelength calibrated object spectra
`lossFactor`: loss factor for each object

Processing:

```
recipe [flxCalSpectra] = ...
                  GirFlxCalSpectra(fltWlenCalSpectra,stdflxCalSpectra,absflxCalSpectra)
{
   [nStdSp,fltWlenStdSp] = stdflxCalSpectra;
   [flxCalSpectra,lossFactor] = giFluxCal(fltWlenCalSpectra,nStdSp,fltWlenStdSp);

}
```

10 High-level functional specifications (F. Royer, P. North)

The detailed specifications below give only the essential information necessary to describe the specific requirements. Several aspects were not taken into account on purpose and left for the design phase. The parameters common to all functions such as error flags and messages, quality parameters, process control parameters or data accesses are not shown. The parameters communicated as a FITS KW in images are, a part of a few exceptions, not listed. The various aspects of the normalization of the calibration data (to a reference exposure time), conversion to the physical units are not included. We do not show the use of *masks* in formulae except where essential for the understanding of the specific step.

10.1 Basic operations on images

Name: **giArithm** Section: All nu: 10

Input/Output data

I/O	Name	Type - Description
Inputs	image1(N1,...,N3)	0-3D - image
	image2(N1,...,N3)	0-3D - image
	cBadPixelMask1(N1,...,N3)	0-3D - current bad pixels mask for image1
	cBadPixelMask2(N1,...,N3)	0-3D - current bad pixels mask for image2
Outputs	resultImage(N1,...,N3)	0-3D - image resulting of the operation
	cBadPixelMask(N1,...,N3)	0-3D - current bad pixels mask for resultImage

- Inputs: Optionally, the bad pixels masks are required during operations preceding the extraction process
- Supported modes: Medusa, Argus, Ifu
- Used low-level functions: Image arithmetics

Description

Compute basic pixel-to-pixel operations between images.

Complete description

Basic arithmetical and logical operations are computed over 0-3D images and/or masks. Mixed dimensions are accepted under condition of the size match. In such a case the low-dimensioned operand is used repeatedly. When needed, the bad pixels mask of the resulting image is taken as the union of both input bad pixels masks.

Quality assessment

The quality of the resulting image will mainly depend on the ratio of the number of pixels with NaN values computed in the operation over the total number of pixels.

10.2 Get bias value over bias test area

Name: **giGetRemoveBias** Section: Preprocessing nu: 20

Input/Output data

I/O	Name	Type - Description
Inputs	biasSigmaClip(3)	1D - sigma-clipping control parameters (biasSigmaMultiple, biasSigmaNloops, biasNpointsMinfrac)
	biasLimits(4,biasNareas)	2D - limits of biasNareas Bias rectangular areas (overscan)
	rawImage(Nx,Ny0)	2D - raw image with overscan lane
Outputs	bSubImage(Nx,Ny)	2D - bias subtracted and overscan trimmed image
	biasValue	N - average bias
	biasSigma	N - sigma of the average bias
	biasNpointsAccepted	N - total number of accepted points
	biasNpointsTotal	N - total number of points in all areas

- Supported modes: Medusa, Argus, Ifu
- Used low-level functions: Image statistics
- Externals required: Expected read-out noise (Detector SW - supplied via FITS KW)
- FITS KW: expectedReadoutRms

Description

The bias and the read-out noise are obtained as the average value and sigma over the *overscan* areas.

Complete description

At least 2 areas (left and right overscan) are defined, but the overscan could be divided in more sub-areas. For each area, the average bias biasValue and the biasSigma are computed separately. Only pixels whose $value$ follows the relation $\mid value - biasValue \mid < biasSigmaMultiple \times biasSigma$ are accepted. The sigma-clipping is repeated for each area till no new points are rejected or the biasSigmaNloops iteration is reached or when too many points are rejected ($\frac{biasNpointsAccepted}{biasNpointsTotal} < biasNpointsMinfrac$). After the subtraction the bias lanes are trimmed out.

Quality assessment

The comparison between results from different areas, the proportion of rejected points and the comparison of the measured sigma with the nominal read-out noise are used.

10.3 Subtract dark

Name: **giSubtractDark** Section: Preprocessing nu: 30

Input/Output data

I/O	Name	Type - Description
Inputs	bSubImage(Nx,Ny)	2D - bias subtracted and overscan trimmed image
	darkImage(Nx,Ny)	2D - master image of dark in electron/second units
	cBadPixelMask(Nx,Ny)	2D - current mask of bad pixels
Outputs	dSubImage(Nx,Ny)	2D - dark and bias-subtracted image

- Supported modes: Medusa, Argus, Ifu

- Used low-level functions: Image arithmetics
- FITS KW: exposureTime

Description

The scaled (with exposure time and CCD gain factors) Master dark is subtracted from the current image.

Complete description

The decision whether the Dark will be subtracted or not (Dark negligible) will be made in the Reduction Recipe

Quality assessment

None

10.4 Estimate pixels values in an image

Name: **giEstimatePix** Section: Preprocessing nu: 40

Input/Output data

I/O	Name	Type - Description
Inputs	image(Nx,Ny)	2D - image at any stage of the reduction
	cBadPixelMask(Nx,Ny)	2D - current mask of bad pixels
	badPixThresh	N - threshold of neighboring bad pixels implying the expansion of the averaging area
	maxAreaRadius	N - maximum radius (in pixels) of the averaging area
	locY(Nx,Ns)	2D - center lines of standard localization
	locWy(Nx,Ns)	2D - width parameter of standard localization
	areaMetric(2)	1D - metric indicating how the active region is extended in x-, in y-direction
Outputs	estimateImage(Nx,Ny)	2D - estimated values of each the pixel
	estimateSigmaImage(Nx,Ny)	2D - estimated of the standard error of each pixel
	estimateNpixels(Nx,Ny)	2D - number of points used to compute average

- Supported modes: Medusa, Argus, Ifu
- Used low-level functions: Image statistics

Description

Apply a median estimation to the input frame over the closest neighboring good pixels.

Complete description

Each pixel of estimateImage (respectively estimateSigmaImage) is the median (respectively the standard error) of the pixel values (excluding bad pixels) of the input Image computed over the averaging area.

$$estimateImage(x,y) = \overline{image(x,y)}_{\text{averaging area}}$$

The averaging area remains inside the localization mask and the area used is *shaped* according to the areaMetrics. The default averaging area with [1,1] metrics has a 1-pixel-radius, so that the 2 (respectively 4) closest pixels are used in the estimate depending whether the estimate is done in one or two dimensions. If the proportion of

bad pixels in this averaging area is greater than badPixThresh, the radius of the averaging area is incremented in a way to be defined (1-pixel increment, or following the series [1,$\sqrt{2}$,2,$\sqrt{5}$,2$\sqrt{2}$,3,$\sqrt{10}$ ...]) and the process is iterated until maxAreaRadius is reached or the ratio of bad pixel is low enough (<badPixThresh).
$estimateNpixels(x, y)$ contains the corresponding radius of the averaging area used to compute the median and the standard error.

Quality assessment

The estimateNpixels(Nx,Ny) gives the map of the number of pixel used for the average. Few global parameters will be made out of this image.

10.5 Detect Cosmic ray hits in a single image

Name: **giDetectCosmicS** Section: Preprocessing nu: 50

Input/Output data

I/O	Name	Type - Description
Inputs	dSubImage(Nx,Ny)	2D - dark and bias subtracted image
	estimateImage(Nx,Ny)	2D - estimated values of each the pixel
	estimateSigmaImage(Nx,Ny)	2D - estimated of the standard error of each pixel
	crhsSigmaMultiple	N - multiple of sigmas used for sigma-clipping
	cBadPixelMask(Nx,Ny)	2D - current mask of bad pixels
Outputs	crhsCount(Nx,Ny)	2D - number of photons attributed to CRH
	cBadPixelMask(Nx,Ny)	2D - current(upgraded) mask of bad pixels

- Supported modes: Medusa, Argus, Ifu
- Used low-level functions: Image statistics

Description

Detection of CRH is made on the image using pixel values and their local estimates. The current bad pixels mask is updated.

Complete description

The CRHs will be localized using the data of the current image and based on local estimates of pixel value and standard error. The excess count $crhsCount(x, y) = dSubImage(x, y) - estimateImage(x, y)$ is attributed to CRHs if $crhsCount(x, y) > crhsSigmaMultiple \times estimateSigmaImage(x, y)$. Otherwise the crhsCount(x,y) is set to 0. The BadPixelMask is updated

$$cBadPixelMask(x, y) = cBadPixelMask(x, y) \cup (crhsCount(x, y) > 0)$$

Quality assessment

Number of CRH photons/sec compared to CRH alert value

10.6 Detect Cosmic ray hits in several images

Name: **giDetectCosmicM** Section: Preprocessing nu: 60

Input/Output data

I/O	Name	Type - Description
Inputs	ldSubImage(Nimage,Nx,Ny)	1D - list of Nimage dark and bias subtracted images
	crhmSigmaClip(3)	1D - sigma-clipping control parameters (crhmSigmaMultiple, crhmSigmaNloops, crhmNpointsMinfrac)
	lcBadPixelMask(Nimage,Nx,Ny)	3D - list of Nimage current masks of bad pixels
Outputs	crhBadPixelMask(Nx,Ny)	2D - bad pixels of the average image due to CRH
	averageImage(Nx,Ny)	2D - number of photons attributed to CRH (total on all images)
	errorImage(Nx,Ny)	2D - standard error image of the Nframes input frames
	crhmCount(Nx,Ny)	

- Supported modes: Medusa, Argus, Ifu
- Used low-level functions: Image statistics

Description

Detection of CRH is obtained via the sigma-clipped average of all input images.

Complete description

The detection of CRH is made simultaneously with the computation of the average and the standard error images. It is assumed that images are taken in similar conditions and with same exposure times.

The average image $averageImage(x, y)$ is computed by attributing to each pixel the average, sigma-clipped value. The standard error sigmaImage(x,y) is set to the error after the rejection of sigma-clipped points.

Then each individual images $subImage(x, y)$ is scaled to average intensity of $averageImage(x, y)$ using the valid points only. The scaling is sigma-clipped and the excess count is determined as $individualCrhmCount(x, y) = scaledSubImage(x, y) - averageImage(x, y)$

If $individualCrhmCount(x, y) < sigmaImage(x, y) \times CRHMsigmaClipMultiple$, then the $individualCrhmCount(x, y$ is set to 0. Otherwise the local excess is considered as a CRH hit.

The CRH bad pixels mask is the product of all detected CRH hits :

$$crhBadPixelMask(x, y) = \prod_{n_{\mathrm{image}}} ((individualCrhmCount(x, y) > 0)$$

Optionally an image of the total count of CRH photons is given as sum of all $individualCrhmCount(x, y)$.

The output average image $averageImage(x, y)$ with $crhBadPixelMask(x, y)$ represents the most complete cleaned image available under specified conditions of the sigma-clipping.

Quality assessment

The total of irrecoverable CRH is computed from $crhBadPixelMask(Nx, Ny)$ (expected to be zero with standard calibration pipeline), the errorImage(Nx,Ny) and the residuals between scaled images compared to the expected photon noise are used.

10.7 Replace the flagged pixels

Name: **giReplaceFlagPix** Section: Preprocessing nu: 70

Input/Output data

I/O	Name	Type - Description
Inputs	image(Nx,Ny)	2D - image at any stage of the reduction
	estimateImage(Nx,Ny)	
	cBadPixelMask(Nx,Ny)	2D - current mask of bad pixels
Outputs	repPixImage(Nx,Ny)	2D - image with replaced flagged pixels

- Supported modes: Medusa, Argus, Ifu
- Used low-level functions: Image arithmetics

Description

Flagged pixels are replaced by the estimated value. This operation is implemented at various stages of the data reduction.

Complete description

Image and estimateImage are merged according to cBadPixelMask :

$$\begin{aligned} repPixImage(x,y) &= iImage(x,y) \quad \text{if} \quad cBadPixelMask(x,y) = 0 \\ &= estimateImage(x,y) \quad \text{otherwise} \end{aligned}$$

Quality assessment

None

10.8 Adjust the scattered light

Name: **giAdjustSL** Section: Preprocessing nu: 80

Input/Output data

I/O	Name	Type - Description
Inputs	dSubImage(Nx,Ny)	2D - dark and bias subtracted image
	isLimits(4,isNareas)	2D - limits of the inter-spectra areas
	slModel	Identifier of the model of scattered light used (TBD)
	phffImage(Nx,Ny)	2D - photometric Flat field image
	cBadPixelMask(Nx,Ny)	2D - current mask of bad pixels
Outputs	slImage	2D - image of scattered light

- Supported modes: Medusa
- Used low-level functions: Image arithmetics
- Externals required: Expected read-out noise (Detector SW - supplied via FITS KW)

Description

The model of scattered light is adjusted on the inter spectra region.

Complete description

The model of scattered light (form TBC $\sum_{i,j} c_{ij} x^i y^j$ or $\frac{\sum_{i,j} a_{ij} x^i y^j}{\sum_{m,n} b_{mn} x^m y^n}$) is fitted on the inter-spectra region dSubImage(Xis,Yis) (optionally corrected by PHFFimage(Xis,Yis)), where Xis and Yis stand for coordinates of inter-spectra pixels . Then the full image of the scattered light SLImage is constructed from the model with, optionally, the transform back to the unflatfielded image.

Quality assessment

The RMS scatter of the residuals around the SLM compared to the expected noise ($\sqrt{RON^2 + N_{e^-}}$)

10.9 Localization of spectra

Name: **giLocalSpectra** Section: Extraction nu: 90

Input/Output data

I/O	Name	Type - Description
Inputs	prepImage(Nx,Ny)	2D - preprocessed image
	locY(Nx,Ns)	2D - center lines of standard localization
	locWy(Nx,Ns)	2D - width parameter of standard localization
	psfParams(Nx,Ny,n)	3D - n-coefficient describing the shape of the PSF (excluding the width) for any position [x,y]
	locSigmaClip(3)	1D - parameters for sigma clipping
Outputs	locYCur(Nx,Ns)	2D - center lines of current localization
	locWyCur(Nx,Ns)	2D - width parameter of current localization
	locCenter(Nx,Ns)	2D - found points of localization
	locWidth(Nx,Ns)	2D - found width parameter

- Inputs: For standard localisation the Narrow Flat Field image is used.
- Supported modes: Medusa, Argus, Ifu
- Externals required: Optical parameters f_{coll}, G

Description

The localization determines the centroid and the width of each spectral bin of each spectrum using the preliminary localization. A new current localization is defined through the LSQ fit of the optical model to positions and widths found.

Complete description

For a better use, the input image should be CRH and defect cleaned, but the localization is very robust and the sigma clipping at two steps of processing will reject any damaged point on a well exposed image.

Though the standard application of the localization function will be with NFF images, the wavelength calibration frames or even the scientific images could be used in order to verify the current localization.

The localization proceeds as follows:

- Using the initial localization $locY(x,n)$, get the y position of the centroid $locCenter(x,n)$ in each x element of each spectrum n. Only the significant points (sigma-clipping) are retained here.

- Get the y-half-width $locWidth(x,n)$ in each x element of each spectrum n (according to $locWy(x,n)$).

  This step is optional and may be skipped. If so, the standard half-width obtained from the PSF will be used for the extraction.

  Technically, the centroid and half-width adjustment could be done in a single step. If no precise specifications are available, a standard PSF, yet TBD, will be fitted.

- In order to insure the continuity of the localization solution, the parameterized optical model $locYCur(x,n)$ is fitted (with sigma clipping) to the $locCenter(x,n)$ (see Wavelength Calibration for used parameterization). In a similar way the width model $locWyCur(x,n)$ is adjusted on $locWidth(x,n)$ data.

- The reduction recipes has the control over the decision whether this new localization becomes the new initial localization. The localization can be used iteratively.

Quality assessment

The rms scatter of the residuals of the $locCenter(x,n)$ positions around the model and the systematic shift in localization given by the difference between input and output models are used. Since the localization is determined independently for each spectrum, several *global* parameters could monitor long-term stability of the solution.

10.10 Spectra Extraction

Name: **giExtractSpectra** Section: Extraction

Input/Output data

I/O	Name	Type - Description
Inputs	prepImage(Nx,Ny)	2D - preprocessed image
	darkImage(Nx,Ny)	2D - master image of dark in electron/second units
	locYCur(Nx,Ns)	2D - center lines of current localization
	locWyCur(Nx,Ns)	2D - width parameter of current localization
	psfParams(n,Nx,Ny)	3D n-coefficient describing the shape of the PSF (excluding the width) for any position [x,y]
	cBadPixelMask(Nx,Ny)	2D - current mask of bad pixels
	extSigmaClip(3)	1D - parameters of the sigma clipping
Outputs	extractSp(Nx,Ns)	2D - extracted spectra
	extractSpError(Nx,Ns)	2D - extracted spectra errors
	extractSpNpixels(Nx,Ns)	2D - pixel count of extracted spectra
	cBadPixelMask(Nx,Ns)	2D - current mask of extracted bad pixels
	extractBack(Nx,K)	2D - K polynomial coefficient for Nx independent backgrounds models

---

- Supported modes: Medusa, Argus, Ifu
- Externals required: Expected read-out noise (Detector SW - supplied via FITS KW)
- FITS KW: expectedReadoutRms

Description

The extraction implements both classical Horne's (RD7) and PSF fitting methods. The local error and the residual background are also computed.

Complete description

Each spectral bin (a slice in y-direction of the preprocessed image) is described as a linear combination of the PSF multiplying the spectral elements $extractSp(x_c, n)$ which we want to extract (note that the y_c and the width of the PSF are supplied through the $locY(x_c, n)$ and $locWy(x_c, n)$ parameters not shown explicitly in the formula below):

$$modelextractSp(x_c, y) = bg(x_c, y) + \sum_{n=1}^{N} extractSp(x_c, n) \times PSF(psfParams(x_c, n, y - y_c)) \quad (1)$$

The extraction is carried out for each x-bin through the LSQ fit of pre-processed data $prepImage(x_c, y)$ to the equation 1. Only the amplitudes extractSp(x_c,n) multiplying the fixed PSF and the global background are free parameters. Each pixel $prepImage(x_c, y)$ is weighted, for the LSQ fit, by the inverse of its variance

$$varPrepImage(x_c, y) = (prepImage(x_c, y) + RON^2 + darkImage(x_c, y))$$

$$weight(x_c, y) = 1/varPrepImage(x_c, y)$$

so that the quantity to be minimized is

$$\chi^2 = \sum_{i=1}^{Ny} weight(x_c, y) \times (prepImage(x_c, y_i) - modelextractSp(x_c, y_i))^2$$

The fitted model is compared to the data and *sigma-clipping* is implemented based on the local sigma expected from the model. This is the optimal way to remove the CRH. A mask of rejected points cBadPixelMask is updated in this process.

Equation 1 is a generalization of Horne's optimal extraction to the case where the spectra are not strictly independent due to the cross-talk. In addition, the background has to be determined simultaneously. We assume here that it is a slowly varying function of y:

$$bg_n(x_c, y) = \sum_k a_k(x_c) y^k$$

so that equation 1 is a linear system of equations.

The extractSpError(x_c,n) is set to the error of the extractSp(x_c,n) estimated from the fit and extractSpNpixels(x_c,n) to the number of pixels used for each extracted point. Note that the extractSpNpixels(x_c,n) plays the role of the bad pixel mask on the extracted spectra.

This is a general method well adapted to any situation (including the case where the inter-spectra contamination is severe). If the crosstalk is very low (expected case in MEDUSA mode) or the adjacent spectra are of similar intensity (close to uniform object in ARGUS/IFU mode) a pure Horne's method could be used. Note, that in that case the SL should be removed previously:

$$extractSp(x_c, n) = \frac{\sum_i W_H(x_c, y_i) \times prepImage(x_c, y_i)/PSF(psfParams(x_c, n, y_i - y_c))}{\sum_i W_H(x_c, y_i)}$$

and

$$W_H(x_c, y_i) = varPrepImage(x_c, y_i)/PSF(psfParams(x_c, n, y_i - y_c))^2$$

:

Quality assessment

The standard deviation extractSpError is compared to the expected SNR. Another element is the extracted background; if the SL was subtracted prior to the extraction, the background should be compatible with the expected local noise.

10.11 Global FF correction - correction for the blaze function, individual fibre transmission and CCD spectral response

Name: **giFlatSpectra** Section: Flux calibration nu: 110

Input/Output data

I/O	Name	Type - Description
Inputs	extractSp(Nx,Ns)	2D - extracted spectra
	extractError(Nx,Ns)	2D - error of extracted spectra
	extractSp(Nx,Ns)	2D - extracted spectra
Outputs	ffSp(Nx,Ns)	2D - flat-fielded spectra modulated by reference spectrum
	ffSpError(Nx,Ns)	2D - errors of Flat-field corrected extracted spectra

- Supported modes: Medusa, Argus, Ifu

Description

The extracted spectra are corrected for spectrograph and detector response, as well as for individual fibre transmission and *modulated* by spectrum of the FF calibration lamp.

Complete description

Both scientific spectra and calibration Narrow Flat-Fields have been pre-processed in the same way, therefore both are sets of uncalibrated spectra. The optical path is *almost* the same (with the exception of the focal ratio of the illuminating beam) and we expect that even the high-frequency corrections (Pixel-to-pixel CCD response and fringing) will be accurate enough. The scientific and NFF extracted spectra could be written respectively:

$$extractSp(x,n) = flxSp(x,n) \times tFibre(x,n) \times tSpectro(x,n) \times QE_n(x,n)$$

$$extractNFF(x,n) = flxNff(x) \times tFibre(x,n) \times tSpectro(x,n) \times QE_n(x,n)$$

The $flxSp(x,n)$ are fluxes at the fiber entrance of the spectral element that fall on the CCD pixel x of the $n-th$ spectrum. The flxNff(x) is considered as unique (independent of n), which assumes the uniform illumination of all fibres during the calibration exposure.

The reference flux flxNff(x) comes from the laboratory calibrations of the lamp and is sampled in the λ-space.

The ratio of the two equations above gives the Flat-field corrected input spectra $ffSp(x,n)$ modulated by the spectrum of the calibration source. The $ffSpError(x,n)$is obtained in a similar way.

$$ffSp(x,n) = \frac{flxSp(x,n)}{flxNff(x)} = \frac{extractSp(x,n)}{extractNff(x,n)}$$

Note that the fibre transmissions $tFibre(x,n)$ and spectrograph response $tSpectro(x,n)$ are canceled out at this step and will not therefore be used explicitly anywhere in the reduction software. Also the *normalization* to the $\lambda - sampled$ reference spectrum flxNff will not be removed here and left to the wavelength re-sampled spectra (if required).

Quality assessment

There is no quality indicator

10.12 Wavelength Solution

Name: **giGetWaveSolution** Section: Wavelength calibration nu: 120

Input/Output data

I/O	Name	Type - Description
Inputs	extractSp(Nx,Ns)	2D - extracted spectra
	initWaveSolution(Ns,NparWv)	2D - parameters of the analytical model of the wavelength solution for all spectra
	lineTable(Nlines,Nmodes)	2D - table of lines wavelength of the calibration spectrum (all spectral domains and modes)
	wlCalSigmaClip(3)	1D - parameters of sigma-clipping
Outputs	waveSolution(Ns,NparWv)	2D - adjusted analytical model of the wavelength solution
	spLambda(Nx,Ns)	2D - lambda for each x-bin of each spectra

- Inputs: Both Flat-fielded or raw extracted spectra may be used.
- Supported modes: Medusa, Argus, Ifu

Description

The wavelength solution is obtained from a separate (all fibres exposed to calibration lamp - SEWC) or simultaneous (5 fibres exposed to calibration lamp - SIWC) exposure.

Complete description

The same processing is used for SEWC and SIWC. The FITS KWs of the input extracted spectra extractSp(Ns,Nx) give the necessary information to distinguish between the twos.

1. An initial solution, if no valid current solution exists, is obtained from the optical design.
2. The analytical model of the dispersion relation is used

   The waveSolution is a set of parameter describing the analytical model representing the wavelength as a function of n and x.

$$\lambda_n(x) = \frac{\sigma}{m\sqrt{xsl_n^2 + ysl_n^2 + f_{\rm coll}^2}} \left(\sin\theta\, f_{\rm coll} + \cos\theta\, xsl_n + \sqrt{f_{\rm coll}^2 + xsl_n^2}\, \frac{(\sin\theta\, f_{\rm coll}\, G - \cos\theta\, x)}{\sqrt{f_{\rm coll}^2\, G^2 + x^2}} \right) + Xoffset_n$$

   where $f_{\rm coll}$ and G are respectively the focal length of the collimator and the magnification factor of the camera. The xsl_n and ysl_n give the reference position of the fibre n on the entrance slit, while σ is the groove spacing, θ is the grating angle and m is the diffraction order; x is the pixel coordinate and $Xoffset_n$ the offset of each spectrum on the CCD.

3. A list of lines (based on the FEROS line list, in the case of instrumental calibration) is associated to each source of the calibration spectra (thorium/argon/neon source, sky). We expect that after preliminary set-up, this list will be *cleaned* in order to avoid unnecessary systematic rejections during the step 4. The number of lines available is currently being determined.
4. A line profile is fitted to each line and the polynomial expansion is adjusted to the differences between the found x positions and those obtained from the current model. The input model is updated. Though the full degree of the polynomial will always be used, the list of fitted coefficients will depend on the calibration spectra used (see below). The sigma clipping is implemented to eliminate a few lines which do not fit the model.
5. The extractSpLambda(x,n) for all spectra is computed using the updated model.
6. Optionally (technical set-up) the control of elimination/iteration will include access to all optical parameters in order to *tune* the process. We expect, that routinely the Wavelength Solution will be adjusted fully automatically.

Quality assessment

The number of rejected lines and the RMS of the LSQ fit will be used.

10.13 Rebinning in wavelength space

Name: **giRebinSpectra** Section: Wavelength calibration nu: 130

Input/Output data

I/O	Name	Type - Description
Inputs	ffSp(Nx,Ns)	2D - flat-fielded spectra modulo reference spectrum
	extractSpLambda(Nx,Ns)	2D - lambda for each x-bin of each spectra
	deltaLambda	N - wavelength step used in the rebinning
	extractSpNpixels(Nx,Ns)	2D - pixel count of extracted spectra
	ffSpError(Nx,Ns)	2D - errors of Flat-field corrected extracted spectra
	rebinMethod	N - flag indicating the interpolation method used in the computation (TBD)
Outputs	wlFfSp(Nlambda,Ns)	2D - rebinned spectra
	wlFfSpError(Nlambda,Ns)	2D - standard error of the rebinned spectra

- Inputs: A default option of deltaLambda best adapted (according to the observational mode) is available
- Supported modes: Medusa, Argus, Ifu

Description

Carry out rebinning of extractSp(x,n) so that the rebinSp(λ,n) is equally spaced in λ.

Complete description

The rebinning is done to obtain the wlFfSp(Nlambda,Ns) spectra. The extracted spectra ffSp(Nx,Ns) and associated information (error, number of points used) as well as the $\lambda - vectors$ SpLambda(Nx,Ns) are used. Since the choice of the best interpolation method may depend on the nature of the data and the scientific objective of the actual observation, several interpolation methods will be implemented (linear, cubic spline, TBD) selected in the Reduction Recipes.

Quality assessment

This is the last step where possible bad pixels are interpolated. Quality rate of the rebinning may be estimated by the quality of the points used in the interpolation via their associated sigma (extractSpError(x,n)) and the number of points used in each of these Nx bins (extractSpNpixels(x,n)).

10.14 Normalize the Sky spectra

Name: **giNormalizeSky** Section: Sky subtraction nu: 140

Input/Output data

I/O	Name	Type - Description
Inputs	FlxWlCalSky(Nlambda,Nsky)	2D - nsky flux and wavelength calibrated sky spectra
	lambdaMin, lambdaMax	N - borders of the wavelength range
	normSkyList(Nsky)	1D - list of spectra to be normalized
Outputs	intSky(Nsky)	1D - integrated light of each $Nsky$ sky spectra
	normSky(Nlambda,Nsky)	2D - nsky normalized sky spectra

- Supported modes: Medusa, Argus, Ifu

Description

The spectra are normalized to the total flux in the specified λ-range and a normalization factor is given.

Complete description

The integrated light is computed for each flux calibrated sky spectrum

$$intSky(n) = \sum_{\lambda=lambdaMin}^{lambdaMax} flxWlCalSky(\lambda, n)$$

If a normSkyList is given, only the spectra identified in the list are normalized (this option avoid the reformating of data when input image with all spectra - both sky and objects - is used). Each corrected sky spectrum is normalized so that its integrated intensity is unity:

$$normSky(n, \lambda) = \frac{flxWlCalSky(n, \lambda)}{intSky(n)}$$

This step will allow to interpolate the sky spectra in the y direction on the CCD detector for each spectral element of the x axis, in order to remove the differential effects of a PSF which varies along the spectra.

Quality assessment

TBD

10.15 Model the Sky spectrum

Name: **giModelSky** Section: Sky subtraction nu: 150

Input/Output data

I/O	Name	Type - Description
Inputs	intSky(Nsky)	1D - integrated light of each $Nsky$ sky spectra
	normSky(Nlambda,Nsky)	2D - nsky normalized sky spectra
	focalXY(2,Ns)	2D - X,Y position in the focal plane as a function of the fibre number
	locWy(Nlambda,Ns)	2D - width parameter of standard localization (rebinned in the λ-space)
	skySigmaClip(3)	1D - sigma clipping parameters
Outputs	skyModel(Nlambda,Ns)	2D - model of sky for all spectra
	sInt(I,J)	2D - coefficients matrix of the [x,y] polynomial expression for the fit of sky intensity
	sNorm(Nlambda,K)	2D - coefficients matrix of the polynomial expression of fit-NormSky(Nlambda,Ns)

- Supported modes: Medusa, Argus, Ifu

Description

The sky spectrum is modelled over the field applying the modified Viton-Milliard method (RD9).

Complete description

1. Build a model of normalized sky spectra for all spectra (NSKY-SP)

   In order to match the PSF of all spectra to be sky-subtracted, a modified Viton-Milliard method, taking into account a discontinuous PSF variation, is implemented.

   Since the PSF does not vary smoothly on the y axis (local defocusing due to the sub-slit mechanical assembly discontinuities), the Viton-Milliard method cannot be applied without modification.

   In the modified method for each $\lambda - bin$ a 1-d polynomial using the width of the PSF as independent variable is fitted to the normalized sky spectra $normSky(\lambda, n)$

   $$fitNormSky(\lambda, n) = \sum_{k=1}^{K} sNorm(k, \lambda) psfWidth^k(\lambda, n)$$

   where $sNorm(k, \lambda)$ are the fitted coefficients and $psfWidth(\lambda, n)$ is the width of the PSF. Then, the sky to be subtracted from any extracted spectrum n can be estimated on the basis of the PSF width corresponding to this position.

2. Build optionally a model of the spatial distribution of the sky integrated light (SKY-INT).

   If the sky is smoothly varying over the FoV, a model depending on the position in the FoV is fitted to the integrated sky intensities obtained, $intSky(n)$, using by default a 2-d polynomial form:

   $$fitIntSky(X, Y) = \sum_{i,j=0}^{I,J} sInt(i, j) X^i Y^j$$

   where I and J are the polynomial degrees adjusted automatically depending on the number of sky fibres and significance of coefficients, and $sInt(i, j)$ are the fitted coefficients. The model is sigma-clipped according to skySigmaClip parameters.

   This fitted model provides a continuous description of the sky intensities all over the field.

3. Transform the model NSKY-SP back to original intensities (de-normalize)

   The amplitude of the PSF-corrected sky is then transformed back to the original intensity:

   if the sky is smoothly varying over the FoV then

$$skyModel(\lambda, n) = fitNormSky(\lambda, n) \times fitIntSky(focalXY(1, n), focalXY(2, n))$$

   if the sky is strongly heterogeneous over the FoV, sky modeling is skipped and

$$skyModel(\lambda, n) = fitNormSky(\lambda, n) \times intSky(m)$$

   where m is the number of the closest sky-fibre neighboring the n-th.

Quality assessment

The measured sky spectra are compared to the modeled ones. Several numerical indicators (rms and higher spectrum moments) will be proposed.

10.16 Subtract Sky spectrum

Name: **giSubtractSky** Section: Sky subtraction nu: 160

Input/Output data

I/O	Name	Type - Description
Inputs	wlFfSp(Nlambda,Ns)	2D - flux and wavelength calibrated spectra
	skyModel(Nlambda,Ns)	2D - model of sky for all spectra
Outputs	skySubSp(Nlambda,Ns)	2D - flux and wavelength calibrated and sky subtracted spectra

- Supported modes: Medusa, Argus, Ifu

Description

The sky model is subtracted from each object spectrum. This could be carried directly by giArithm

Complete description

The model is subtracted directly:

$$skySubSp(\lambda, n) = FlxWlCalSp(\lambda, n) - skyModel(\lambda, n)$$

Note, that this done on all spectra, including the measured sky spectra.

Quality assessment

An *a posteriori* check of the subtraction of the telluric emission lines component is done by cross-correlating the modeled sky with the object sky: if any correlation peak remains, either in emission or absorption, it means that the amplitude of the subtracted sky is too low or too high respectively. This could only be used in the yellow-red only where emission lines are strong enough. Alternatively, the correlation with solar spectrum template could be used when justified.

10.17 Flux spectrophotometric calibration

Name: **giFluxCal** Section: Flux calibration nu: 170

Input/Output data

I/O	Name	Type - Description
Inputs	wlFfSp(Nlambda,Ns)	2D - Ns FF corrected and wavelength calibrated IFU spectra
	Nstdstar	N - number of observed standard stars
	wlFfStd(Nlambda,Nstdstar*20)	2D - Nstdstar FF corrected and wavelength calibrated standard stars spectra
	wlFlxStd(Nlambda,Nstdstar)	2D - absolute fluxes for the observed standard stars
Outputs	wlFlxSp(Nlambda,Nifu)	2D - Nifu flux and wavelength calibrated object spectra
	loss(Nifu,Nlambda)	2D - loss factor for each object

- Inputs: atmosphere and telescope models could be used in the calibration
- Supported modes: Ifu, Argus
- Used low-level functions: Image arithmetics

Description

The Flux calibration is done either using spectrophotometric standard star(s) measured simultaneously, or using a model of transmission of atmosphere and telescope.

Complete description

For the point sources, fluxes have to be corrected from the PSF, so the flux-and-wavelength calibrated spectra of each IFU should be summed and multiplied by the loss factor computed for each object

$$IFU(\lambda, i) = loss(\lambda, i) \times \sum_{n=N1_i}^{N2_i} wlFfSp(\lambda, n)$$

where $loss(\lambda, i) > 1$ and $[N1_i, N2_i]$ is the fibre number interval corresponding to the i-th IFU-object. Then the PSF corrected fluxes should be calibrated using the source which applies.

- **spectrophotometric standard:**

  The ratio of absolute fluxes $stdFlx(\lambda, n_{\mathrm{stdstar}})$ over the observed ones $IFU(\lambda, n_{\mathrm{stdstar}})$ is computed for the spectrophotometric standard star(s), taking into account the airmass $M(z)$.

  $$RABS(\lambda) = \frac{stdFlx(\lambda, n_{\mathrm{stdstar}})}{IFU(\lambda, n_{\mathrm{stdstar}})} \times \mathrm{e}^{-\kappa(\lambda)M(z)}$$

  where z is the zenital distance and $\kappa(\lambda)$ is the atmospheric absorption coefficient.

  Then all observed fluxes are multiplied by this ratio, which should essentially reflect the atmospheric and telescope transmission, since the correction for the overall fibre and spectrograph sensitivity has already been applied.

  $$wlFlxSp(\lambda, i) = IFU(\lambda, i) \times RABS(\lambda) \times \mathrm{e}^{\kappa(\lambda)M(z_i)}$$

- **no standard available:**

  The calibration lamp then plays the role of the standard star. Since we already took the absolute spectrum of the calibration source into account, only the atmosphere $\mathrm{e}^{-\kappa(\lambda)M(z)}$ and Telescope transmission models $transVlt(\lambda)$ should be included so that

  $$wlFlxSp(\lambda, i) = \frac{IFU(\lambda, i)}{transVlt(\lambda)} \times \mathrm{e}^{\kappa(\lambda)M(z_i)}$$

Quality assessment

TBD

11 Input/output references ordered according the source file

Name	I/O	Function	Description
cBadPixelMask1(N1,...,N3)	I	giArithm	0-3D - current bad pixels mask for image1
cBadPixelMask2(N1,...,N3)	I	giArithm	0-3D - current bad pixels mask for image2
image1(N1,...,N3)	I	giArithm	0-3D - image
image2(N1,...,N3)	I	giArithm	0-3D - image
cBadPixelMask(N1,...,N3)	O	giArithm	0-3D - current bad pixels mask for resultImage
resultImage(N1,...,N3)	O	giArithm	0-3D - image resulting of the operation
biasLimits(4,biasNareas)	I	giGetRemoveBias	2D - limits of biasNareas Bias rectangular areas (overscan)
biasSigmaClip(3)	I	giGetRemoveBias	1D - sigma-clipping control parameters (biasSigmaMultiple, biasSigmaNloops, biasNpointsMinfrac)
rawImage(Nx,Ny0)	I	giGetRemoveBias	2D - raw image with overscan lane
bSubImage(Nx,Ny)	O	giGetRemoveBias	2D - bias subtracted and overscan trimmed image
biasNpointsAccepted	O	giGetRemoveBias	N - total number of accepted points
biasNpointsTotal	O	giGetRemoveBias	N - total number of points in all areas
biasSigma	O	giGetRemoveBias	N - sigma of the average bias
biasValue	O	giGetRemoveBias	N - average bias
bSubImage(Nx,Ny)	I	giSubtractDark	2D - bias subtracted and overscan trimmed image
cBadPixelMask(Nx,Ny)	I	giSubtractDark	2D - current mask of bad pixels
darkImage(Nx,Ny)	I	giSubtractDark	2D - master image of dark in electron/second units
dSubImage(Nx,Ny)	O	giSubtractDark	2D - dark and bias-subtracted image
areaMetric(2)	I	giEstimatePix	1D - metric indicating how the active region is extended in x-, in y-direction
badPixThresh	I	giEstimatePix	N - threshold of neighboring bad pixels implying the expansion of the averaging area
cBadPixelMask(Nx,Ny)	I	giEstimatePix	2D - current mask of bad pixels
image(Nx,Ny)	I	giEstimatePix	2D - image at any stage of the reduction
locWy(Nx,Ns)	I	giEstimatePix	2D - width parameter of standard localization
locY(Nx,Ns)	I	giEstimatePix	2D - center lines of standard localization
maxAreaRadius	I	giEstimatePix	N - maximum radius (in pixels) of the averaging area
estimateImage(Nx,Ny)	O	giEstimatePix	2D - estimated values of each the pixel
estimateNpixels(Nx,Ny)	O	giEstimatePix	2D - number of points used to compute average
estimateSigmaImage(Nx,Ny)	O	giEstimatePix	2D - estimated of the standard error of each pixel
cBadPixelMask(Nx,Ny)	I	giDetectCosmicS	2D - current mask of bad pixels
crhsSigmaMultiple	I	giDetectCosmicS	N - multiple of sigmas used for sigma-clipping
dSubImage(Nx,Ny)	I	giDetectCosmicS	2D - dark and bias subtracted image
estimateImage(Nx,Ny)	I	giDetectCosmicS	2D - estimated values of each the pixel
estimateSigmaImage(Nx,Ny)	I	giDetectCosmicS	2D - estimated of the standard error of each pixel
cBadPixelMask(Nx,Ny)	O	giDetectCosmicS	2D - current(upgraded) mask of bad pixels
crhsCount(Nx,Ny)	O	giDetectCosmicS	2D - number of photons attributed to CRH
crhmSigmaClip(3)	I	giDetectCosmicM	1D - sigma-clipping control parameters (crhmSigmaMultiple, crhmSigmaNloops, crhmNpointsMinfrac)
lcBadPixelMask(Nimage,Nx,Ny)	I	giDetectCosmicM	3D - list of Nimage current masks of bad pixels
ldSubImage(Nimage,Nx,Ny)	I	giDetectCosmicM	1D - list of Nimage dark and bias subtracted images
averageImage(Nx,Ny)	O	giDetectCosmicM	2D - number of photons attributed to CRH (total on all images)

.../...

Name	I/O	Function	Description
crhBadPixelMask(Nx,Ny)	O	giDetectCosmicM	2D - bad pixels of the average image due to CRH
crhmCount(Nx,Ny)	O	giDetectCosmicM	
errorImage(Nx,Ny)	O	giDetectCosmicM	2D - standard error image of the Nframes input frames
cBadPixelMask(Nx,Ny)	I	giReplaceFlagPix	2D - current mask of bad pixels
estimateImage(Nx,Ny)	I	giReplaceFlagPix	
image(Nx,Ny)	I	giReplaceFlagPix	2D - image at any stage of the reduction
repPixImage(Nx,Ny)	O	giReplaceFlagPix	2D - image with replaced flagged pixels
cBadPixelMask(Nx,Ny)	I	giAdjustSL	2D - current mask of bad pixels
dSubImage(Nx,Ny)	I	giAdjustSL	2D - dark and bias subtracted image
isLimits(4,isNareas)	I	giAdjustSL	2D - limits of the inter-spectra areas
phffImage(Nx,Ny)	I	giAdjustSL	2D - photometric Flat field image
slModel	I	giAdjustSL	Identifier of the model of scattered light used (TBD)
slImage	O	giAdjustSL	2D - image of scattered light
locSigmaClip(3)	I	giLocalSpectra	1D - parameters for sigma clipping
locWy(Nx,Ns)	I	giLocalSpectra	2D - width parameter of standard localization
locY(Nx,Ns)	I	giLocalSpectra	2D - center lines of standard localization
prepImage(Nx,Ny)	I	giLocalSpectra	2D - preprocessed image
psfParams(Nx,Ny,n)	I	giLocalSpectra	3D - n-coefficient describing the shape of the PSF (excluding the width) for any position $[x,y]$
locCenter(Nx,Ns)	O	giLocalSpectra	2D - found points of localization
locWidth(Nx,Ns)	O	giLocalSpectra	2D - found width parameter
locWyCur(Nx,Ns)	O	giLocalSpectra	2D - width parameter of current localization
locYCur(Nx,Ns)	O	giLocalSpectra	2D - center lines of current localization
cBadPixelMask(Nx,Ny)	I	giExtractSpectra	2D - current mask of bad pixels
darkImage(Nx,Ny)	I	giExtractSpectra	2D - master image of dark in electron/second units
extSigmaClip(3)	I	giExtractSpectra	1D - parameters of the sigma clipping
locWyCur(Nx,Ns)	I	giExtractSpectra	2D - width parameter of current localization
locYCur(Nx,Ns)	I	giExtractSpectra	2D - center lines of current localization
prepImage(Nx,Ny)	I	giExtractSpectra	2D - preprocessed image
psfParams(n,Nx,Ny)	I	giExtractSpectra	3D n-coefficient describing the shape of the PSF (excluding the width) for any position $[x,y]$
cBadPixelMask(Nx,Ns)	O	giExtractSpectra	2D - current mask of extracted bad pixels
extractBack(Nx,K)	O	giExtractSpectra	2D - K polynomial coefficient for Nx independent backgrounds models
extractSp(Nx,Ns)	O	giExtractSpectra	2D - extracted spectra
extractSpError(Nx,Ns)	O	giExtractSpectra	2D - extracted spectra errors
extractSpNpixels(Nx,Ns)	O	giExtractSpectra	2D - pixel count of extracted spectra
extractError(Nx,Ns)	I	giFlatSpectra	2D - error of extracted spectra
extractSp(Nx,Ns)	I	giFlatSpectra	2D - extracted spectra
extractSp(Nx,Ns)	I	giFlatSpectra	2D - extracted spectra
ffSp(Nx,Ns)	O	giFlatSpectra	2D - flat-fielded spectra modulated by reference spectrum
ffSpError(Nx,Ns)	O	giFlatSpectra	2D - errors of Flat-field corrected extracted spectra
extractSp(Nx,Ns)	I	giGetWaveSolution	2D - extracted spectra
initWaveSolution(Ns,NparWv)	I	giGetWaveSolution	2D - parameters of the analytical model of the wavelength solution for all spectra
lineTable(Nlines,Nmodes)	I	giGetWaveSolution	2D - table of lines wavelength of the calibration spectrum (all spectral domains and modes)
wlCalSigmaClip(3)	I	giGetWaveSolution	1D - parameters of sigma-clipping

.../...

Name	I/O	Function	Description
spLambda(Nx,Ns)	O	giGetWaveSolution	2D - lambda for each x-bin of each spectra
waveSolution(Ns,NparWv)	O	giGetWaveSolution	2D - adjusted analytical model of the wavelength solution
deltaLambda	I	giRebinSpectra	N - wavelength step used in the rebinning
extractSpLambda(Nx,Ns)	I	giRebinSpectra	2D - lambda for each x-bin of each spectra
extractSpNpixels(Nx,Ns)	I	giRebinSpectra	2D - pixel count of extracted spectra
ffSp(Nx,Ns)	I	giRebinSpectra	2D - flat-fielded spectra modulo reference spectrum
ffSpError(Nx,Ns)	I	giRebinSpectra	2D - errors of Flat-field corrected extracted spectra
rebinMethod	I	giRebinSpectra	N - flag indicating the interpolation method used in the computation (TBD)
wlFfSp(Nlambda,Ns)	O	giRebinSpectra	2D - rebinned spectra
wlFfSpError(Nlambda,Ns)	O	giRebinSpectra	2D - standard error of the rebinned spectra
FlxWlCalSky(Nlambda,Nsky)	I	giNormalizeSky	2D - nsky flux and wavelength calibrated sky spectra
lambdaMin, lambdaMax	I	giNormalizeSky	N - borders of the wavelength range
normSkyList(Nsky)	I	giNormalizeSky	1D - list of spectra to be normalized
intSky(Nsky)	O	giNormalizeSky	1D - integrated light of each $Nsky$ sky spectra
normSky(Nlambda,Nsky)	O	giNormalizeSky	2D - nsky normalized sky spectra
focalXY(2,Ns)	I	giModelSky	2D - X,Y position in the focal plane as a function of the fibre number
intSky(Nsky)	I	giModelSky	1D - integrated light of each $Nsky$ sky spectra
locWy(Nlambda,Ns)	I	giModelSky	2D - width parameter of standard localization (rebinned in the λ-space)
normSky(Nlambda,Nsky)	I	giModelSky	2D - nsky normalized sky spectra
skySigmaClip(3)	I	giModelSky	1D - sigma clipping parameters
sInt(I,J)	O	giModelSky	2D - coefficients matrix of the $[x,y]$ polynomial expression for the fit of sky intensity
sNorm(Nlambda,K)	O	giModelSky	2D - coefficients matrix of the polynomial expression of fitNormSky(Nlambda,Ns)
skyModel(Nlambda,Ns)	O	giModelSky	2D - model of sky for all spectra
skyModel(Nlambda,Ns)	I	giSubtractSky	2D - model of sky for all spectra
wlFfSp(Nlambda,Ns)	I	giSubtractSky	2D - flux and wavelength calibrated spectra
skySubSp(Nlambda,Ns)	O	giSubtractSky	2D - flux and wavelength calibrated and sky subtracted spectra
Nstdstar	I	giFluxCal	N - number of observed standard stars
wlFfSp(Nlambda,Ns)	I	giFluxCal	2D - Ns FF corrected and wavelength calibrated IFU spectra
wlFfStd(Nlambda,Nstdstar*20)	I	giFluxCal	2D - Nstdstar FF corrected and wavelength calibrated standard stars spectra
wlFlxStd(Nlambda,Nstdstar)	I	giFluxCal	2D - absolute fluxes for the observed standard stars
loss(Nifu,Nlambda)	O	giFluxCal	2D - loss factor for each object
wlFlxSp(Nlambda,Nifu)	O	giFluxCal	2D - Nifu flux and wavelength calibrated object spectra

12 Input/output references by name and I/O

Name	**I/O**	**Function**	**Description**(as referenced in rdb)
wlFlxStd(Nlambda,Nstdstar)	I	giFluxCal	2D - absolute fluxes for the observed standard stars
wlFlxSp(Nlambda,Nifu)	O	giFluxCal	2D - Nifu flux and wavelength calibrated object spectra
wlFfStd(Nlambda,Nstdstar*20)	I	giFluxCal	2D - Nstdstar FF corrected and wavelength calibrated standard stars spectra
wlFfSpError(Nlambda,Ns)	O	giRebinSpectra	2D - standard error of the rebinned spectra
• wlFfSp(Nlambda,Ns)	O	giRebinSpectra	2D - rebinned spectra
• wlFfSp(Nlambda,Ns)	I	giSubtractSky	2D - flux and wavelength calibrated spectra
• wlFfSp(Nlambda,Ns)	I	giFluxCal	2D - Ns FF corrected and wavelength calibrated IFU spectra
wlCalSigmaClip(3)	I	giGetWaveSolution	1D - parameters of sigma-clipping
waveSolution(Ns,NparWv)	O	giGetWaveSolution	2D - adjusted analytical model of the wavelength solution
spLambda(Nx,Ns)	O	giGetWaveSolution	2D - lambda for each x-bin of each spectra
slModel	I	giAdjustSL	Identifier of the model of scattered light used (TBD)
slImage	O	giAdjustSL	2D - image of scattered light
skySubSp(Nlambda,Ns)	O	giSubtractSky	2D - flux and wavelength calibrated and sky subtracted spectra
skySigmaClip(3)	I	giModelSky	1D - sigma clipping parameters
• skyModel(Nlambda,Ns)	O	giModelSky	2D - model of sky for all spectra
• skyModel(Nlambda,Ns)	I	giSubtractSky	2D - model of sky for all spectra
sNorm(Nlambda,K)	O	giModelSky	2D - coefficients matrix of the polynomial expression of fitNormSky(Nlambda,Ns)
sInt(I,J)	O	giModelSky	2D - coefficients matrix of the [x,y] polynomial expression for the fit of sky intensity
resultImage(N1,...,N3)	O	giArithm	0-3D - image resulting of the operation
repPixImage(Nx,Ny)	O	giReplaceFlagPix	2D - image with replaced flagged pixels
rebinMethod	I	giRebinSpectra	N - flag indicating the interpolation method used in the computation (TBD)
rawImage(Nx,Ny0)	I	giGetRemoveBias	2D - raw image with overscan lane
psfParams(n,Nx,Ny)	I	giExtractSpectra	3D n-coefficient describing the shape of the PSF (excluding the width) for any position [x,y]
psfParams(Nx,Ny,n)	I	giLocalSpectra	3D - n-coefficient describing the shape of the PSF (excluding the width) for any position [x,y]
• prepImage(Nx,Ny)	I	giLocalSpectra	2D - preprocessed image
• prepImage(Nx,Ny)	I	giExtractSpectra	2D - preprocessed image
phffImage(Nx,Ny)	I	giAdjustSL	2D - photometric Flat field image
normSkyList(Nsky)	I	giNormalizeSky	1D - list of spectra to be normalized
• normSky(Nlambda,Nsky)	O	giNormalizeSky	2D - nsky normalized sky spectra
• normSky(Nlambda,Nsky)	I	giModelSky	2D - nsky normalized sky spectra

.../...

Name	**I/O**	**Function**	**Description**(as referenced in rdb)
maxAreaRadius	I	giEstimatePix	N - maximum radius (in pixels) of the averaging area
loss(Nifu,Nlambda)	O	giFluxCal	2D - loss factor for each object
• locYCur(Nx,Ns)	O	giLocalSpectra	2D - center lines of current localization
• locYCur(Nx,Ns)	I	giExtractSpectra	2D - center lines of current localization
• locY(Nx,Ns)	I	giLocalSpectra	2D - center lines of standard localization
• locY(Nx,Ns)	I	giEstimatePix	2D - center lines of standard localization
• locWyCur(Nx,Ns)	O	giLocalSpectra	2D - width parameter of current localization
• locWyCur(Nx,Ns)	I	giExtractSpectra	2D - width parameter of current localization
• locWy(Nx,Ns)	I	giLocalSpectra	2D - width parameter of standard localization
• locWy(Nx,Ns)	I	giEstimatePix	2D - width parameter of standard localization
locWy(Nlambda,Ns)	I	giModelSky	2D - width parameter of standard localization (rebinned in the λ-space)
locWidth(Nx,Ns)	O	giLocalSpectra	2D - found width parameter
locSigmaClip(3)	I	giLocalSpectra	1D - parameters for sigma clipping
locCenter(Nx,Ns)	O	giLocalSpectra	2D - found points of localization
lineTable(Nlines,Nmodes)	I	giGetWaveSolution	2D - table of lines wavelength of the calibration spectrum (all spectral domains and modes)
ldSubImage(Nimage,Nx,Ny)	I	giDetectCosmicM	1D - list of Nimage dark and bias subtracted images
lcBadPixelMask(Nimage,Nx,Ny)	I	giDetectCosmicM	3D - list of Nimage current masks of bad pixels
lambdaMin, lambdaMax	I	giNormalizeSky	N - borders of the wavelength range
isLimits(4,isNareas)	I	giAdjustSL	2D - limits of the inter-spectra areas
• intSky(Nsky)	O	giNormalizeSky	1D - integrated light of each $Nsky$ sky spectra
• intSky(Nsky)	I	giModelSky	1D - integrated light of each $Nsky$ sky spectra
initWaveSolution(Ns,NparWv)	I	giGetWaveSolution	2D - parameters of the analytical model of the wavelength solution for all spectra
image2(N1,...,N3)	I	giArithm	0-3D - image
image1(N1,...,N3)	I	giArithm	0-3D - image
• image(Nx,Ny)	I	giReplaceFlagPix	2D - image at any stage of the reduction
• image(Nx,Ny)	I	giEstimatePix	2D - image at any stage of the reduction
focalXY(2,Ns)	I	giModelSky	2D - X,Y position in the focal plane as a function of the fibre number
• ffSpError(Nx,Ns)	O	giFlatSpectra	2D - errors of Flat-field corrected extracted spectra
• ffSpError(Nx,Ns)	I	giRebinSpectra	2D - errors of Flat-field corrected extracted spectra
• ffSp(Nx,Ns)	O	giFlatSpectra	2D - flat-fielded spectra modulated by reference spectrum

.../...

Name	**I/O**	**Function**	**Description**(as referenced in rdb)
• ffSp(Nx,Ns)	I	giRebinSpectra	2D - flat-fielded spectra modulo reference spectrum
• extractSpNpixels(Nx,Ns)	O	giExtractSpectra	2D - pixel count of extracted spectra
• extractSpNpixels(Nx,Ns)	I	giRebinSpectra	2D - pixel count of extracted spectra
extractSpLambda(Nx,Ns)	I	giRebinSpectra	2D - lambda for each x-bin of each spectra
extractSpError(Nx,Ns)	O	giExtractSpectra	2D - extracted spectra errors
• extractSp(Nx,Ns)	O	giExtractSpectra	2D - extracted spectra
• extractSp(Nx,Ns)	I	giGetWaveSolution	2D - extracted spectra
• extractSp(Nx,Ns)	I	giFlatSpectra	2D - extracted spectra
• extractSp(Nx,Ns)	I	giFlatSpectra	2D - extracted spectra
extractError(Nx,Ns)	I	giFlatSpectra	2D - error of extracted spectra
extractBack(Nx,K)	O	giExtractSpectra	2D - K polynomial coefficient for Nx independent backgrounds models
extSigmaClip(3)	I	giExtractSpectra	1D - parameters of the sigma clipping
• estimateSigmaImage(Nx,Ny)	O	giEstimatePix	2D - estimated of the standard error of each pixel
• estimateSigmaImage(Nx,Ny)	I	giDetectCosmicS	2D - estimated of the standard error of each pixel
estimateNpixels(Nx,Ny)	O	giEstimatePix	2D - number of points used to compute average
• estimateImage(Nx,Ny)	O	giEstimatePix	2D - estimated values of each the pixel
• estimateImage(Nx,Ny)	I	giReplaceFlagPix	
• estimateImage(Nx,Ny)	I	giDetectCosmicS	2D - estimated values of each the pixel
errorImage(Nx,Ny)	O	giDetectCosmicM	2D - standard error image of the Nframes input frames
deltaLambda	I	giRebinSpectra	N - wavelength step used in the rebinning
• darkImage(Nx,Ny)	I	giSubtractDark	2D - master image of dark in electron/second units
• darkImage(Nx,Ny)	I	giExtractSpectra	2D - master image of dark in electron/second units
• dSubImage(Nx,Ny)	O	giSubtractDark	2D - dark and bias-subtracted image
• dSubImage(Nx,Ny)	I	giDetectCosmicS	2D - dark and bias subtracted image
• dSubImage(Nx,Ny)	I	giAdjustSL	2D - dark and bias subtracted image
crhsSigmaMultiple	I	giDetectCosmicS	N - multiple of sigmas used for sigma-clipping
crhsCount(Nx,Ny)	O	giDetectCosmicS	2D - number of photons attributed to CRH
crhmSigmaClip(3)	I	giDetectCosmicM	1D - sigma-clipping control parameters (crhmSigmaMultiple, crhmSigmaNloops, crhmNpointsMinfrac)
crhmCount(Nx,Ny)	O	giDetectCosmicM	
crhBadPixelMask(Nx,Ny)	O	giDetectCosmicM	2D - bad pixels of the average image due to CRH
cBadPixelMask2(N1,...,N3)	I	giArithm	0-3D - current bad pixels mask for image2
cBadPixelMask1(N1,...,N3)	I	giArithm	0-3D - current bad pixels mask for image1
• cBadPixelMask(Nx,Ny)	O	giDetectCosmicS	2D - current(upgraded) mask of bad pixels

.../...

Name	**I/O**	**Function**	**Description**(as referenced in rdb)
• cBadPixelMask(Nx,Ny)	I	giSubtractDark	2D - current mask of bad pixels
• cBadPixelMask(Nx,Ny)	I	giReplaceFlagPix	2D - current mask of bad pixels
• cBadPixelMask(Nx,Ny)	I	giExtractSpectra	2D - current mask of bad pixels
• cBadPixelMask(Nx,Ny)	I	giEstimatePix	2D - current mask of bad pixels
• cBadPixelMask(Nx,Ny)	I	giDetectCosmicS	2D - current mask of bad pixels
• cBadPixelMask(Nx,Ny)	I	giAdjustSL	2D - current mask of bad pixels
cBadPixelMask(Nx,Ns)	O	giExtractSpectra	2D - current mask of extracted bad pixels
cBadPixelMask(N1,...,N3)	O	giArithm	0-3D - current bad pixels mask for resultImage
biasValue	O	giGetRemoveBias	N - average bias
biasSigmaClip(3)	I	giGetRemoveBias	1D - sigma-clipping control parameters (biasSigmaMultiple, biasSigmaNloops, biasNpointsMinfrac)
biasSigma	O	giGetRemoveBias	N - sigma of the average bias
biasNpointsTotal	O	giGetRemoveBias	N - total number of points in all areas
biasNpointsAccepted	O	giGetRemoveBias	N - total number of accepted points
biasLimits(4,biasNareas)	I	giGetRemoveBias	2D - limits of biasNareas Bias rectangular areas (overscan)
badPixThresh	I	giEstimatePix	N - threshold of neighboring bad pixels implying the expansion of the averaging area
• bSubImage(Nx,Ny)	O	giGetRemoveBias	2D - bias subtracted and overscan trimmed image
• bSubImage(Nx,Ny)	I	giSubtractDark	2D - bias subtracted and overscan trimmed image
averageImage(Nx,Ny)	O	giDetectCosmicM	2D - number of photons attributed to CRH (total on all images)
areaMetric(2)	I	giEstimatePix	1D - metric indicating how the active region is extended in x-, in y-direction
Nstdstar	I	giFluxCal	N - number of observed standard stars
FlxWlCalSky(Nlambda,Nsky)	I	giNormalizeSky	2D - nsky flux and wavelength calibrated sky spectra

• means the I/O is referenced more then once.

13 Input/output references by I/O and name

Name	I/O	Function	Description
areaMetric(2)	I	giEstimatePix	1D - metric indicating how the active region is extended in x-, in y-direction
badPixThresh	I	giEstimatePix	N - threshold of neighboring bad pixels implying the expansion of the averaging area
biasLimits(4,biasNareas)	I	giGetRemoveBias	2D - limits of biasNareas Bias rectangular areas (overscan)
biasSigmaClip(3)	I	giGetRemoveBias	1D - sigma-clipping control parameters (biasSigmaMultiple, biasSigmaNloops, biasNpointsMinfrac)
bSubImage(Nx,Ny)	I	giSubtractDark	2D - bias subtracted and overscan trimmed image
cBadPixelMask(Nx,Ny)	I	giAdjustSL	2D - current mask of bad pixels
cBadPixelMask(Nx,Ny)	I	giDetectCosmicS	2D - current mask of bad pixels
cBadPixelMask(Nx,Ny)	I	giEstimatePix	2D - current mask of bad pixels
cBadPixelMask(Nx,Ny)	I	giExtractSpectra	2D - current mask of bad pixels
cBadPixelMask(Nx,Ny)	I	giReplaceFlagPix	2D - current mask of bad pixels
cBadPixelMask(Nx,Ny)	I	giSubtractDark	2D - current mask of bad pixels
cBadPixelMask1(N1,...,N3)	I	giArithm	0-3D - current bad pixels mask for image1
cBadPixelMask2(N1,...,N3)	I	giArithm	0-3D - current bad pixels mask for image2
crhmSigmaClip(3)	I	giDetectCosmicM	1D - sigma-clipping control parameters (crhmSigmaMultiple, crhmSigmaNloops, crhmNpointsMinfrac)
crhsSigmaMultiple	I	giDetectCosmicS	N - multiple of sigmas used for sigma-clipping
darkImage(Nx,Ny)	I	giExtractSpectra	2D - master image of dark in electron/second units
darkImage(Nx,Ny)	I	giSubtractDark	2D - master image of dark in electron/second units
deltaLambda	I	giRebinSpectra	N - wavelength step used in the rebinning
dSubImage(Nx,Ny)	I	giAdjustSL	2D - dark and bias subtracted image
dSubImage(Nx,Ny)	I	giDetectCosmicS	2D - dark and bias subtracted image
estimateImage(Nx,Ny)	I	giDetectCosmicS	2D - estimated values of each the pixel
estimateImage(Nx,Ny)	I	giReplaceFlagPix	
estimateSigmaImage(Nx,Ny)	I	giDetectCosmicS	2D - estimated of the standard error of each pixel
extractError(Nx,Ns)	I	giFlatSpectra	2D - error of extracted spectra
extractSp(Nx,Ns)	I	giFlatSpectra	2D - extracted spectra
extractSp(Nx,Ns)	I	giFlatSpectra	2D - extracted spectra
extractSp(Nx,Ns)	I	giGetWaveSolution	2D - extracted spectra
extractSpLambda(Nx,Ns)	I	giRebinSpectra	2D - lambda for each x-bin of each spectra
extractSpNpixels(Nx,Ns)	I	giRebinSpectra	2D - pixel count of extracted spectra
extSigmaClip(3)	I	giExtractSpectra	1D - parameters of the sigma clipping
ffSp(Nx,Ns)	I	giRebinSpectra	2D - flat-fielded spectra modulo reference spectrum
ffSpError(Nx,Ns)	I	giRebinSpectra	2D - errors of Flat-field corrected extracted spectra
FlxWlCalSky(Nlambda,Nsky)	I	giNormalizeSky	2D - nsky flux and wavelength calibrated sky spectra
focalXY(2,Ns)	I	giModelSky	2D - X,Y position in the focal plane as a function of the fibre number
image(Nx,Ny)	I	giEstimatePix	2D - image at any stage of the reduction
image(Nx,Ny)	I	giReplaceFlagPix	2D - image at any stage of the reduction
image1(N1,...,N3)	I	giArithm	0-3D - image
image2(N1,...,N3)	I	giArithm	0-3D - image
initWaveSolution(Ns,NparWv)	I	giGetWaveSolution	2D - parameters of the analytical model of the wavelength solution for all spectra

.../...

Name	I/O	Function	Description
intSky(Nsky)	I	giModelSky	1D - integrated light of each $Nsky$ sky spectra
isLimits(4,isNareas)	I	giAdjustSL	2D - limits of the inter-spectra areas
lambdaMin, lambdaMax	I	giNormalizeSky	N - borders of the wavelength range
lcBadPixelMask(Nimage,Nx,Ny)	I	giDetectCosmicM	3D - list of Nimage current masks of bad pixels
ldSubImage(Nimage,Nx,Ny)	I	giDetectCosmicM	1D - list of Nimage dark and bias subtracted images
lineTable(Nlines,Nmodes)	I	giGetWaveSolution	2D - table of lines wavelength of the calibration spectrum (all spectral domains and modes)
locSigmaClip(3)	I	giLocalSpectra	1D - parameters for sigma clipping
locWy(Nlambda,Ns)	I	giModelSky	2D - width parameter of standard localization (rebinned in the λ-space)
locWy(Nx,Ns)	I	giEstimatePix	2D - width parameter of standard localization
locWy(Nx,Ns)	I	giLocalSpectra	2D - width parameter of standard localization
locWyCur(Nx,Ns)	I	giExtractSpectra	2D - width parameter of current localization
locY(Nx,Ns)	I	giEstimatePix	2D - center lines of standard localization
locY(Nx,Ns)	I	giLocalSpectra	2D - center lines of standard localization
locYCur(Nx,Ns)	I	giExtractSpectra	2D - center lines of current localization
maxAreaRadius	I	giEstimatePix	N - maximum radius (in pixels) of the averaging area
normSky(Nlambda,Nsky)	I	giModelSky	2D - nsky normalized sky spectra
normSkyList(Nsky)	I	giNormalizeSky	1D - list of spectra to be normalized
Nstdstar	I	giFluxCal	N - number of observed standard stars
phffImage(Nx,Ny)	I	giAdjustSL	2D - photometric Flat field image
prepImage(Nx,Ny)	I	giExtractSpectra	2D - preprocessed image
prepImage(Nx,Ny)	I	giLocalSpectra	2D - preprocessed image
psfParams(n,Nx,Ny)	I	giExtractSpectra	3D n-coefficient describing the shape of the PSF (excluding the width) for any position [x,y]
psfParams(Nx,Ny,n)	I	giLocalSpectra	3D - n-coefficient describing the shape of the PSF (excluding the width) for any position [x,y]
rawImage(Nx,Ny0)	I	giGetRemoveBias	2D - raw image with overscan lane
rebinMethod	I	giRebinSpectra	N - flag indicating the interpolation method used in the computation (TBD)
skyModel(Nlambda,Ns)	I	giSubtractSky	2D - model of sky for all spectra
skySigmaClip(3)	I	giModelSky	1D - sigma clipping parameters
slModel	I	giAdjustSL	Identifier of the model of scattered light used (TBD)
wlCalSigmaClip(3)	I	giGetWaveSolution	1D - parameters of sigma-clipping
wlFfSp(Nlambda,Ns)	I	giFluxCal	2D - Ns FF corrected and wavelength calibrated IFU spectra
wlFfSp(Nlambda,Ns)	I	giSubtractSky	2D - flux and wavelength calibrated spectra
wlFfStd(Nlambda,Nstdstar*20)	I	giFluxCal	2D - Nstdstar FF corrected and wavelength calibrated standard stars spectra
wlFlxStd(Nlambda,Nstdstar)	I	giFluxCal	2D - absolute fluxes for the observed standard stars
averageImage(Nx,Ny)	O	giDetectCosmicM	2D - number of photons attributed to CRH (total on all images)
biasNpointsAccepted	O	giGetRemoveBias	N - total number of accepted points
biasNpointsTotal	O	giGetRemoveBias	N - total number of points in all areas
biasSigma	O	giGetRemoveBias	N - sigma of the average bias
biasValue	O	giGetRemoveBias	N - average bias
bSubImage(Nx,Ny)	O	giGetRemoveBias	2D - bias subtracted and overscan trimmed image
cBadPixelMask(N1,...,N3)	O	giArithm	0-3D - current bad pixels mask for resultImage

.../...

Name	I/O	Function	Description
cBadPixelMask(Nx,Ns)	O	giExtractSpectra	2D - current mask of extracted bad pixels
cBadPixelMask(Nx,Ny)	O	giDetectCosmicS	2D - current(upgraded) mask of bad pixels
crhBadPixelMask(Nx,Ny)	O	giDetectCosmicM	2D - bad pixels of the average image due to CRH
crhmCount(Nx,Ny)	O	giDetectCosmicM	
crhsCount(Nx,Ny)	O	giDetectCosmicS	2D - number of photons attributed to CRH
dSubImage(Nx,Ny)	O	giSubtractDark	2D - dark and bias-subtracted image
errorImage(Nx,Ny)	O	giDetectCosmicM	2D - standard error image of the Nframes input frames
estimateImage(Nx,Ny)	O	giEstimatePix	2D - estimated values of each the pixel
estimateNpixels(Nx,Ny)	O	giEstimatePix	2D - number of points used to compute average
estimateSigmaImage(Nx,Ny)	O	giEstimatePix	2D - estimated of the standard error of each pixel
extractBack(Nx,K)	O	giExtractSpectra	2D - K polynomial coefficient for Nx independent backgrounds models
extractSp(Nx,Ns)	O	giExtractSpectra	2D - extracted spectra
extractSpError(Nx,Ns)	O	giExtractSpectra	2D - extracted spectra errors
extractSpNpixels(Nx,Ns)	O	giExtractSpectra	2D - pixel count of extracted spectra
ffSp(Nx,Ns)	O	giFlatSpectra	2D - flat-fielded spectra modulated by reference spectrum
ffSpError(Nx,Ns)	O	giFlatSpectra	2D - errors of Flat-field corrected extracted spectra
intSky(Nsky)	O	giNormalizeSky	1D - integrated light of each $Nsky$ sky spectra
locCenter(Nx,Ns)	O	giLocalSpectra	2D - found points of localization
locWidth(Nx,Ns)	O	giLocalSpectra	2D - found width parameter
locWyCur(Nx,Ns)	O	giLocalSpectra	2D - width parameter of current localization
locYCur(Nx,Ns)	O	giLocalSpectra	2D - center lines of current localization
loss(Nifu,Nlambda)	O	giFluxCal	2D - loss factor for each object
normSky(Nlambda,Nsky)	O	giNormalizeSky	2D - nsky normalized sky spectra
repPixImage(Nx,Ny)	O	giReplaceFlagPix	2D - image with replaced flagged pixels
resultImage(N1,...,N3)	O	giArithm	0-3D - image resulting of the operation
sInt(I,J)	O	giModelSky	2D - coefficients matrix of the [x,y] polynomial expression for the fit of sky intensity
skyModel(Nlambda,Ns)	O	giModelSky	2D - model of sky for all spectra
skySubSp(Nlambda,Ns)	O	giSubtractSky	2D - flux and wavelength calibrated and sky subtracted spectra
slImage	O	giAdjustSL	2D - image of scattered light
sNorm(Nlambda,K)	O	giModelSky	2D - coefficients matrix of the polynomial expression of fitNormSky(Nlambda,Ns)
spLambda(Nx,Ns)	O	giGetWaveSolution	2D - lambda for each x-bin of each spectra
waveSolution(Ns,NparWv)	O	giGetWaveSolution	2D - adjusted analytical model of the wavelength solution
wlFfSp(Nlambda,Ns)	O	giRebinSpectra	2D - rebinned spectra
wlFfSpError(Nlambda,Ns)	O	giRebinSpectra	2D - standard error of the rebinned spectra
wlFlxSp(Nlambda,Nifu)	O	giFluxCal	2D - Nifu flux and wavelength calibrated object spectra

14 Compliance matrix

Item (spec.)	Compliance	Reference
Image arithmetics	YES	10.1
Cosmic rays and bad pixels correction	YES	9.5 (pp 25-26), 10.4, 10.5, 10.6, 10.7
Image statistics	YES	
Image truncation	YES	10.2 (overscan zones)
Image combination	YES	low level functional specifications of 10.6
Optimal spectrum extraction	YES	9.5 (pp 26-27) 10.10
Sky subtraction	YES	9.5 (p 29), 10.14, 10.15, 10.16
Optimal weighted sum of spectra with different S/N	PARTIAL	ADAS uses the functional specification 10.1
λ calibration based on calibration frames or on sky spectra	YES	9.5 (pp 27-29), 10.12
λ calibration based on a model of dispersion law vs. slit coordinates	YES	(mixed with preceding item) 9.5 (pp 27-29), 10.12
Flux calibration	YES	9.5 (p 29), 10.17
Multi-fibre image reconstruction	NO	This overlaps with ADAS. The specification is being postponed till the ADAS release

GIRAFFE BLDRS Functions Index

G

GIRAFFE BLDRS Variables Index

R

S

T

W

oOo