

EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

VISIR Pipeline User Manual

VLT-MAN-ESO-19500-3852

Issue 1.11

Date 17th May 2023

Prepared:	L.C.Luvaul	17th May 2023	
	Name	Date	Signature
Approved:	P.Ballester		
	Name	Date	Signature
Released:	M. Peron		
	Name	Date	Signature

No content on this page

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	1st February 2006	All	For VISIR pipeline 1.3.7
1.1	1st October 2007	All	For VISIR pipeline 2.0.0
1.3	21st January 2008	All	For VISIR pipeline 3.2.0
1.4	4th April 2014	All	For VISIR pipeline 3.5.1
1.5	25th February 2016	All	For VISIR pipeline 4.1.7
1.6	18th April 2016	All	For VISIR pipeline 4.2.0
1.7	3rd June 2020	All	For VISIR pipeline 4.3.10
1.8	20th May 2021	All	For VISIR pipeline 4.3.11
1.9	10th March 2022	All	For VISIR pipeline 4.4.0
1.10	30th May 2022	All	For VISIR pipeline 4.4.2
1.11	17th May 2023	All	For VISIR pipeline 4.4.5

No content on this page

Contents

1 Introduction **9**

1.1 Purpose . . . 9

1.2 Acknowledgements . . . 9

1.3 Scope . . . 9

1.4 Reference Documents and Applicable Documents . . . 10

2 Overview **12**

3 VISIR Instrument Description **13**

3.1 Instrument overview . . . 13

4 Quick start **15**

4.1 VISIR pipeline recipes . . . 15

4.2 An introduction to Gasgano and EsoRex . . . 15

4.2.1 Using Gasgano . . . 16

4.2.2 Using EsoRex . . . 18

4.3 Reflex . . . 21

4.3.1 Imaging GUI . . . 21

4.3.2 Spectroscopic GUI, or the “Aperture Editor” . . . 22

5 Known Problems **30**

6 Instrument Data Description **31**

6.1 General Data Layout - Aquarius detector . . . 31

6.1.1 Half-Cycle mode . . . 31

6.1.2 Burst mode . . . 31

6.2 General Data Layout - DRS detector . . . 32

6.3 General frames . . . 32

6.4 Imaging frames . . . 33

6.5 Spectroscopy frames . . . 33

7 Static Calibration Data **35**

7.1 Imaging Standard Stars . . . 35

7.2 Spectroscopy Standard Stars . . . 35

7.3 Spectrometer Detector Quantum Efficiency . . . 36

7.4 Atmospheric Emission Spectrum . . . 36

8 Data Reduction **37**

8.1 Reduction Cascade . . . 37

8.2 VISIR pipeline recipes . . . 37

8.3 Unsupported Observation Modes . . . 37

8.4 Sensitivity Function Correction . . . 38

9 Pipeline Recipe Interfaces **40**

9.1 visir_img_reduce . . . 40

9.1.1 Input files for visir_img_reduce . . . 40

9.1.2 Input Parameters for visir_img_reduce . . . 41

9.1.3 Products from visir_img_reduce . . . 41

9.1.4 QC Parameters from visir_img_reduce . . . 42

9.2 visir_img_dark . . . 43

9.2.1 Input files for visir_img_dark . . . 43

9.2.2 Input Parameters for visir_img_dark . . . 43

9.2.3 Products from visir_img_dark . . . 43

9.2.4 QC Parameters from visir_img_dark . . . 44

9.3 visir_spc_reduce . . . 44

9.3.1 Input files for visir_spc_reduce . . . 44

9.3.2 Input Parameters for visir_spc_reduce . . . 45

9.3.3 Products from visir_spc_reduce . . . 48

9.3.4 QC Parameters from visir_spc_reduce . . . 48

9.4 Product Description . . . 49

9.4.1 Imaging product data . . . 49

9.4.2 Spectroscopic product data . . . 49

10 Algorithms **53**

10.1 General Algorithms . . . 53

10.1.1 Bad pixel detection and cleaning . . . 53

10.1.2 Distortion correction . . . 53

10.1.3 Creation of combined image . . . 53

10.1.4 Imaging Photometric Calibration . . . 55

10.1.5 Wavelength Calibration . . . 56

10.1.6 Spectrum Extraction . . . 57

10.1.7 Spectral Photometric Calibration . . . 58

10.1.8 Imaging Photometric Calibration . . . 59

10.1.9 Computation of Strehl Ratio . . . 59

10.2 Recipe Algorithms . . . 59

10.2.1 visir_img_ff . . . 59

10.2.2 visir_img_dark . . . 60

10.2.3 visir_img_reduce . . . 60

10.2.4 visir_spc_reduce . . . 60

A Installation **65**

A.1 Supported platforms . . . 65

A.2 Building the VISIR pipeline . . . 65

A.2.1 Requirements . . . 65

A.2.2 Compiling and installing the VISIR pipeline . . . 66

B QC Parameters **67**

C Abbreviations and acronyms **80**

1 Introduction

1.1 Purpose

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

This manual is a complete description of the data reduction recipes used by the VISIR pipeline, reflecting the status of the VISIR pipeline as of 17th May 2023 (version 4.4.5).

1.2 Acknowledgements

We want to thank Eric Pantin, CEA and Ralf Siebenmorgen for providing valuable ideas for improving the pipeline recipes. We thank also Yves Jung, who played a major role in the development of the first version of the pipeline. The feedback we received in numerous discussions with our "beta-testers", VISIR Instrument Scientists Alain Smette, Stephane Brillant and Konrad Tristram, VISIR Quality Control Scientists Danuta Dobrzycka and Burkhard Wolff, and VISIR User Support Scientist Mario van den Ancker was very much appreciated.

In particular we want to thank Daniel Asmus and Valentin Ivanov, who were a continuous source of useful ideas for improving the pipeline recipes, and for their extensive testing.

1.3 Scope

This document describes the VISIR pipeline used at ESO-Garching and ESO-Paranal for the purpose of data assessment and data quality control.

Updated versions of the present document may be found on [17]. For general information about the current instrument pipelines status we remind the user of [7]. Quality control information are at [6].

Additional information on QFITS, the Common Pipeline Library (CPL) and ESOREX can be found respectively at [12], [15]. The Gasgano tool is described in [16]. A description of the instrument is in [8]. The VISIR instrument user manual is in [9] while results of Science Verifications (SV) are at [3].

The examples on running individual pipeline recipes in this manual use the *EsoRex* command and manually created list of input files. Several interfaces to automatically organise the data, create the list of input files and execute the pipeline recipes in the proper sequence are available, see the ESO Pipeline web page *https://www.eso.org/pipelines* for details.

Please note that the use of Gasgano as a GUI for processing data is deprecated. Its use is no longer recommended and the related section in this manual, as well as support for Gasgano as a data processing GUI application in general will be dropped entirely in a future release.

1.4 Reference Documents and Applicable Documents

[1] ESO. *Parameters for setting the VISIR Spectrometer*. VLT-TRE-VIS-14321-5046.

[2] ESO. *VISIR Calibration Plan*. VLT-PLA-VIS-14300-0009.

[3] ESO, *http://www.eso.org/science/vltsv/visirsv/*. *VISIR Science Verification home page*. 9

[4] ESO. *VLT Data Flow System Specifications for Pipeline and Quality Control*. VLT-SPE-ESO-19600-1233.

[5] ESO/DMO. *ESO DICB – Data Interface Control Document*. GEN-SPE-ESO-19400-0794 (3.0). 32, 33, 34

[6] ESO/DMO/DFO, *http://www.eso.org/observing/dfo/quality/*. *ESO-Data Flow Operation home page*. 9

[7] ESO/DMO/DFO, *http://www.eso.org/sci/software/pipelines/*. *VLT Instrument Pipelines*. 9

[8] ESO/INS, *http://www.eso.org/instruments/visir/*. *VISIR instrument home page*. 9

[9] ESO/INS, *http://www.eso.org/instruments/visir/doc/*. *VISIR instrument home page*. 9, 13, 35, 36, 53

[10] ESO/SDD/DFS, *http://www.eso.org/observing/cpl/download.html*. *Common Pipeline Library Reference Manual*. VLT-ESO-MAN-19500-2721.

[11] ESO/SDD/DFS, *http://www.eso.org/observing/cpl/download.html*. *Common Pipeline Library User Manual*. VLT-MAN-ESO-19500-2720.

[12] ESO/SDD/DFS, *http://www.eso.org/cpl*. *CPL home page*. 9, 53, 59

[13] ESO/SDD/DFS. *Deliverables Specification*. VLT-SPE-ESO-19000-1618 (2.0).

[14] ESO/SDD/DFS. *DFS Pipeline & Quality Control – User Manual*. VLT-MAN-ESO-19500-1619.

[15] ESO/SDD/DFS, *http://www.eso.org/cpl/esorex.html*. *ESOREX home page*. 9

[16] ESO/SDD/DFS, *http://www.eso.org/gasgano/*. *Gasgano User's Manual*. VLT-PRO-ESO-19000-1932. 9

[17] ESO/SDD/DFS, *http://www.eso.org/pipelines*. *VISIR Pipeline Web Page*. 9

[18] Rio Y. et al. *VISIR: A mid infrared imager and spectrometer for the VLT*, spie vol. 2475, pp. 286-295, orlando edition, April 1995. VLT-TRE-VIS-14321-5046.

[19] Rio Y. et al. *VISIR: The mid infrared imager and spectrometer for the VLT*, spie vol. 3354, pp. 615-626, kona, hawaii edition, March 1998.

[20] Smette A. et al. *Molecfit: A general tool for telluric absorption correction*. Astronomy & Astrophysics 576, A77, April 2015. 38

[21] P.-O. Lagage. *The final design of VISIR, the mid-infrared imager and spectrometer for the VLT*.

[22] P.-O. Lagage et al. *Result of the phase A study for the VLT Mid-infrared instrument: VISIR*, the eso messenger no. 80, pp. 13-16 edition, June 1995.

[23] P.-O. Lagage et al. *VISIR at PDR*, the (eso) messenger, no. 91, pp. 17-21 edition, March 1998.

2 Overview

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

Data quality control: pipelines are used to produce the quantitative information necessary to monitor instrument performance.

Master calibration product creation: pipelines are used to produce master calibration products (*e.g.*, combined bias frames, super-flats, wavelength dispersion solutions).

Science product creation: using pipeline-generated master calibration products, science products are produced for the supported instrument modes (*e.g.*, combined ISAAC jitter stacks; bias-corrected, flat-fielded FORS images, wavelength-calibrated UVES spectra). The accuracy of the science products is limited by the quality of the available master calibration products and by the algorithmic implementation of the pipelines themselves. In particular, adopted automatic reduction strategies may not be suitable or optimal for every scientific goal.

Instrument pipelines consist of a set of data processing modules that can be called from the command line, from the automatic data management tools available on Paranal or from Gasgano.

ESO offers two front-end applications for launching pipeline recipes, *Gasgano* [6] and *EsoRex*, both included in the pipeline distribution (see Appendix A on page 65). These applications can also be downloaded separately from *http://www.eso.org/gasgano* and *http://www.eso.org/cpl/esorex.html*. An illustrated introduction to Gasgano is provided in the "Quick Start" section 4 on page 15.

The VISIR instrument and the different types of VISIR raw frames and auxiliary data are described in sections 3 on the facing page, 6 on page 31, and 7 on page 35.

A brief introduction to the usage of the available reduction recipes using Gasgano or EsoRex is presented in section 4 on page 15. In section 5 on page 30 we advice the user about known data reduction problems providing also possible solutions.

An overview of the data reduction, what are the input data, and the recipes is provided in section 8 on page 37.

More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe is given in section 9 on page 40.

More detailed descriptions of the data reduction algorithms used by the individual pipeline recipes can be found in section 10 on page 53.

In appendix A on page 65 the installation of the VISIR pipeline recipes is given.

3 VISIR Instrument Description

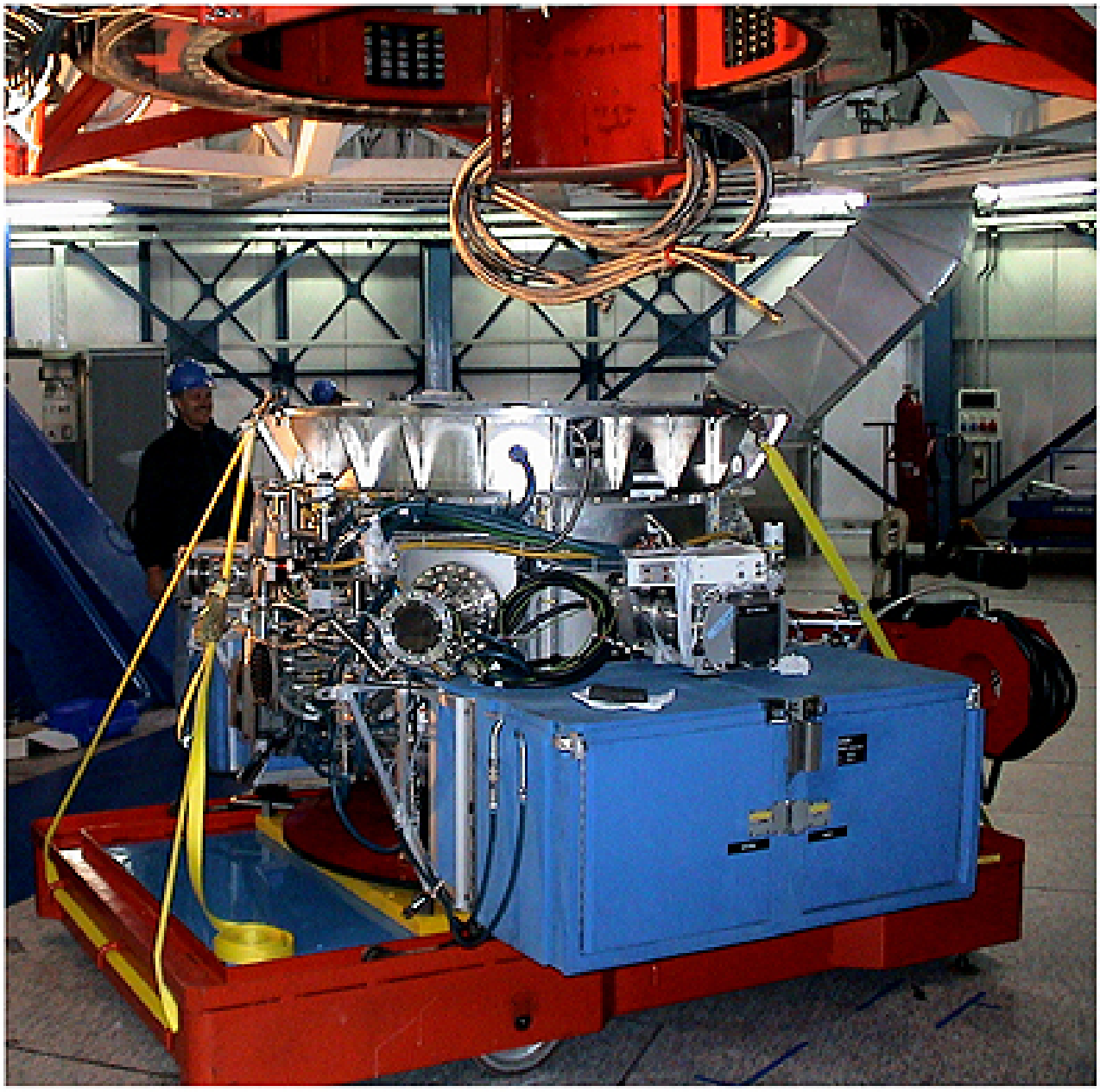
This section provides a brief description of the VISIR instrument.

A more complete documentation can be found in the VISIR User Manual, downloadable from *http://www.eso.org/instruments/visir/doc*

VISIR has been developed under ESO contract by CEA/DAPNIA/SAP and NFRA/ASTRON. The instrument has been made available to the community and started operations in Paranal in April 2005.

3.1 Instrument overview

For a description of the VISIR instrument, please see [9]. *Parameters for setting the VISIR Spectrometer* (VLT-TRE-VIS-14321-5046) contains more information about the VISIR spectrometer.



VISIR under the Cassegrain Focus of the 8.2-m VLT Melipal Telescope

ESO PR Photo 16a/04 (12 May 2004)

© European Southern Observatory

Figure 3.1.1: *VISIR under the Cassegrain Focus of the 8.2m VLT Melipal Telescope.*

4 Quick start

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

4.1 VISIR pipeline recipes

For reduction of science and calibration data, the current VISIR pipeline offers several recipes of which most of them are internal utility or legacy recipes. Currently the following four recipes can be used to handle raw input data:

visir_img_reduce Executes the full imaging reduction chain including stacking of exposures and photometry. Internally it calls a collection of sub recipes which are also available as standalone versions.

visir_spc_reduce Executes the full spectroscopy reduction chain including stacking of exposures, undistortion, wavelength calibration and photometry. Internally it calls a collection of sub recipes which are also available as standalone versions.

visir_util_repack Repacks the various raw detector data formats (see 6) into a format more suitable for following processing steps. This includes extracting the single exposures into separate files and optionally applying the chop/nod background subtraction.

visir_img_dark creates a dark image and maps of hot, cold and deviant pixels from a sequence of dark exposures in both imaging and spectroscopy.

Additionally, there are other recipes provided to help with non-reduction tasks:

visir_util_img_std_cat Takes a properly formatted text file and generates a standard star catalogue from it in the form of a BINTABLE FITS file. Usage is like **esorex visir_util_img_std_cat test.sof** where `test.sof` contains a single line containing the name of the text file and `IMG_LINES_ASCII` separated by whitespace. The text file content must be formatted as discussed by the recipe manpage, which can be seen by running **esorex -man visir_util_img_std_cat**.

4.2 An introduction to Gasgano and EsoRex

Before being able to call pipeline recipes on a set of data, the data must be properly classified, and associated with the appropriate calibrations. The *Data Classification* consists of tasks such as: "What kind of data am I?", *e.g.*, FLAT, "to which group do I belong?", *e.g.*, to a particular Observation Block or template. *Data Association* is the process of selecting appropriate calibration data for the reduction of a set of raw science frames. Typically, a set of frames can be associated if they share a number of properties, such as instrument and detector configuration. As all the required information is stored in the FITS headers, data association is based on a set of keywords (called "association keywords") and is specific to each type of calibration[1].

[1]The data association is based on the value of the triplet of FITS keys DPR.CATG, DPR.TYPE and DPR.TECH

The process of data classification and association is known as data organisation. The *DO Category* is the label assigned to a data type as a result of data classification.

An instrument pipeline consists of a set of data processing modules that can be called from different host applications, either from the command line with *EsoRex*, from the automatic data management tools available at Paranal, or from the graphical *Gasgano* tool.

Gasgano is a data management tool that simplifies the data organisation process, offering automatic data classification and making the data association easier (*even if automatic association of frames is not yet provided*). *Gasgano* determines the classification of a file by applying an instrument specific rule, while users must provide this information to the recipes when they are executed manually using *EsoRex* from the command line. In addition, *Gasgano* allows the user to execute directly the pipeline recipes on a set of selected files.

4.2.1 Using Gasgano

To get familiar with the VISIR pipeline recipes and their usage, it is advisable to begin with *Gasgano*, because it provides a complete graphic interface for data browsing, classification and association, and offers several other utilities such as easy access to recipes documentation and preferred data display tools.

Gasgano can be started from the Command Line Interface in the following way:

```
gasgano &
```

Figure 4.2.1 shows the *Gasgano* main window with the 4 VISIR calibration files automatically loaded, which is the configuration of the publicly available version 4.4.5 of the VISIR pipeline.

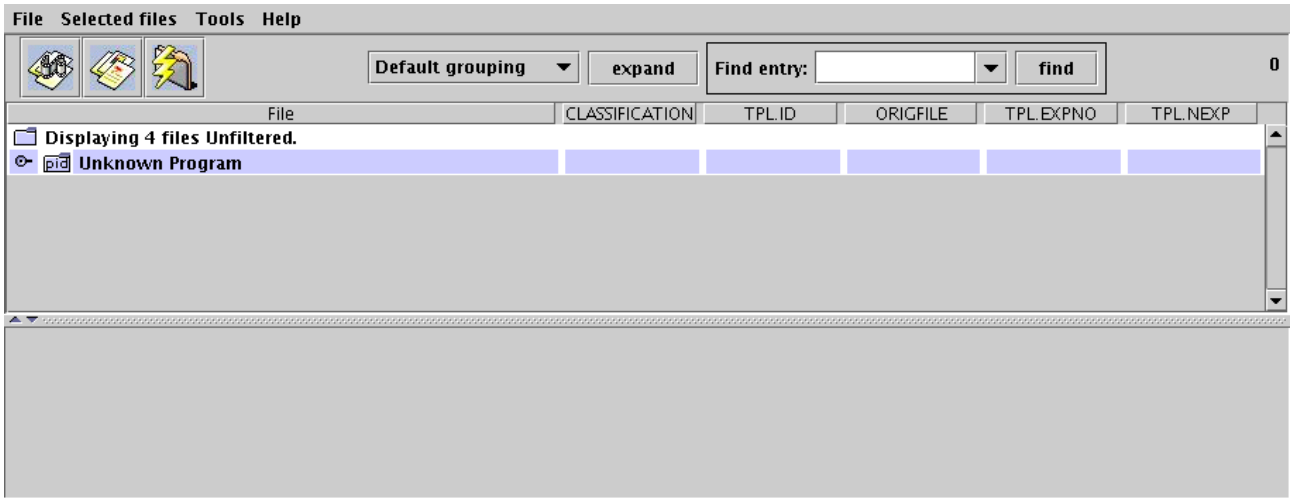


Figure 4.2.1: *The Gasgano main window with the VISIR calibration files.*

The VISIR calibration files become visible with their DO categories, when the file browser is expanded as shown in figure 4.2.2 on the facing page. With the pull-down-menu *File->Add/Remove Files* directories containing

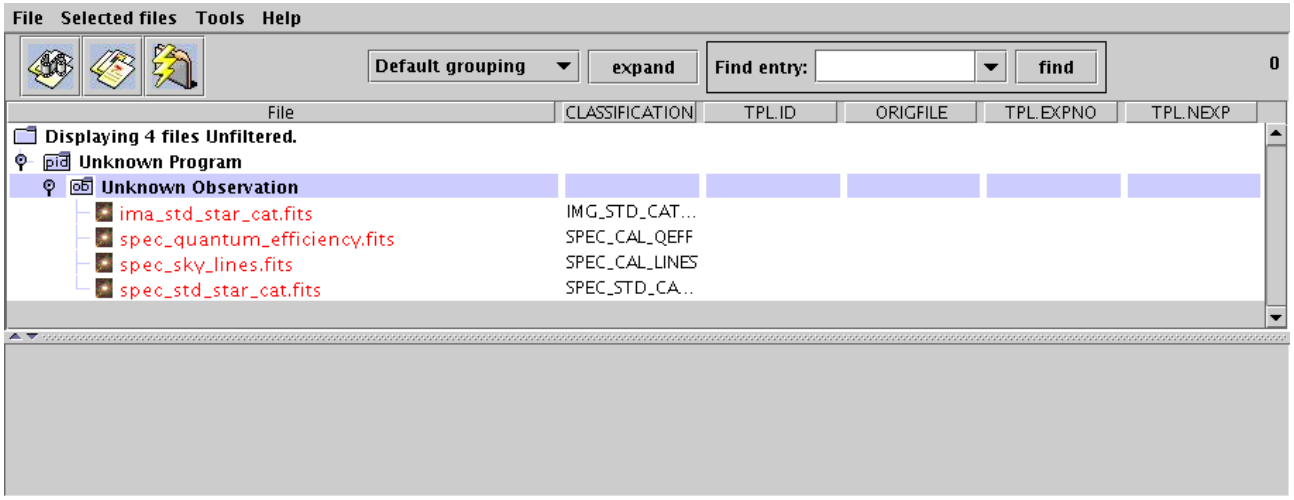


Figure 4.2.2: *The Gasgano main window with expanded file view of the initially loaded calibration files.*

VISIR data can be added, as shown in figure 4.2.3 on the next page. Figure 4.2.4 on page 19 shows the example data set distributed with the public release of the VISIR pipeline.

The data are hierarchically organised as preferred by the user. After each file name are shown the classification, the template id, the original filename, the template exposure number and the number of exposures in the template.

More information about a single frame can be obtained by clicking on its name: the corresponding FITS file header will be displayed on the bottom panel, where specific keywords can be opportunely filtered and searched. Images and tables may be easily displayed using the viewers specified in the appropriate *Preferences* fields.

Frames can be selected from the main window with a <CTRL>-left-click for processing by the appropriate recipe: on Figure 4.2.5 on page 20, the example spectroscopic FITS-files and the three spectroscopic calibration files have been selected and the pull-down-menu with the VISIR recipes is shown.

Selecting the appropriate recipe, *visir_spc_reduce* , will open a *Gasgano* recipe execution window (see Figure 4.2.6 on page 25), having all the specified files listed in its *Input Frames* panel.

Help about the recipe may be obtained from the *Help* menu. Before launching the recipe, its parameters may be modified on the *Parameters* panel (on top). The window contents might be saved for later use by selecting the *Save Current Settings* entry from the *File* menu, as shown in figure.

At this point the recipe can be launched by pressing the *Execute* button. Messages from the running recipe will appear on the *Log Messages* panel at bottom, and in case of successful completion the products will be listed on the *Output Frames* panel, where they can be easily viewed and located back on the Gasgano main window. The successful processing of the example data can be seen in figure 4.2.7 on page 26.

Please refer to the *Gasgano User's Manual* [6] for a more complete description of the *Gasgano* interface.

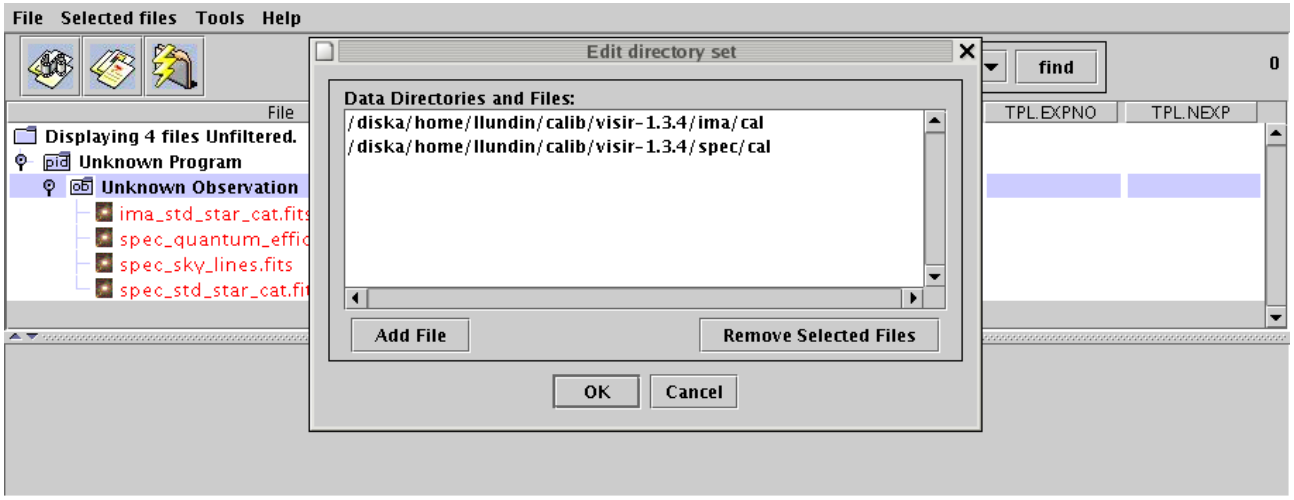


Figure 4.2.3: *The Gasgano main window with the file loader window on top.*

4.2.2 Using EsoRex

EsoRex is a command line utility for running pipeline recipes. It may be embedded by users into data reduction scripts for the automation of processing tasks. On the other side, *EsoRex* doesn't offer all the facilities available with *Gasgano*, and the user must classify and associate the data using the information contained in the FITS header keywords (see section 6.3 on page 32). The user should also take care of defining the input set-of-frames and the appropriate configuration parameters for each recipe run:

The set-of-frames: Each pipeline recipe is run on a set of input FITS data files. When using *EsoRex* the filenames must be listed together with their DO category in an ASCII file, the *set-of-frames* (SOF), that is required when launching a recipe. [2]

Here is an example of an SOF suitable for the *visir_old_spc_obs* recipe:

```
VISIR.2004-09-03T23:23:11.260.fits SPEC_OBS_LMR
VISIR.2004-09-03T23:24:05.263.fits SPEC_OBS_LMR
VISIR.2004-09-03T23:24:52.508.fits SPEC_OBS_LMR
VISIR.2004-09-03T23:25:45.026.fits SPEC_OBS_LMR
spec/cal/spec_sky_lines.fits SPEC_CAL_LINES
spec/cal/spec_quantum_efficiency.fits SPEC_CAL_QEFF
```

Note that the VISIR pipeline recipes do not verify the correctness of the DO category specified by the user in the SOF. The reason of this lack of control is that VISIR recipes are just one component of the complete pipeline running on Paranal, where the task of data classification and association is carried out by separate applications. Moreover, using *Gasgano* as an interface to the pipeline recipes will always ensure a correct classification of all the data frames, assigning the appropriate DO category to each one of them (see section 4.2.1 on page 16).

[2] The set-of-frames corresponds to the *Input Frames* panel of the *Gasgano* recipe execution window (see figure 4.2.6 on page 25).

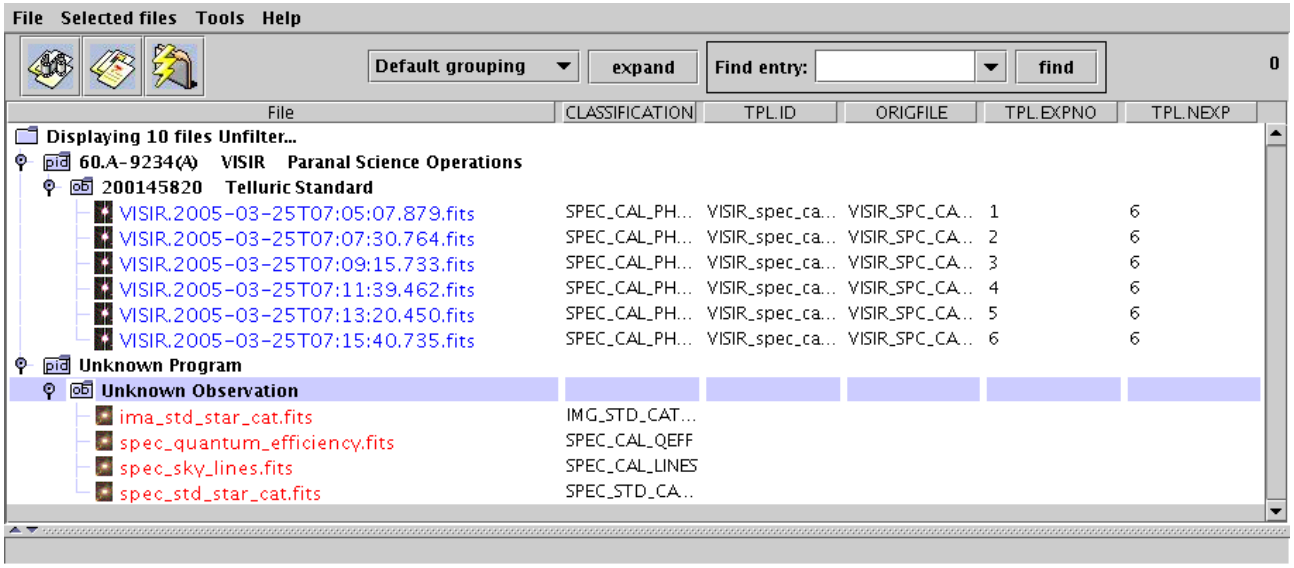


Figure 4.2.4: *The Gasgano main window with the example FITS files loaded.*

Such a procedure requires that the input FITS data are given a classification different from the one they should have.

If such a classification is done with FITS data that are not suitable for the given recipe, the recipe will most likely complete with a more or less descriptive error message, but there is a risk that the recipe will complete without any indication that the input is in fact invalid and the output flawed.

EsoRex syntax: The basic syntax to use ESOREX is the following:

esorex [esorex_options] recipe_name [recipe_options] set_of_frames

To get more information on how to customise ESOREX (see also [6]) run the command:

esorex --help

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

esorex --create-config

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

esorex --recipes

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

esorex --params recipe_name

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

esorex --help recipe_name

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

esorex --man-page recipe_name

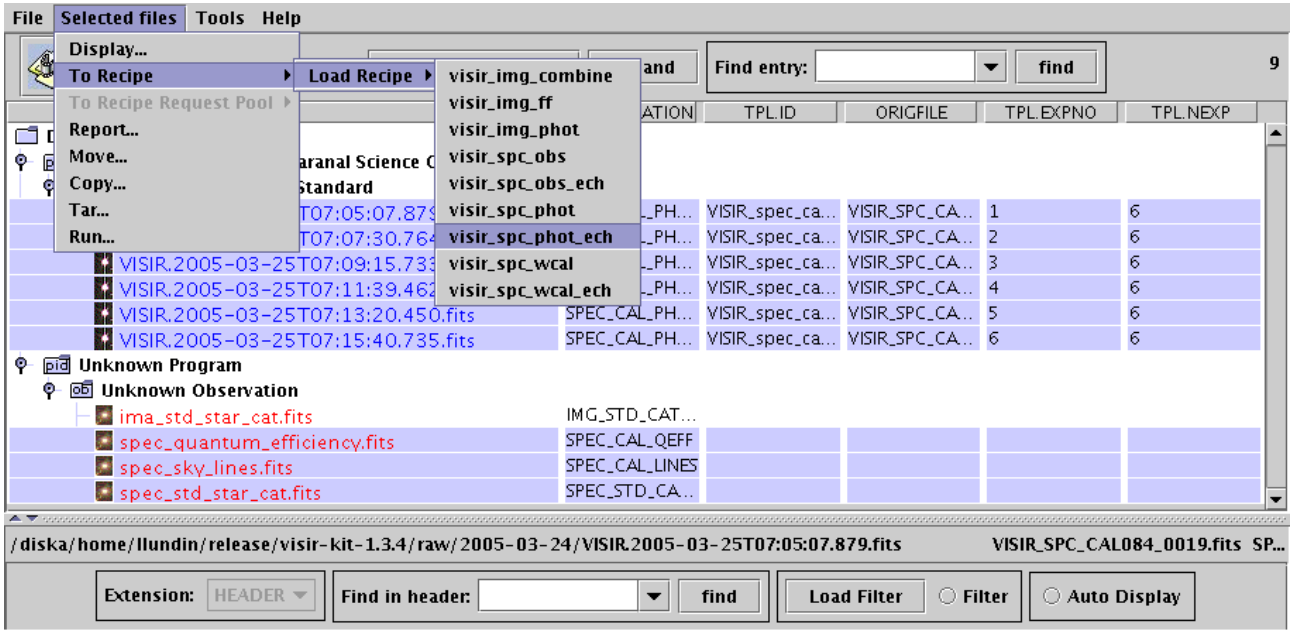


Figure 4.2.5: *The Gasgano main window with selected files and the pull-down-menu with the VISIR recipes.*

Recipe configuration: Each pipeline recipe may be assigned an *EsoRex* configuration file, containing the default values of the parameters related to that recipe.[3] The configuration files are normally generated in the directory `$HOME/.esorex`, and have the same name as the recipe to which they are related, with the filename extension `.rc`. For instance, the recipe *visir_img_reduce* has its *EsoRex* generated configuration file named `visir_img_reduce.rc`, and is generated with the command:

esorex - -create-config visir_img_reduce - -jy_val=5

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

```
visir.visir_img_reduce.jy_val=5.
```

In this example, the parameter visir.visir_img_reduce.low is set to the value 5. In the configuration file generated by *EsoRex*, one or more comment lines are added containing information about the possible values of the parameter, and an alias that could be used as a command line option.

Given a recipe named *visir_recipe_name* the command

esorex - -create-config visir_recipe_name

Generates a default configuration file **visir_recipe_name.rc** in the directory **$HOME/.esorex**[4].

A recipe configuration file different from the default one can be specified on the command line:

esorex - -recipe-config=my_alternative_recipe_config

[3]The *EsoRex* recipe configuration file corresponds to the *Parameters* panel of the *Gasgano* recipe execution window (see figure 4.2.6 on page 25).

[4]If a number of recipe parameters are specified on the command line, the given values will be used in the created configuration file.

Recipe parameters are provided in section 9 on page 40 and their role is described in section 10 on page 53.

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

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

esorex visir_spc_reduce visir_spc_reduce.sof

The recipe parameters can be modified either by editing directly the used configuration file, or by specifying new parameter values on the command line using the command line options defined for this purpose. Such command line options should be inserted after the recipe name and before the SOF name, and they will supersede the system defaults and/or the configuration file settings. For instance, to set the *visir_img_reduce* recipe *jy_val* threshold parameter to `5`, the following should be typed:

esorex visir_img_reduce - -jy_val=0.5 visir_img_reduce.sof

Here are some more examples of running a recipe:

```
esorex --output-prefix=test visir_img_reduce test.sof
esorex --msg-level=debug visir_spc_reduce spc_phot.sof
```

For more information on *EsoRex*, see *http://www.eso.org/cpl/esorex.html*.

4.3 Reflex

`EsoReflex` is the ESO Recipe Flexible Execution Workbench, an environment to run ESO VLT pipelines which employs a workflow engine to provide a real-time visual representation of a data reduction cascade, called a workflow, which can be easily understood by most astronomers. The basic philosophy and concepts of Reflex have been discussed by Freudling et al. (2013A&A...559A..96F). Please reference this article if you use Reflex in a scientific publication.

Reflex and the data reduction workflows have been developed by ESO and instrument consortia and they are fully supported. If you have any issue, please have a look to https://support.eso.org to see if this has been reported before or open a ticket for further support.

EsoReflex is the recommended environment to run ESO pipelines. More information is available in the VISIR Reflex tutorial.

4.3.1 Imaging GUI

The imaging GUI provides a quick overview of the results of the processing step which is in charge of detecting objects, determining shifts between them and rejecting bad exposures. The GUI (Figure 4.3.1 on page 27) presents these results in several plots and allows to tune the parameters the recipe.

On the right hand side the recipe parameters used in the last iteration are displayed (see 1 on 4.3.1 on page 27) Below the parameter boxes are two buttons to either accept the current results or repeat the recipe with the changed parameters.

The left hand area contains several plots which allow to quickly explore the data and the results of the object detection. The primary line plot (see 2 on 4.3.1 on page 27) shows several statistics for each chop nod corrected image in the input data. By default it shows the correlation to of each image to the correlation template shown in plot 5. By clicking on the radiobox next to the image the shown statistics can be changed, currently available are:

- Correlation: normalized correlation to the template
- Full width half maximum of the object in X and Y
- Median absolute deviation of the image
- Standard deviation of the image

The statistic plot can be clicked to see the corresponding image in the center middle plot (4).

The image middle left image (see 3 on 4.3.1 on page 27) shows the average of all chop-nod corrected images in the raw data without any correction applied. The diamonds in this image show the detected pivot beam positions.[5] When the beam detection failed the button above 3 will be colored red. If one wants the rest of the reduction to not consider the whole image as one beam the button must be clicked to select the rough positions of the beams manually. It can also be used to only select a single object for the rest of the reduction instead of all of them.

By right clicking on the image plots one can change the used color map and scale of the images. Also one can detach the images to view them at a larger scale.

The bottom plot 6 shows a scatter plot detected shifts between the objects. These shifts will be applied during the coaddition phase of the reduction.

4.3.2 Spectroscopic GUI, or the "Aperture Editor"

The Aperture Editor is responsible for creating an aperture definitions file and passing it to the `visir_old_spc_obs` recipe (via its `--apfile`[6] parameter) whenever the user clicks the *Re-run Recipe* button. Figures 4.3.2 and 4.3.3 show the Aperture Editor illustrating an optimal extraction and an aperture sum configuration, respectively. Within the GUI, the left mouse button is used exclusively in most panels. However the combined spectral image panel to the left uses a combination of the left and middle mouse buttons: the left mouse button is used to zoom or pan the image, while the middle mouse button is used to add, modify, or delete apertures.

On the left the GUI displays the combined 2D spectrum. The horizontal bar on the bottom shows the width of the aperture that is to be added or resized. The vertical panel in the center allows the user to perform various actions, set up some parameters and, in the bottom block, see the status of the current aperture selections (e.g.

[5] These positions must not be exact, it is just used to determine the region in the image where a beam is present

[6] The `visir_spc_reduce` recipe also supports this parameter.

if there are conflicts like missing sky apertures or overlapping science and sky apertures). The right side is identical with the GUIs for other interactive workflows. The users have two color schemes available, depending on the contrast - they can be alternated by clicking on the "Light" and "Contrast" buttons in the central vertical panel. The display labels the order ID (relevant for cross-dispersed spectra) along the X-axis and the wavelength in microns for the active order along the Y-axis (in case of cross-dispersed spectra, a star next to the order ID indicates which order the wavelength on the Y-axis corresponds to).

Figure 4.3.4 shows the GUI for a cross-dispersed spectroscopic mode. Green and red order boundaries indicate which orders have been successfully extracted or not, respectively. For the example shown in the figure, the algorithm failed for order -2, hence it is shown in red. The most common problem is with the wavelength calibration (which usually happens for spectral regions with strong atmospheric absorption) although a number of other failure modes are possible.

Textured apertures within the orders indicate the extraction algorithm (in case of optimal extraction, no sky apertures are defined) and the sky calculation algorithm (in case of extraction with a sum within the target aperture). **Circles** are for optimal extraction whereas, in the case of summing, the aperture is textured depending on the way the sky is calculated, with **crossed lines** for averaging of all pixels within all sky apertures, **horizontal lines** for linear fitting of all sky pixels, and **stars** for median averaging of all sky pixels. Note that the sky apertures themselves are not textured, so they can clearly be distinguished from the target aperture (which is important if the sky and target apertures touch).

The central vertical panel is composed of six (6) separate blocks, whose purpose is described below:

- **The top block:** This block contains parameters rl and rr - these define how many pixels from the order to skip, counting from the detected left and right edges inwards, respectively. Negative values would make the orders wider, but are not recommended. For long-slit the recommended values of rl and rr, to avoid the negative spectra on the stacked 2D images, are in the range 70-100; for cross-dispersed mode in most cases they should be close to zero.
- **The second block from the top:** This block defines the main actions. They are selected with a left mouse button click and serve respectively to add a new aperture (if there is already an aperture, it will be moved), to re-size and shift an existing aperture, to delete an aperture and finally at the bottom - to define the method of extraction within that aperture. Once the action is selected with the left button click on the oval buttons, then the actual action is performed by clicking with the middle mouse button on the image on the left side of the GUI. Note that optimal extraction is applied across the entire aperture, so the apertures for this method need to be wider, to encompass enough sky.
- **The third block from the top:** This block is just a legend of the selected extraction method. The method alternates along the sequence by (i) selecting "Set Method" and (ii) middle-button clicking on the selected aperture enough times until the desired method comes up, which is recognizable from the changed pattern on the left 2D image.
- **The fourth block from the top:** This block defines the layer: only one aperture can be defined per order per layer; up to ten layers can be defined for each order. Therefore, if the user needs multiple apertures, e.g. with different extraction methods and/or different widths, each of those apertures must be defined in a different layer. The layers are selected by left mouse button clicking on the layer button.
- **The fifth block from the top:** This block sets the display mode - respectively optimized for bright or faint sources.

- **The bottom block:** This block reports the status of the current aperture definitions - the GUI automatically checks for errors such as missing sky apertures for aperture (non-optimal) extraction or for overlapping apertures. This is automatic and the user only needs to inspect the reported order/layer combo and correct the problematic aperture.

The aperture definitions file that the Aperture Editor generates is an ASCII file, where each line defines a set of related apertures for a single order. More than one line may target the same order, so there can be multiple aperture sets, and thus many apertures, defined for the same order.

A line starts with `O` or `A` for optimal extraction or aperture sum extraction, respectively. This is followed by two integer numbers defining the width of the target aperture. They are the offsets, in pixels, of the left and right edges of the target aperture from the left edge of the image.

If aperture sum extraction (`A`) is selected, the line must then continue with a description of the sky apertures. This description begins with a letter `A`, `F` or `M` (respectively: average, linear fit, or median) defining the method used to determine the sky level, which is calculated from the pixels in all sky apertures. After the sky method character must come one or more sky aperture definitions, each composed of two integer numbers (representing the distance of the left and right edges of the aperture, in pixels, from the image's left edge). So there must be an even number of integers after the sky method character.

Below are some example aperture file lines:

```
O 248 279
A 253 275 A 247 250 277 280
A 339 357 A 329 334
A 126 142 M 117 122 144 149 154 169
```

Users should try to define two sky apertures, one on each side of the target, to bracket the spatial sky variation. The combined width of all sky apertures should be equal to or greater than the width of the aperture of the science target except for the optimal extraction where sky apertures are not required or allowed. Instead, the science aperture should be wide enough to cover clear sky. Three or more sky apertures may be useful if there is a bad pixel or a cluster of bad pixels or nearby source that needs to be avoided, so they don't contaminate the sky (this is not likely to happen for VISIR, but the GUI is aimed at other long slit spectrographs).

The aperture file is generated by the VISIR spectroscopic workflow, but if the pipeline is executed with esorex, this is an optional input file. If the user wants to take advantage of this feature and define the apertures this way, the file has to be generated by the user with an ordinary text editor.

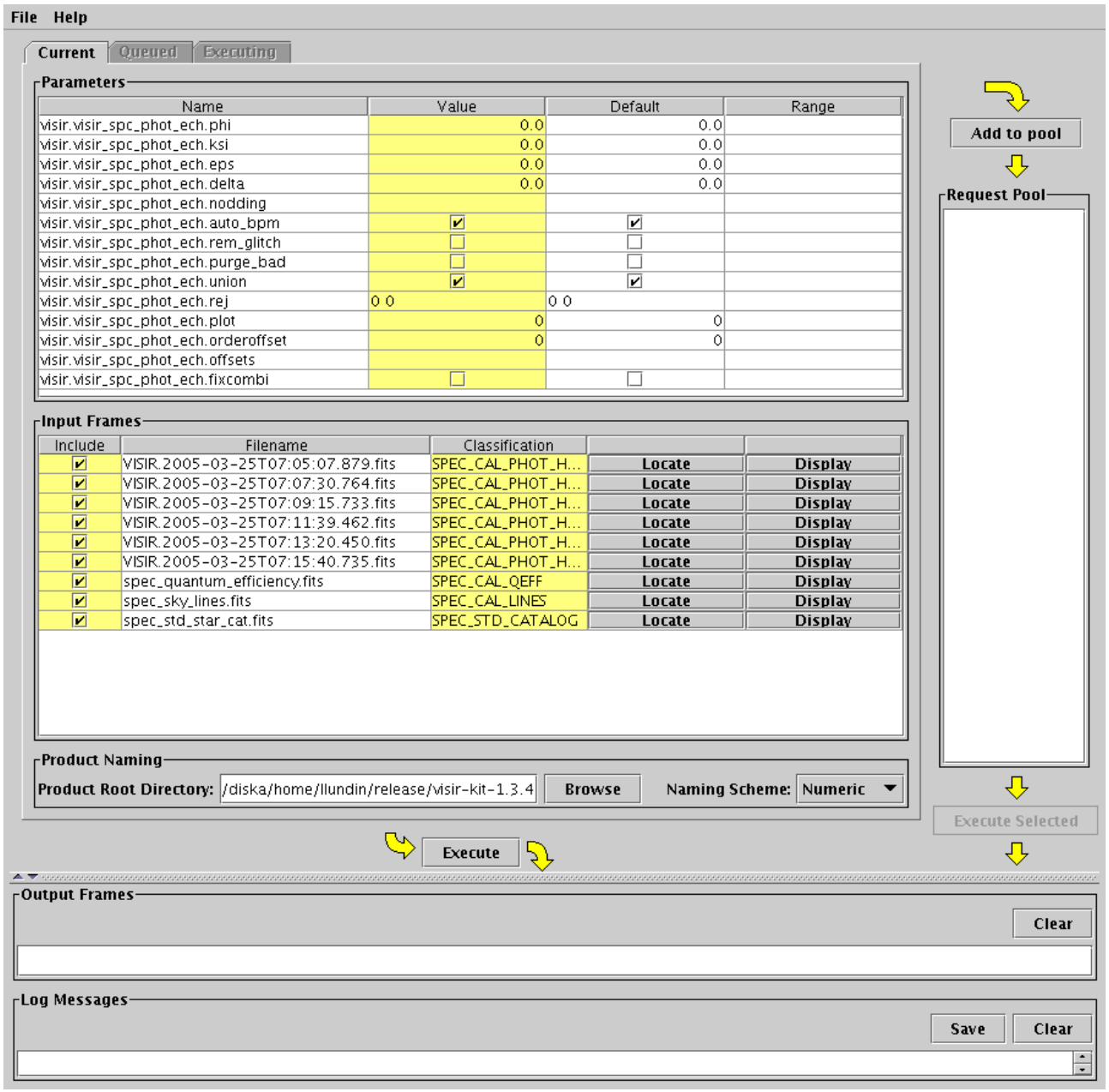


Figure 4.2.6: *The Gasgano recipe window with the recipe visir_spc_phot_ech.*

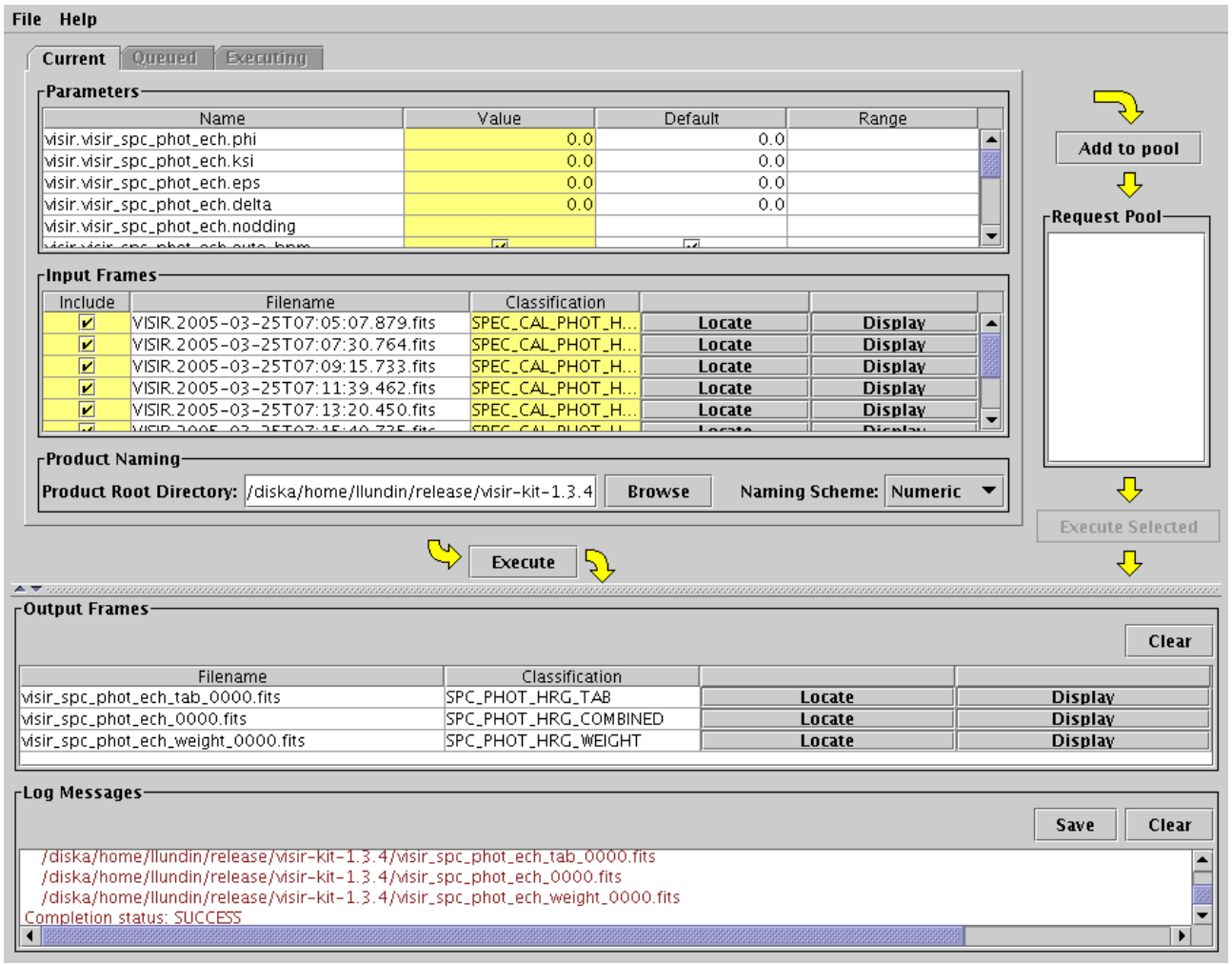


Figure 4.2.7: *The Gasgano recipe window with the recipe visir_spc_phot_ech successfully completed.*

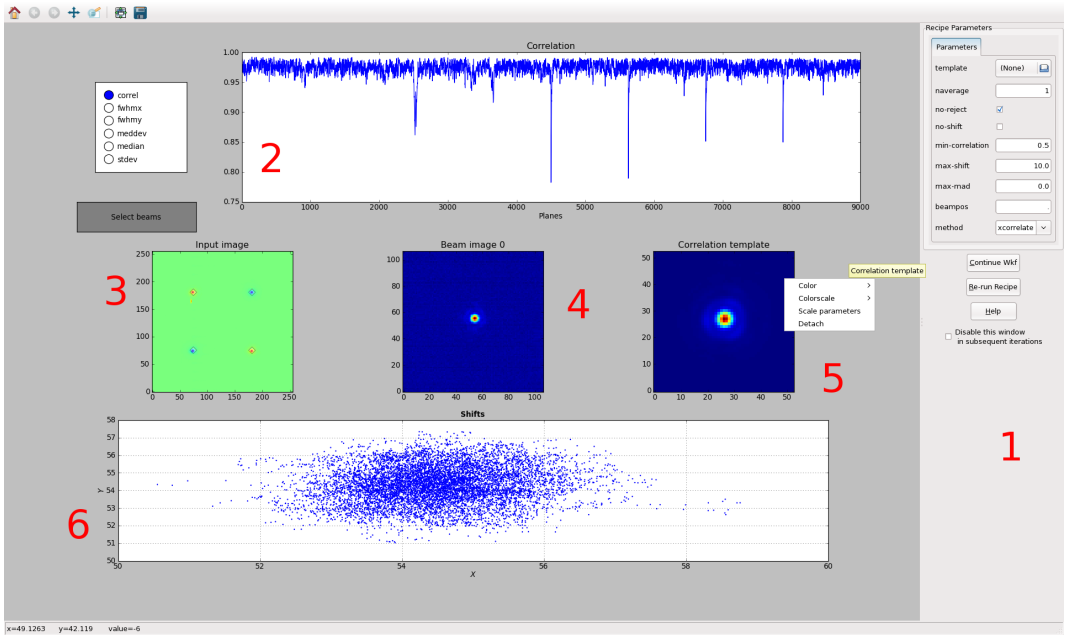


Figure 4.3.1: Screenshot of the imaging GUI.

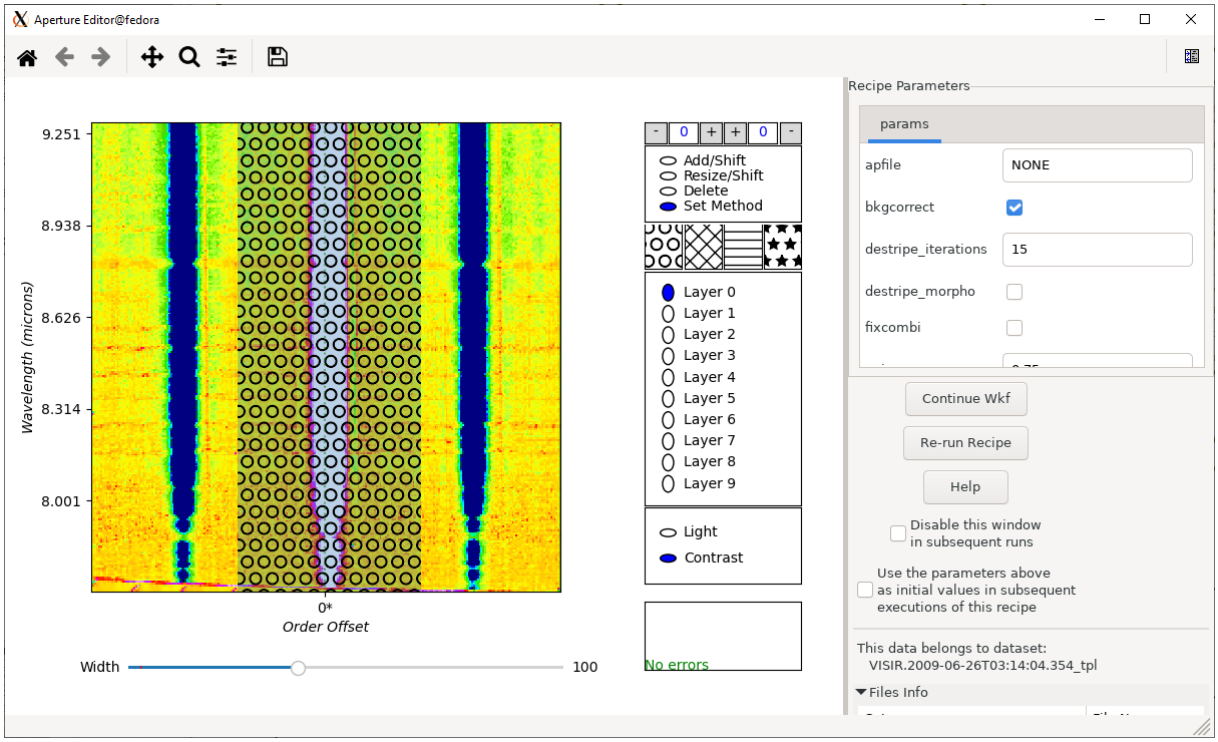


Figure 4.3.2: *Long slit mode: optimal extraction aperture being configured.*

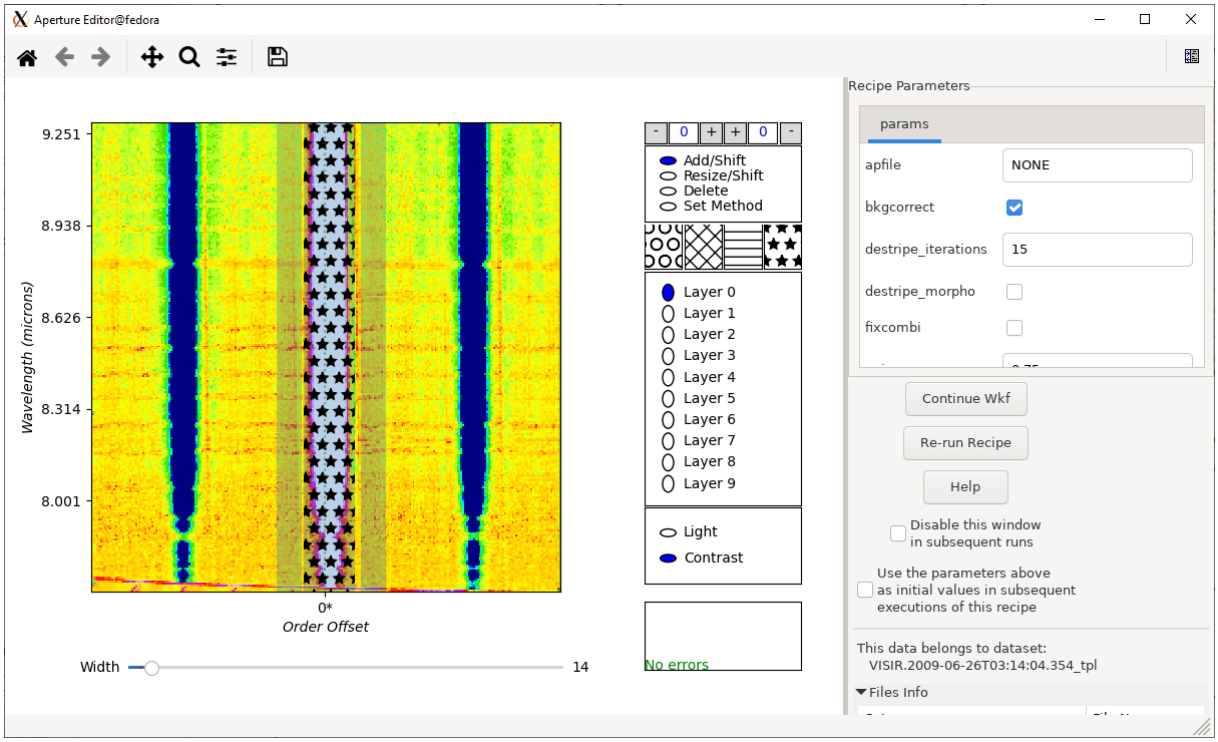


Figure 4.3.3: *Long slit mode: target aperture (to be summed) and sky apertures being configured.*

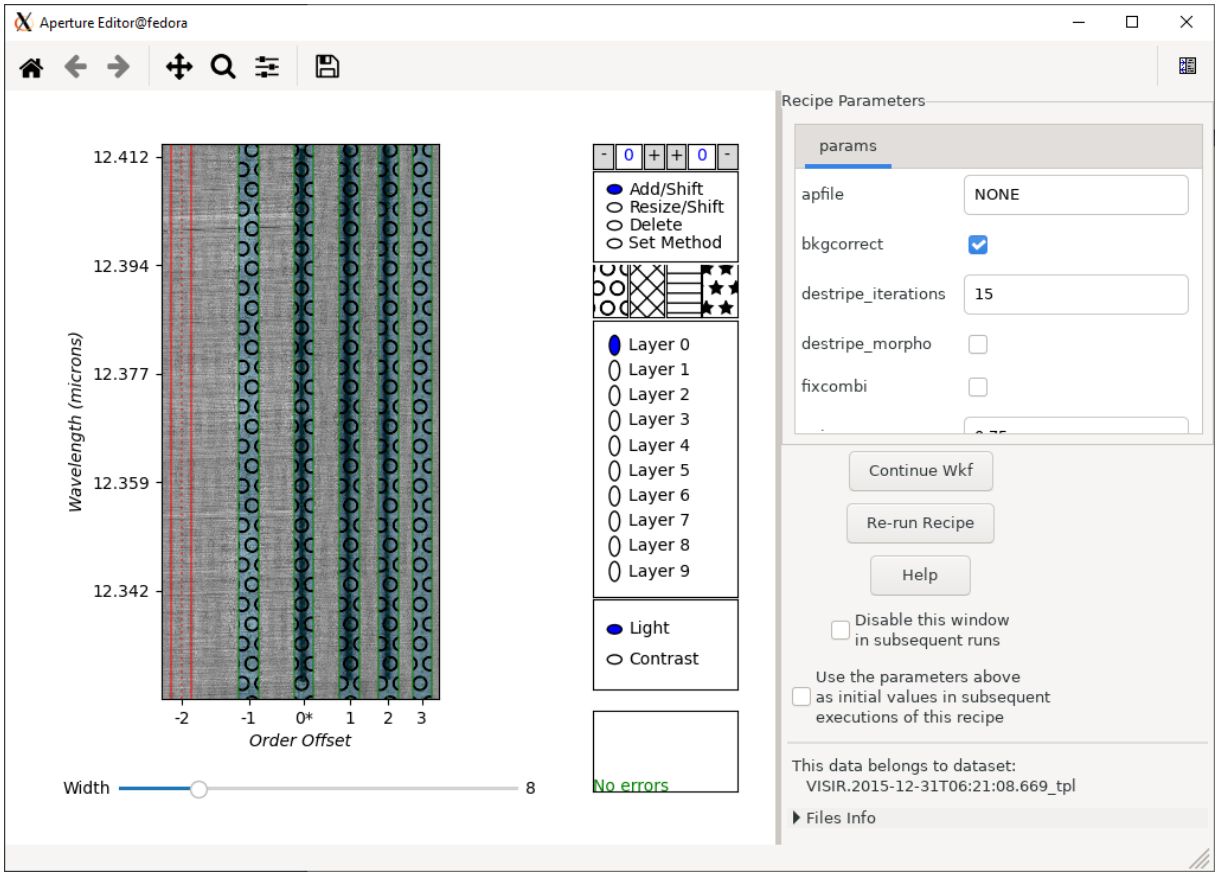


Figure 4.3.4: *Cross-dispersed mode.*

5 Known Problems

The known problems of the VISIR pipeline version 4.4.5 are:

- On *MacOS* and under *Gasgano* the parallelization of the recipes does not work. Processing on these platforms can be significantly slower than on *Linux*.
- The long slit spectrum extraction assumes that all beams fall on the detector.
- The spectral calibration does not take the drift of the high resolution spectroscopy scanner into account.
- In a few cases the Strehl ratio exceeds 1.
- The linearity correction is not well determined and will degrade low signal count data.
- For the Aquarius detector in low resolution mode the wavelength calibration has not yet been optimized and an error of about 5 pixels or $0.05\mu m$ must be assumed.

6 Instrument Data Description

VISIR data uses the FITS format and can be separated into *raw* frames and *product* frames. Raw frames are the unprocessed output of the VISIR instrument observations, while product frames are the result of the VISIR pipeline processing. In addition the VISIR pipeline uses a set of calibration (FITS-) files (standard stars catalogs, detector characteristics, etc.).

Any raw or product frame can be classified on the basis of a set of keywords read from its header. Data classification is typically carried out by the DO or by *Gasgano* [6], that apply the same set of classification rules. The association of a raw frame with calibration data ($e.g.$, of a science frame with a standard star catalogue) can be obtained by matching the values of a different set of header keywords.

Each kind of *raw* frame is typically associated to a single VISIR pipeline recipe, *i.e.*, the recipe assigned to the reduction of that specific frame type. In the pipeline environment this recipe would be launched automatically.

In the following all raw and product VISIR data frames, that can be reduced by the VISIR pipeline version 4.4.5, are listed, together with the keywords used for their classification and correct association. The indicated *DO category* is a label assigned to any data type after it has been classified, which is then used to identify the frames listed in the *Set of Frames* (see section 4.2.2 on page 18).

Raw frames can be classified as *imaging* frames or *spectroscopy* frames. Their intended use is implicitly defined by the assigned recipe.

6.1 General Data Layout - Aquarius detector

6.1.1 Half-Cycle mode

A raw VISIR file for the regular half cycle image mode is a FITS-file containing several extensions. The primary header contains only information about the observation, like the date, target, detector integration time, the number of detector integration etc. The image data is contained in FITS extensions following the primary header. Each image data extension contains its own header which contains only information which is specific to this extension. The type of image contained in the extension is defined in the `ESO DET FRAM TYPE` key. The main data has the type `HCYCLE1` or `HCYCLE2` which means it are half cycle exposures, containing the averaged integrations of one chop cycle. Depending on the observation settings the frames can also contain the average of all half cycles of one type.
Typically the last extension in the file contains the total integration image with the type `INT`. This image contains the average of all half cycles, were the on and off chop states are subtracted. This usually allows one to see the observed object as the subtraction removes most of the background.

6.1.2 Burst mode

In burst mode no averaging of integration images is done and each integration is stored in the FITS file. In this case it contains all images in one extension stored as a three dimensional cube. The cube can be in located in the primary extension together with the primary header or it can be stored in the extension following the primary header, depending on the mode in which the data was taken. Due to limitations in the FITS standard the latter is preferred if one wants to compress the file to save storage space. The lossless rice compression algorithm

is recommended for integer type burst data. Averaged burst data (as indicated by the `ESO DET NAVRG` key) typically compresses less efficiently unless one applies a lossy compression algorithm.

6.2 General Data Layout - DRS detector

A raw VISIR file is an extension-less FITS-file. The data unit is a cube with NAXIS3$= 2n + 1$ planes[7], where n is the number of chopping cycles, which is specified in the FITS-card with the key `HIERARCH ESO DET CHOP NCYCLES`. For each chopping cycle two so called Half-Cycle exposures are made, the A-image from the on-source position of the chopper, and the B-image from the off-source position of the chopper. Each half-cycle image is normalized by IRACE to an exposure time of one DIT; in other words, each half-cycle image is the average over the NDIT individual exposures. For each chopping cycle two planes are stored in the cube. The first two planes correspond to the first chopping cycle and contain:

- The Half-Cycle A-image, A_1. The pixel-values in each Half-Cycle image are offset by -32768, i.e. 32768 has to be added to each pixel in order to obtain the physical pixel value.
- The difference between the two Half-Cycle images, A_1-B_1.

Similarly, the $(2 \times i - 1)$th and $(2 \times i)$th planes correspond to the ith chopping cycle and contain

- The Half-Cycle A-image, A_i, stored with an offset identical to A_1.
- The average of the current and all previous Half-Cycle difference images, (A_1-B_1 + A_2-B_2 + ... + A_i-B_i)/i.

The last plane of the cube contains the average of all Half-Cycle difference images, i.e. it is identical to the $(2 \times n)$th plane.

6.3 General frames

These are data that can be obtained using any of the two instrument modes (imaging, spectroscopy), as is the case for flat field exposures. The keyword `ESO INS MODE` is set accordingly to 'IMG' for imaging frames, and to 'SPC' for spectroscopy frames, to indicate the intended use for the data.

- **Dark image:**
  Processed by: `visir_img_dark`
  Association keywords: `INSTRUME = VISIR`

  Classification:

DPR.CATG	DPR.TYPE	DPR.TECH	DO Category
CALIB	DARK	IMAGE	IM_CAL_DARK
CALIB	DARK	SPECTRUM	SPEC_CAL_DARK

See [5] for a definition of the values of DPR.CATG, DPR.TYPE and DPR.TECH.

[7] Before 2004-08-31 another data layout was used. The description of this now obsolete format is limited to the statement that it is also supported by the VISIR pipeline

6.4 Imaging frames

- **Science Observation:**
  Processed by: `visir_img_reduce`
  Association keywords: `INSTRUME = VISIR`

  Classification:

DPR.CATG	DPR.TYPE	DPR.TECH	DO Category
SCIENCE	OBJECT	IMAGE,CHOPNOD,JITTER	IM_OBS_CHO_NOD_JIT
SCIENCE	OBJECT	IMAGE,CHOPPING,JITTER	IM_OBS_CHO_JIT
SCIENCE	OBJECT	IMAGE,NODDING,JITTER	IM_OBS_NOD_JIT
SCIENCE	OBJECT	IMAGE,DIRECT,JITTER	IM_OBS_DIR_JIT

- **Standard Star:**
  Processed by: `visir_img_reduce`
  Association keywords: `INSTRUME = VISIR`

See [5] for a definition of the values of DPR.CATG, DPR.TYPE and DPR.TECH.

6.5 Spectroscopy frames

These frames are generated with the VISIR spectrometer.

- **Long Slit Science Observation:**
  Processed by: `visir_spc_reduce`
  Association keywords: `INSTRUME = VISIR`

  Classification:

DPR.CATG	DPR.TYPE	DPR.TECH	DO Category
SCIENCE	OBJECT	SPECTRUM,CHOPNOD	SPEC_OBS_LMR

- **Long Slit Standard Star:**
  Processed by: `visir_spc_reduce`
  Association keywords: `INSTRUME = VISIR`

  Classification:

DPR.CATG	DPR.TYPE	DPR.TECH	DO Category
CALIB	STD	SPECTRUM,CHOPNOD	SPEC_CAL_PHOT

- **Echelle Science Observation:**
  Processed by: `visir_spc_reduce`
  Association keywords: `INSTRUME = VISIR`

  Classification:

DPR.CATG	DPR.TYPE	DPR.TECH	DO Category
`SCIENCE`	`OBJECT`	`ECHELLE`	`SPEC_OBS_HRG`

- **Echelle Standard Star:**
  Processed by: `visir_spc_reduce`
  Association keywords: `INSTRUME = VISIR`

  Classification:

DPR.CATG	DPR.TYPE	DPR.TECH	DO Category
`CALIB`	`STD`	`ECHELLE`	`SPEC_CAL_PHOT_HRG`

See [5] for a definition of the values of DPR.CATG, DPR.TYPE and DPR.TECH.

7 Static Calibration Data

This section describes the input required by the VISIR pipeline in addition to the instrument data. This input data is all stored as single-extension FITS tables.

7.1 Imaging Standard Stars

The catalogue of imaging standard stars currently contains information about 426 stars. For each star the catalogue contains the following information:

- The star name, e.g. HD108903.
- Spectral type, e.g. M3.5V.
- Right ascension and declination.
- The flux $[Jy]$ for each of the 23 supported imaging filters.
- The flux $[Jy]$ for each of the 6 supported spectroscopy filters.
- The flux $[Jy]$ for each of the 3 supported coronography filters.

This catalogue is stored in the file `ima/cal/ima_std_star_cat.fits`.

See also [9].

7.2 Spectroscopy Standard Stars

The catalogue of spectroscopy standard stars currently contains information about 469 stars, namely the 425 standard stars used for imaging and an additional 44 Hipparcos standard stars. For each star the catalogue contains the following information:

- The star name, e.g. HD108903 or HIP100469.
- Right ascension and declination.
- The model flux $[W/m2/m]$ for 2300 wavelengths in the range 5 to 28 μm.

This catalogue is stored in the file `spec/cal/spec_std_star_cat.fits`.

See also [9].

7.3 Spectrometer Detector Quantum Efficiency

The spectrometer detector quantum efficiency at various wavelengths is stored in `spec/cal/spec_quantum_efficiency.fits`. The quantum efficiency ranges from about 1% to close to 70% (at 11.9μm). See figure 7.3.1. The quantum efficiency for the Aquarius detector is not yet fully characterised. Currently values provided by the manufacturer are used which may not be correct for the mode the detector is used in and does not account for any optical elements in the instrument.

See also [9].

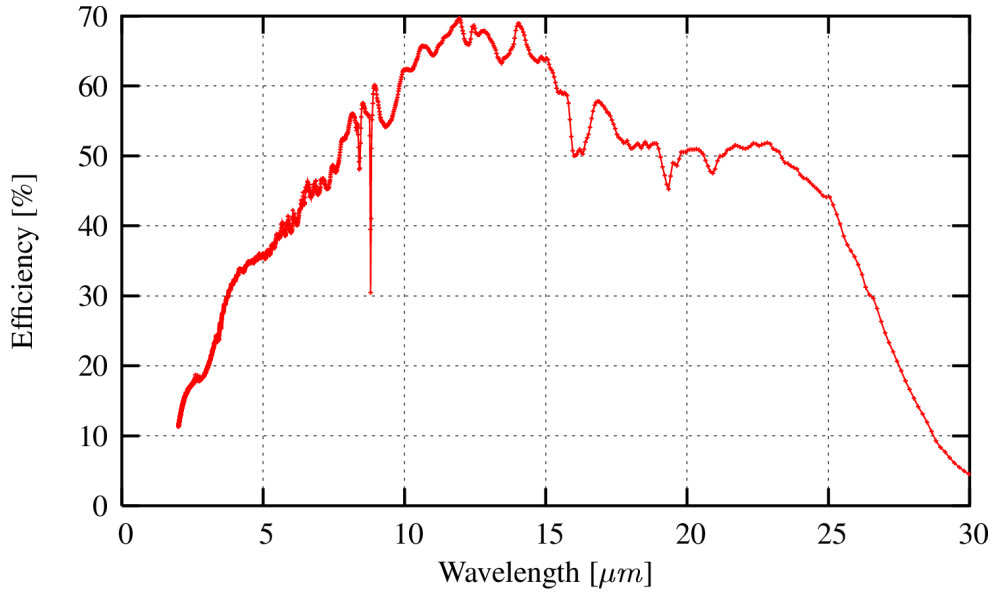


Figure 7.3.1: The DRS Detector Quantum Efficiency.

7.4 Atmospheric Emission Spectrum

The atmospheric emission spectrum (normalized to 1) has been created using the HITRAN database of molecular line parameters and the US Standard Atmosphere atmospheric profile, for an altitude of 2600m and 1.5mm of precipitable water vapor at zenith.

The atmospheric emission is stored in `spec/cal/spec_sky_lines.fits`.

See also [9].

8 Data Reduction

In this section the usage of the VISIR pipeline recipes is described.

8.1 Reduction Cascade

The reduction cascade is a schematic representation of the recipe dependencies for pipeline products. The new pipeline technically has a dependency chain between recipes many utility recipes, but in general it is recommended to use the reduce recipes which will execute the sub-recipes in the right order. The Reflex workflows shipped with the pipeline shows a graphical representation of the dependency chain.

The old pipeline recipes for the DRS data is still available, it does not contain any dependencies with the exception that the output bad-pixel map of the **visir_img_ff** recipe may optionally be used as input by the other recipes.

8.2 VISIR pipeline recipes

The new Aquarius VISIR pipeline version 4.4.5 offers a set of 2 stand-alone recipes intended for fundamental operations. The new recipes also support the old DRS data.

Imaging data reduction:

visir_img_reduce: Runs the complete imaging reduction chain. Depending on the input data and its tags it will reduce standard star calibration data or science observations. In order to do that it repacks the data (visir_util_repack), detects objects and corrects shifts (visir_util_detect_shift) and shift-and-adds the frames together (visir_util_run_swarp).

Spectroscopic data reduction:

visir_spc_reduce: Runs the complete spectroscopic reduction chain. It supports low, medium and high resolution mode. Depending on the input data and its tags it will reduce standard star calibration data or science observations. In order to do that it repacks the data (visir_util_repack), corrects distortion (visir_util_undistort) and does a wavelength calibration (visir_old_spc_obs). Notably it does **not** apply the calibration (visir_util_apply_calib).

Section 9 on page 40 gives a general description on the use of recipes, together with more detailed information on the individual recipes.

8.3 Unsupported Observation Modes

A few VISIR templates lead to observations that are currently not supported by the pipeline. Amongst the currently offered science templates, the `VISIR_img_obs_GenericChopNod` template used for raster (or mosaic) imaging, the SAM template `VISIR_img_obs_SAMAutoNod`, and also the coronographic template `VISIR_img_obs_CoroChopNod` are not supported.

8.4 Sensitivity Function Correction

VISIR spectra, especially in MR and HR data show sinusoidal variations (Fig. 8.4.1), first reported in [20]. During "normal" data processing, when the telluric correction is applied dividing the science spectrum by a telluric spectrum, these variations cancel out. However, if the telluric correction is applied with Molefit, they become a problem, because Molecfit treats them as telluric features.

Luckily, these features are reasonably stable - Fig. 8.4.2 shows spectra taken over period of many years and with different central wavelengths. Therefore, these features can be corrected: the science spectra must be divided by a sinusoidal function (normalized to unity to preserve the target flux). This function can be derived either from the science spectra, or from telluric standards, if the science spectra has too many or too broad absorption features (Fig. 8.4.3). Once the correction is applied and the science spectrum is flattened (but still contains the telluric absorption features), Molecfit can be run on that spectrum.

To facilitate this, the `visir_old_spc_obs` and `visir_spc_reduce` recipes take an optional sensitivity correction via the `--respcal` input parameter. This is a path to a FITS table containing two columns: wavelength and sensitivity correction (normalized to unity). An early version of this function with limited spectral coverage (with the correction set to unity across most of the VISIR spectroscopic wavelength range) is available in the static calibrations of the pipeline. The default for `--respcal` is NONE (e.g. no correction applied) but users may point it towards the early version shipped with the pipeline, or use it to point to a version of their own construction.

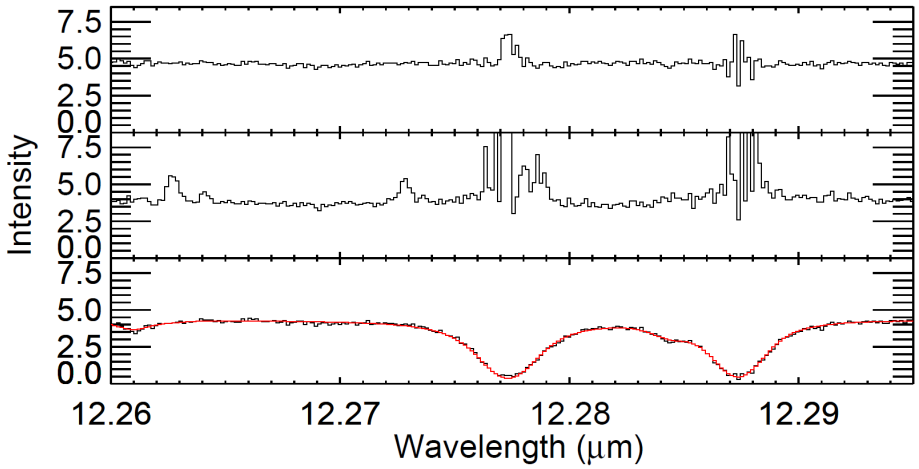


Figure 8.4.1: VISIR high-resolution λ=12.279 μm spectrum of HD 104327, reproduced from [20]. *Top:* Science spectrum divided by the transmission spectrum calculated with MOLECFIT. *Medium:* Science spectrum divided by the telluric star. *Bottom:* Pipeline reduced spectrum. In red, the result of the fit obtained with MOLECFIT.

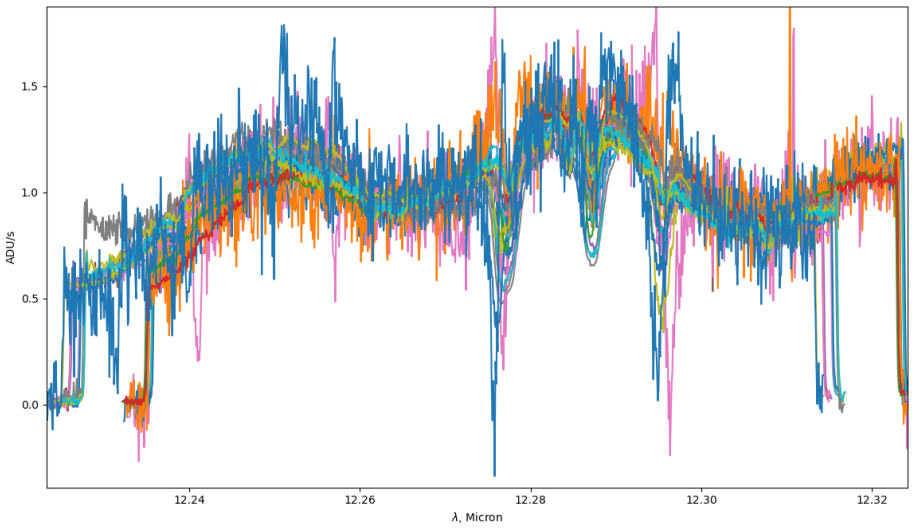


Figure 8.4.2: Spectra of multiple spectra of telluric standards in the same region, but unlike on the left panel, the fringing has not been removed and it is obvious. The spectra are normalized to the same continuum level for clarity. Top: same as on the top left panel, but for the wider wavelength range. The fringes are overlaid on the top of the smoothly varying blaze function.

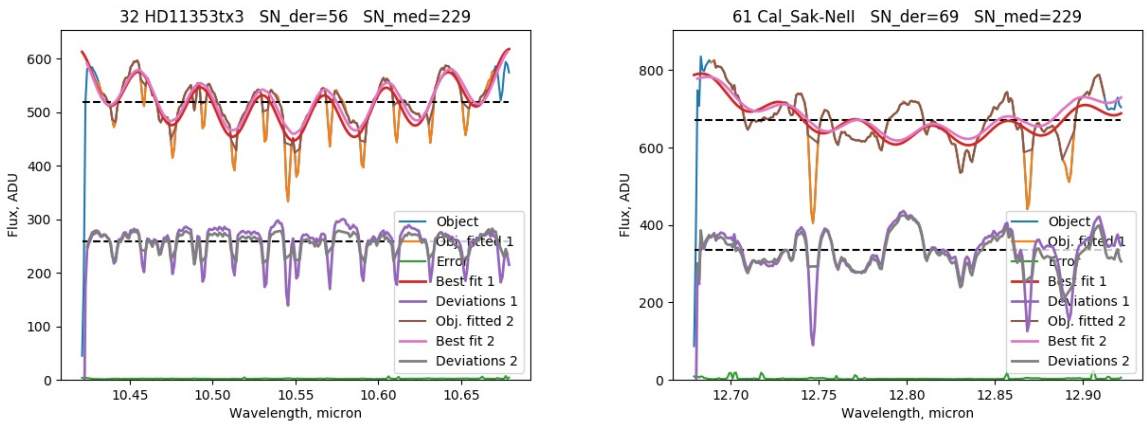


Figure 8.4.3: Two examples of fits to the sinusoidal sensitivity variations (upper curves in each panel), with different quality (in the second case strong intrinsic absorption intervene). The residuals to the fits are also shown (lower curves in each panel). Two fitting iterations were performed, with 3-σ rejection of the outliers, which makes a difference for spectra affected by deep telluric features.

9 Pipeline Recipe Interfaces

This section provides for each recipe examples of the required input data (and their classification). In the following we assume that /path_file_raw/filename_raw.fits and /path_file_cal/filename_cal.fits are existing FITS files (e.g. /data1/visir/com2/VISIR.2004-09-01T09:47:39.316.fits and /cal/visir/spec/cal/spec_quantum_efficiency.fits).

We also provide a list of the pipeline products for each recipe, indicating their default recipe name (eventually replaced by esorex to a given standard), the value of their FITS keyword HIERARCH ESO PRO CATG (in short PRO CATG) and a short description.

For each recipe we also list in a table the input parameters (as they appear in the recipe configuration file), the corresponding aliases (the corresponding names to be eventually set on command line) and their default values. Also quality control parameters are listed. Those are written in the headers of the relevant pipeline products. More information on instrument quality control can be found on *http://www.eso.org/qc*.

In addition to the products mentioned below, all recipes produce a PAF (VLT PArameter File), which is used in the ESO pipeline operations for quality control. The information in this file is the quality control data also found in the recipe products and as such this intermediate file can be ignored.

9.1 visir_img_reduce

The VISIR pipeline recipe *visir_img_reduce* combines a stack of chopped, jittered and/or nodded exposures and can compute the sensitivity for standard star observations.

9.1.1 Input files for visir_img_reduce

The input Set-Of-Frames shall specify at least one pair of files with one of the DO categories:

DO Category	Type	Explanation
IM_OBS_CHO_NOD_JIT	Raw Frame	Pairs of science Exposures

When provided with calibration data and a standard star catalog the recipe will compute the sensitivties.

DO Category	Type	Explanation
IM_CAL_PHOT	Raw Frame	Pairs of photometric calibration Exposures
or		
IM_OBS_CHO_NOD_JIT	Raw Frame	Pairs of science Exposures
IMG_STD_CATALOG	Calibration	Catalogue of imaging standard stars

Optionally a bad pixel map and a linearity table can be provided to perform the appropriate corrections on the data. The linearity correction is is currently not very accurate and likely only suitable for observations with large signal counts. On low counts the correction will severely distort the results. To also apply it to the data the parameter `--lincorrect=TRUE` must be set.

DO Category	Type	Explanation
STATIC_MASK	Calibration	Bad pixel map
LINEARITY_TABLE	Calibration	Linearity table

9.1.2 Input Parameters for visir_img_reduce

The recognized recipe options are

Parameter	Possible Values (with **default**)	Explanation
planestart	**0**	Plane number to start repacking from, earlier planes are skipped.
planelimit	**-1**	Limit number of processed input planes. It will process until at least this number of input images have been processed or the full dataset has been completed. Always full chop cycles need to be repacked so the number is adjusted upward to the next multiple of images per chop cycle. smaller-equal 0 for no limit.
trimlow	**0**	Burst data only. Number of additional planes to cut from before each plane with chopper.
trimhigh	**0**	Burst data only. Number of additional planes to cut from after each plane with chopper. A value of -1 does not skip the plane of the movement.
lincorrect	**FALSE**,TRUE	Apply linearity correction to data. May distort low signal count data.
naverage	**1**	Number of planes to average before attempting to detect the shifts.
beampos	**NONE**	Define the positions of the beams. These positions are cut by the window to extract the single beam images. Format: sign:x,y,window;sign:x,y,window;... where sign is "pos" or "neg" depending on whether the beam is positive or negative. The window is optional and defines the size of the cut image around the beam. The default window is the chop throw. E.g.: pos:50,60;neg:50,160;
output-all	**FALSE**	Output a coadded image for each input file in addition to the complete coaddition.
jy_val	**-999.**	The flux of the standard star in Jansky. Defaults to the value from the standard star catalogue
delete-temp	**TRUE**	Delete temporary files created during processing

9.1.3 Products from visir_img_reduce

Successful completion of this recipe will, depending on the input DO category, create one of these FITS-files

File name	Product Category (PRO CATG)	Explanation
visir_util_join_bCOMBINED.fits	IMG_OBS_COMBINED	the combined image (science).
visir_util_join_bCOMBINED.fits	IMG_PHOT_COMBINED	the combined image (calibration).

Additionally, the raw chop-nod corrected corrected image is produced: visir_util_repack_mean.fits See 9.4.1 on page 49 for a description of this product.

9.1.4 QC Parameters from visir_img_reduce

This recipe generates the Quality Control parameters

```
QC BACKGD MEAN
QC BACKGD SIGMA
QC EXPTIME
QC EXECTIME
QC JYVAL
QC STARNAME
QC FILTER
QC NELEC
QC NPHOT
QC FLUXTOT
QC FLUXSNR
QC FLUXSNR NOISE
QC FLUXSNR RADIUS1
QC GAUSSFIT FWHM_MAX
QC GAUSSFIT FWHM_MAX_ERR
QC GAUSSFIT FWHM_MIN
QC GAUSSFIT FWHM_MIN_ERR
QC GAUSSFIT ANGLE
QC GAUSSFIT ANGLE_ERR
QC GAUSSFIT PEAK
QC GAUSSFIT PEAK_ERR
QC FWHMX
QC FWHMY
QC SENSIT
QC AREA SENSIT
QC CONVER
QC STREHL
QC STREHL ERROR
```

and writes them in the FITS header of its products. See appendix B on page 67 for their definition.

9.2 visir_img_dark

The VISIR pipeline recipe *visir_img_dark* creates a dark image and three maps of hot, cold and deviant pixels in both imaging and spectroscopy.

9.2.1 Input files for visir_img_dark

The input Set-Of-Frames shall specify at least two files with one of the DO categories:

DO Category	Type	Explanation
IM_CAL_DARK	Raw Frame	Calibration Exposure
SPEC_CAL_DARK	Raw Frame	Calibration Exposure

9.2.2 Input Parameters for visir_img_dark

The recognized recipe options are

Parameter	Possible Values (with **default**)	Explanation
rej_bord	"%u %u %u %u", **"50 50 50 50"**	Rejected left right bottom and top border [pixel].
hot_t	`float`, **10.0**	Hot pixel map threshold.
cold_t	`float`, **6.0**	Cold pixel map threshold.
dev_t	`float`, **5.0**	Deviant pixel map threshold.
nsamples	`integer`, **100**	Number of samples for Read-Out Noise (RON) computation.
hsize	`integer`, **2**	Half size of the window for Read-Out Noise (RON) computation.

9.2.3 Products from visir_img_dark

Successful completion of this recipe will, depending on the input DO category, create one of these two sets of FITS-files

File name	Product Category (PRO CATG)	Explanation
visir_img_dark_set01_avg.fits	IMG_DARK_AVG	The dark image (imaging).
visir_img_dark_set01_hotpix.fits	IMG_DARK_HOT	The map of hot pixels (imaging).
visir_img_dark_set01_coldpix.fits	IMG_DARK_COLD	The map of cold pixels (imaging).
visir_img_dark_set01_devpix.fits	IMG_DARK_DEV	The map of deviant pixels (imaging).
visir_img_dark_set01_avg.fits	SPEC_DARK_AVG	The dark image (spectroscopy).
visir_img_dark_set01_hotpix.fits	SPEC_DARK_HOT	The map of hot pixels (spectroscopy).
visir_img_dark_set01_coldpix.fits	SPEC_DARK_COLD	The map of cold pixels (spectroscopy).
visir_img_dark_set01_devpix.fits	SPEC_DARK_DEV	The map of deviant pixels (spectroscopy).

9.2.4 QC Parameters from visir_img_dark

This recipe generates the Quality Control parameters

```
QC DARKMED
QC NBCOLPIX
QC NBHOTPIX
QC NBDEVPIX
QC RON1
QC RON2
QC RON3
...
```

and writes them in the FITS header of its products. See appendix B on page 67 for their definition.

9.3 visir_spc_reduce

The VISIR pipeline recipe *visir_spc_reduce* performs a wavelength calibration followed by spectrum extraction from a combined image. It can also compute sensitivities for standard star observations. It works for low and high resolution including echelle mode.

9.3.1 Input files for visir_spc_reduce

The input Set-Of-Frames shall specify files with DO categories:

DO Category	Type	Explanation
SPEC_OBS_LMR	Raw Frame	Science Exposures (at least one pair)
or		
SPEC_OBS_HR	Raw Frame	Science Exposures (at least one pair)
or		
SPEC_OBS_HRG	Raw Frame	Science Exposures (at least one pair)
or		
SPEC_CAL_PHOT	Raw Frame	Calibration Exposures (at least one pair)
or		
SPEC_CAL_PHOT_HRG	Raw Frame	Calibration Exposures (at least one pair)
SPEC_CAL_LINES	Calibration	Atmospheric Transmission
SPEC_CAL_QEFF	Calibration	Detector Quantum-Efficiency

Optionally a bad pixel map and a linearity table can be provided to perform the appropriate corrections on the data. The linearity correction is is currently not very accurate and likely only suitable for observations with large signal counts. On low counts the correction will severely distort the results. For this reason the correction is by default only applied to the sky spectrum used for the wavelength calibration. To also apply it to the data the parameter `--lincorrect=TRUE` must be set.

DO Category	Type	Explanation
STATIC_MASK	Calibration	Bad pixel map
LINEARITY_TABLE	Calibration	Linearity table

9.3.2 Input Parameters for visir_spc_reduce

The recognized recipe options are

Parameter	Possible Values (with **default**)	Explanation
apfile	`string`, **NONE**	An optional ASCII file specification of the aperture definitions to use during spectral extraction. Each line must contain either 3 fields for optimal extraction, or an even number of fields greater than 5 for aperture extraction. For optimal extraction, the 1st field should be 'O', and for aperture extraction it should be 'A'. In both cases the next pair of fields indicate the left & right edges of the source aperture in pixel coordinates. For aperture extraction, fields after the 3rd define the sky apertures coordinates, with the 4th field indicating the method used to determine the sky background [one of 'A' (average), 'F' (linear fit), or 'M' (median)] followed by at least one pair of pixel coordinates representing the left & right edge of a sky aperture. You may supply more sky apertures by supplying additional pixel coordinate pairs after the first. Multiple lines targeting the same order are allowed.
bkgcorrect	**TRUE**, FALSE	Subtract the median from the spectral column before extracting the wavelength. This is required when the skylines do not correctly cancel due to gratting oscillations.
delete-temp	**TRUE**, FALSE	Delete temporary files created during processing.
destripe	TRUE, **FALSE**	Attempt to remove stripes.
destripe_iterations	`integer`, **15**	Max number of destriping iterations (0 to disable destriping). Horizontal destriping is done first and if no horizontal striping is detected, vertical destriping is performed.
destripe_morpho	TRUE, **FALSE**	Destripe with morphological cleaning.
emis_tol	`float`, **1.0**	The computation of the mean and standard deviation of the sensitivity is done for wavelengths with an atmospheric emissivity of at most emis_min + emis_tol * (emis_max - emis_min), where emis_min is the minimum emissivity in the observed wavelength range and emis_max is the ditto maximum. Thus emis_tol = 1 means that all wavelengths are included.

fixcombi	TRUE, **FALSE**	Perform the distortion correction on the combined image, and not on each of the jittered images. This will reduce excution time and degrade the quality of the combined image.
gain	`float`, **0.75**	Detector gain.
hori_arc	`float`, **0.0**	Distortion correction: LMR Detector horizontal curvature (pixel). Increased by a factor 1.5 in HR A-side. Reduced by a factor 2 in HR B-side.
vert_arc	`float`, **-0.8**	Distortion correction: LR Detector vertical curvature (pixel). Reduced by a factor 4 in MR. Not used in HR A-side. Increased by a factor 115/52 in HR B-side.
lincorrect	TRUE, **FALSE**	Apply linearity correction. Should only be enabled on high flux observations, may degrade results otherwise.
ox_kernel	`integer`, **3**	Size of square smoothing kernel, in pixels, to apply to science frame before optimal extraction (ignored during aperture extraction). A median filter is used.
ox_niters	`integer`, **2**	Number of optimal extraction iterations to perform.
ox_sigma	`float`, **5.0**	Sigma to use for clipping in optimal extraction.
ox_smooth	`integer`, **31**	Width of smoothing window to use along spectral dimension during optimal extraction. A median filter is used.
planestart	`integer`, **0**	Plane number to start repacking from in each nod cycle,earlier planes are skipped.
planelimit	`integer`, **-1**	Limit number of processed inputplanes. It will repack until at least this number of input images have been processed or the full dataset has been repacked. Always full chop cycles need to be repacked so the number is adjusted upward to the next multiple of images per chop cycle. <= 0 for no limit.
plot	`integer`, **0**	The recipe can produce a number of predefined plots. Zero means that none of the plots are produced, while increasing values (e.g. 1 or 2) increases the number of plots produced. If the plotting fails a warning is produced, and the recipe continues. The default behaviour of the plotting is to use gnuplot (with option -persist). The recipe currently produces 1D-plots using gnuplot commands. The recipe user can control the actual plotting-command used by the recipe to create the plot by setting the environment variable CPL_PLOTTER. Currently, if CPL_PLOTTER is set it must contain the string 'gnuplot'. Setting it to 'cat > my_gnuplot_$$.txt' causes a number of ASCII-files to be created, which each produce a plot when given as standard input to gnuplot (e.g. later or on a different computer). A finer control of the plotting options can

		be obtained by writing an executable script, e.g. my_gnuplot.pl, that executes gnuplot after setting the desired gnuplot options (e.g. set terminal pslatex color) and then setting CPL_PLOTTER to my_gnuplot.pl. The predefined plots include plotting of images. Images can be plotted not only with gnuplot, but also using the pnm format. This is controlled with the environment variable CPL_IMAGER. If CPL_IMAGER is set to a string that does not contain the word gnuplot, the recipe will generate the plot in pnm format. E.g. setting CPL_IMAGER to 'display - &' will produce a gray-scale image using the image viewer display.
rej	`string`, **0-0**	Each resulting pixel is the average of the corresponding (interpolated) pixel value in each jittered image. A positive value, n1, for the first of the two integers specifies that for each pixel the smallest n1 pixel values shall be ignored in the averaging. Similarly, a positive value, n2, for the second of the two integers specifies that for each pixel the largest n2 pixel values shall be ignored in the averaging.
respcal	`string`, **NONE**	An optional path to a FITS file containing a 1-D fringe model to be divided into the 1-D extracted spectra in order to remove the fringes.
rl	`integer`, **0**	Reject leftmost columns in spectrum extraction, zero means all columns on the left are used. In cross-dispersion mode a (small) negative number may be used (pixel).
rr	`integer`, **0**	Reject rightmost columns in spectrum extraction, zero means all columns on the right are used. In cross-dispersion mode a (small) negative number may be used (pixel).
ro_noise	`float`, **14.5**	Readout noise of the detector.
slit_skew	`float`, **0.52**	Distortion correction: Skew of slit (degrees) (clockwise).
spectrum_skew	`float`, **1.73**	Distortion correction: LMR Skew of spectrum (degrees) (counter-clockwise). Not used in High Resolution.
trimlow	`integer`, **0**	Burst data only. Number of additional planes to cut from before each plane with chopper movement.
trimhigh	`integer`, **0**	Burst data only. Number of additional planes to cut from after each plane with chopper movement. A value of -1 does not skip the plane of the movement.
xl	`integer`, **117**	Coordinate in spatial direction. Together with yl it defines the lower point of a rectangle containing only skylines for the wavelength shift detection.
yl	`integer`, **110**	Coordinate in wavelength direction. See xl.

xh	`integer`, **125**	Coordinate in spatial direction. Together with yl it defines the higher point of a rectangle containing only skylines for the wavelength shift detection.
yh	`integer`, **150**	Coordinate in wavelength direction. See xh.

Please see 10.1.2 on page 53 for a description of the distortion correction. The default values for the two parameters of the curvature correction are set to values that are optimal in low resolution spectroscopy. High resolution is currently not corrected for curvature.

9.3.3 Products from visir_spc_reduce

Successful completion of this recipe will create the FITS-file

File name	Product Category (PRO CATG)	Explanation
visir_spc_obs_tab.fits	SPC_PHOT_TAB	the spectral table.
or		
visir_spc_obs_tab.fits	SPC_OBS_LMR_TAB	the spectral table.

See 9.4.2 on the next page for a description of this product.

9.3.4 QC Parameters from visir_spc_reduce

This recipe generates the Quality Control parameters

```
QC BACKGD MEAN
QC XC
QC XCSHIFT
QC PHDEGREE
QC PHDISPX0
QC PHDISPX1
QC PHDISPX2
QC XCWLEN
QC XCDEGREE
QC XCDISPX0
QC XCDISPX1
QC XCDISPX2
QC GAUSSFIT FWHM
QC GAUSSFIT FWHM_ERR
QC GAUSSFIT PEAK
QC GAUSSFIT PEAK_ERR
QC XFWHM
QC XCENTROI
```

```
QC EXPTIME
QC EXECTIME
QC SENS MEDIAN
QC SENS MEAN
QC SENS STDEV
QC STARNAME
```

and writes them in the FITS header of its products. Some keys may not be written depending on whether the input were calibration or science frames. See appendix B on page 67 for their definition.

9.4 Product Description

9.4.1 Imaging product data

This product has one of these product categories: `IMG_OBS_COMBINED`, `IMG_PHOT_COMBINED`

The file contains data in its primary data unit and three extensions. The primary data unit contains the combined image the combination of a list of chopped, nodded and optionally jittered images. See also subsection 10.1.3 on page 53.

The first extension with extension name `BAD PIXEL MAP` contains the pixels which must be considered bad for the data in the combined image. Note that due to the chop-nodding data the bad pixel map may be empty even when the input did have a static bad pixel map as a bad pixel in one chop cycle can be corrected by the data from other cycles which have a different throw offset.

The second extension with extension name `ERROR MAP` contains the errors associated to each pixel of the data image. Note that the error is determined from the sky background of the chop-nod corrected images and thus can be a constant value if there are no bad pixels. Pixels which marked as bad in the bad pixel map will have infinite value.

The third extension with extension name `WEIGHT MAP` contains the weights associated to each pixel of the data image. It is computed from the error map via: $\omega = \frac{1}{\sigma^2}$

9.4.2 Spectroscopic product data

This product has one of these following product categories: `SPC_OBS_LMR_TAB`, `SPC_OBS_HRG_TAB`, `SPC_PHOT_TAB` or `SPC_PHOT_HRG_TAB`

The file contains three extensions for each extracted 1D spectrum: the first extension is a binary table with the extracted spectra, sky model and sensitivities for photometric data. The second extension contains the 2d undistorted and chop-nod corrected images and the third extension contains the 1d spectral extraction weights extended to a 2d image along the wavelength axis (note: this is not related to the errors). If one aperture was defined, there will be only one set of these three aperture-related extensions; if ten apertures were defined, there will be ten sets, or 30 aperture-related extensions.

The file additionally contains one more extension, the last, which is the full 2D combined spectrum.

Each table extension contains 6 to 10 columns and one row per detector pixel in wavelength direction. The last four columns are only present for flux-calibrated data.

WLEN The wavelength of the light detected on that detector row [m].

SPC_MODEL_PH The intensity of the model spectrum at the wavelength that the physical model predicts will be detected on that detector row. The displacement (in pixels) between the two model spectra is written in the FITS card with the key `HIERARCH ESO QC XCSHIFT` [$Jradian\, m^{-3}s^{-1}$].

SPC_MODEL_XC The intensity of the model spectrum at the wavelength in column `WLEN` [$Jradian\, m^{-3}s^{-1}$] shifted by the displacement.

SPC_SKY The intensity of the extracted sky spectrum at the wavelength in column `WLEN` [$ADU\, s^{-1}$]

SPC_EXTRACTED The intensity of the extracted (object) spectrum at the wavelength in column `WLEN` [$ADU\, s^{-1}$].

SPC_ERROR The error (noise per pixel) on the extracted intensity at the wavelength in column `WLEN` [$ADU\, s^{-1}$].

STD_STAR_MODEL The flux of the standard star at the wavelength in column `WLEN` [mJy].

SENSITIVITY The sensitivity at the wavelength in column `WLEN` [mJy].

STD_STAR_MODEL The flux of the standard star at the wavelength in column `WLEN` [mJy].

SPC_CALIBRATED The intensity of the extracted (object) spectrum at the wavelength in column `WLEN`, flux-calibrated using a standard star. The units are microns and Jy, for the wavelength and the flux, respectively.

SPC_CALIBRATED_ERROR The error (noise per pixel) on the extracted intensity at the wavelength in column `WLEN`, flux-calibrated using a standard star.

The number of extensions and their sizes may be different, but meaning, units, columns, etc. are the same as with the previous pipeline version.

An example for a product is shown in figure 9.4.1. The displayed image is the full 2D combined spectrum, and the FITS table is 1D extracted spectrum from the first aperture. Figure 9.4.2 shows some extracted spectra from different orders.

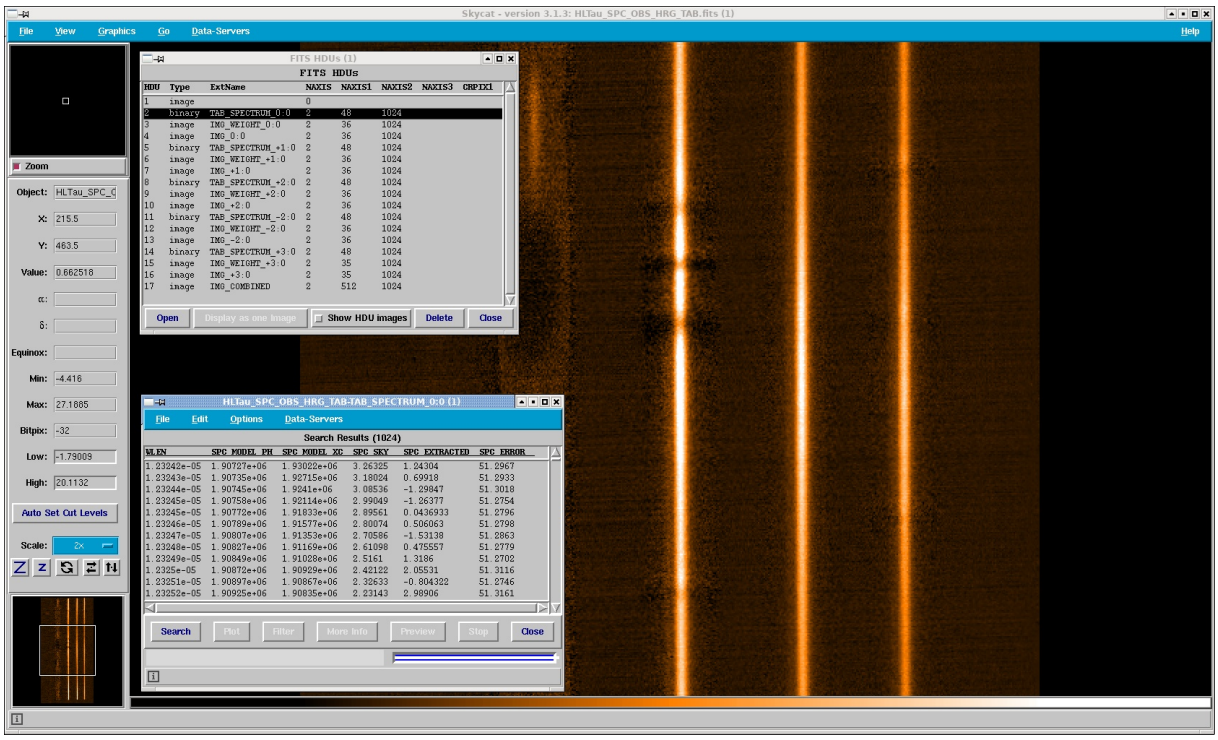


Figure 9.4.1: *A view of some of the spectroscopic products: the combined 2D spectrum is displayed in the background, the list of extensions is visible in the upper overlay, and the content of the FITS table with the extracted spectrum is in the lower overlay.*

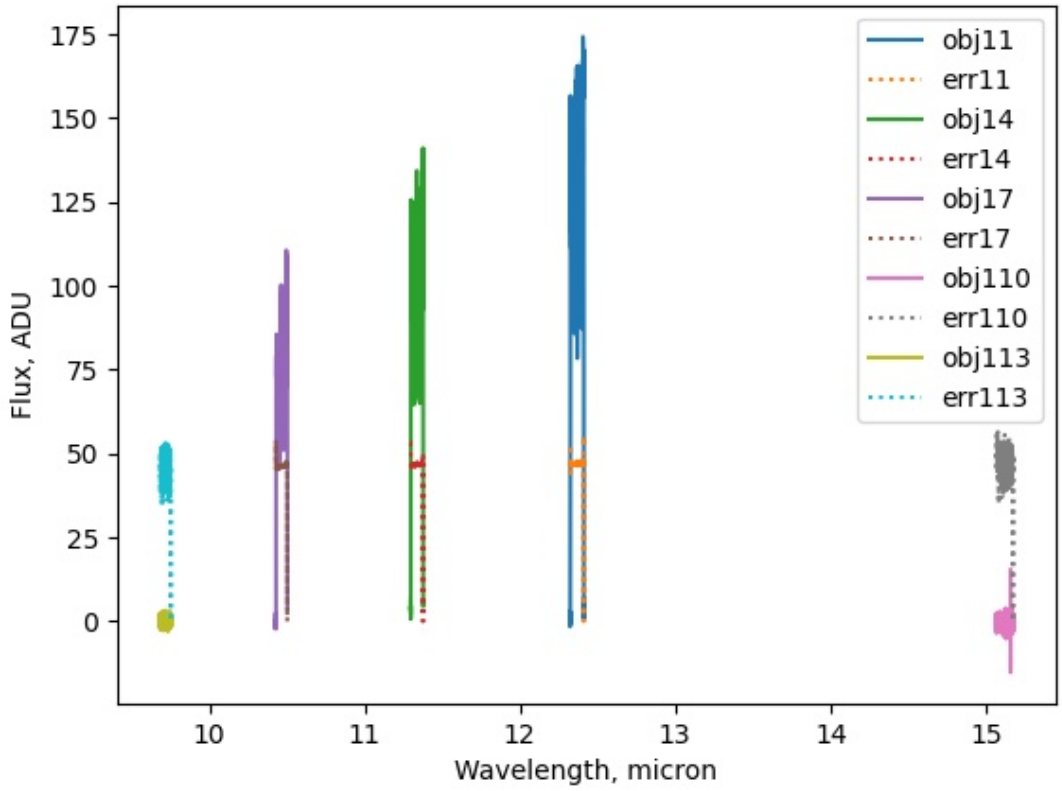


Figure 9.4.2: *Spectra from multi-order extraction. The missing order falls in a wavelength region with low atmospheric transmission.*

10 Algorithms

In this section the data reduction procedures applied by the pipeline recipes currently in use (see section 8.2 on page 37) are described in some detail.

10.1 General Algorithms

Several algorithms, such as wavelength calibration or bad pixel cleaning, are used by several recipes, and are thus described separately.

10.1.1 Bad pixel detection and cleaning

The bad pixels are detected in the Half-Cycle frames as those whose pixel value exceeds the fixed limit 65000 (this comparison is done after the offset of 32768 has been added to the pixels of the Half-Cycle frames). In spectroscopic long slit mode the cleaning of bad pixels can be avoided, since the subsequent distortion correction ignores the bad pixels. Otherwise, bad pixels are cleaned by interpolation with the neighboring pixels, (using the CPL function *cpl_detector_interpolate_rejected()*, see [12]).

10.1.2 Distortion correction

In spectroscopic long slit mode the optical distortion is known analytically. This is used to directly correct the distortion, by interpolating the distortion corrected pixel value from the source pixels. This interpolation ignores source pixels that are marked as bad. The VISIR User's manual [9] describes the optical distortion correction in greater detail. In that description, Φ is equal to slit_skew, Ψ is equal to spectrum_skew, Δ is equal to hori_arc and ϵ is equal to vert_arc. The interpolation itself is done with using the CPL function *cpl_image_get_interpolated()*, see [12].

10.1.3 Creation of combined image

The final combined image is created from the input images via the process depicted in 10.1.1 on the next page.

- The raw data is unpacked into the four different states, two chopping states, called on and off state, and the two nodding states, named A and B. These states are then subtracted in order to obtain the background corrected images:

  $$A_{on} - A_{off} - (B_{on} - B_{off})$$

  The chop/nod corrected images thus contain four or three beams depending on whether perpendicular or parallel mode is used.

- The average of all the background corrected data is used to determine the initial beam positions. Then the image is cut around these positions with a window of the chop throw in pixel. The sign of the images

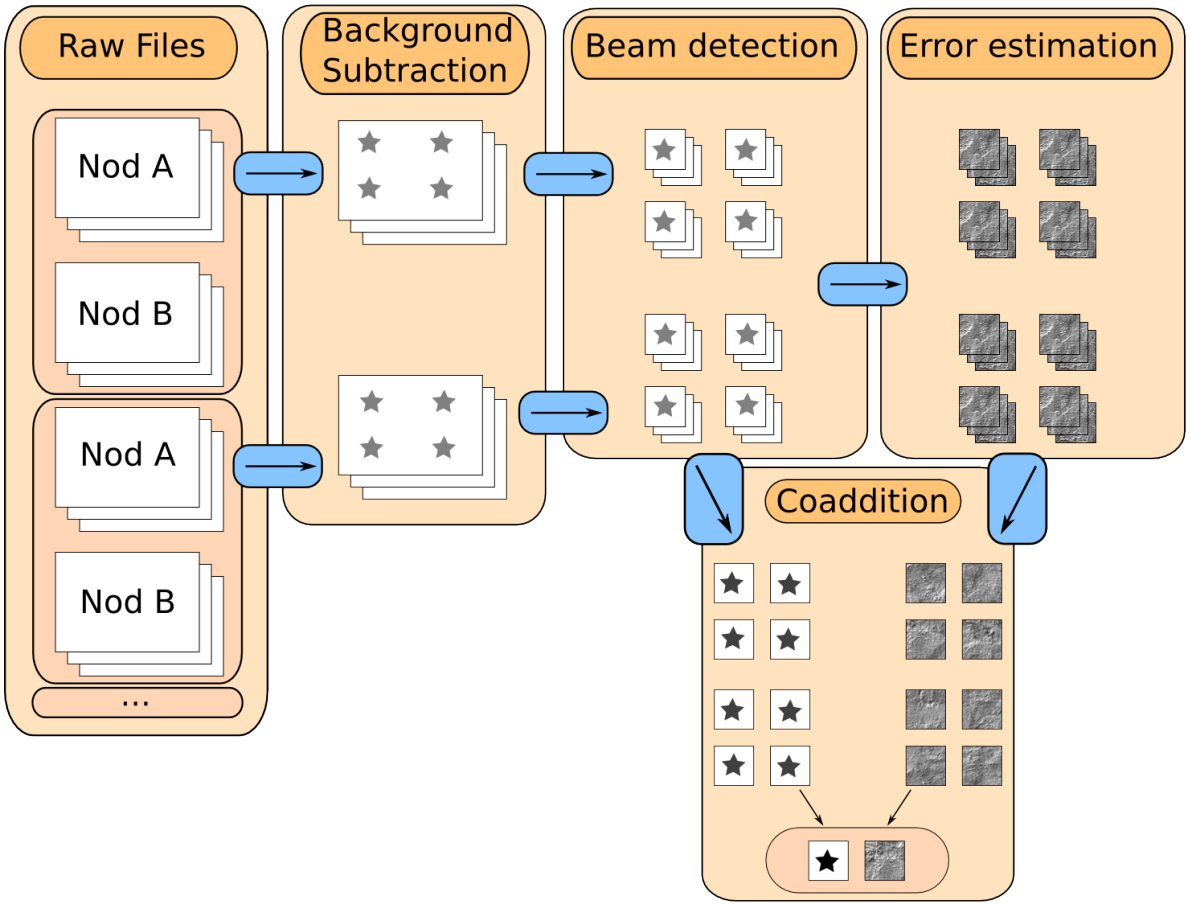


Figure 10.1.1: *The imaging combination process.*

originating from the subtractions is corrected so one receives four or three images containing only one beam with a positive sign.

- To correct for fluctuations in the object positions between exposures two methods are available. The simpler one suitable, for bright point-like objects, is to determine the brightest pixel of each image and apply an appropriate shift so all brightest pixels are on the same position.
  For objects without a well defined brightest pixel cross-correlation against a template is used. The template can be provided by the user or generated automatically from the data. The automatic generation uses the median of the first 200 images as a first guess. This guess is then correlated against the same set of images to obtain a better template. The generated template is correlated against each image in the complete dataset to obtain the final shifts in the objects to each other. The shifts are recorded in the WCS coordinates of the FITS header, no resampling of images is done at this stage yet. Additionally the correlation factor to the template can be used as a threshold to reject bad exposures.

- The noise estimation method depends on the purpose of the data. If it is calibration data used to determine

the sensitivities, the noise must be estimated from the background of the image.
This is estimated by assuming the background noise has a gaussian distribution. The median absolute deviation is calculated and multiplied with 1.4826 to obtain an estimate for the standard deviation. The median absolute deviation is used so one is only sensitive to the background fluctuations and not the object.
In science measurements one is interested in the actual uncertainty of the object flux. In order to estimate the standard deviation of each pixel in the time series of each pixel in the images is calculated. The accuracy of this method depends on the number of exposures available. For observations with very few exposures the uncertainties generated by the pipeline should not be trusted. It is very well suited for burst mode observations were the number of frames is high.

- Now that the offset between each single beam image and its uncertainties is known, the images are combined via subpixel precision shift-and-add coaddition.
  This is done in two steps. First each image is resampled to the same projection with a lancosz3 kernel.
  Each beam state (chop on/off, nod A/B) is then averaged separately. It works on per input file basis so if you input four raw files created in perpendicular mode [8] you receive eight averaged images and their weight maps, one per beam per input rawfile.
  These images are then summed to receive the final flux of the object.

10.1.4 Imaging Photometric Calibration

The photometric calibration is carried out on the sum of all combined beam files.
Each pixel is scaled to the same contribution to prevent bad pixels present in the object aperture from influencing the result too much. It is recommended to position the object so the number of bad pixels around the object is minimal. If that cannot be prevented a jittered observation mode should be chosen.

- The background flux, F_{bg} is estimated as the median intensity of the pixels that are between R and $R+10$ pixels away from the star center, where $R = 20$.

- The flux of the star, $F(r)$ is computed as the flux within r pixels from the star corrected for F_{bg}, for $r = 1, 2, 3, ..., R$

- The uncertainty of the flux is the propagated background pixel error obtained from the chop-nod corrected images.

$$\sigma\left(F(r)\right) = \sqrt{\sum_{i=1}^{n_{pix}} \sigma_{\mathrm{pix}}^2}$$

  In the general case this will be almost equal to the background pixel error of the summation image scaled by the square root of number of pixels in the flux aperture but more precise as the error is not obtained from a resampled image.

[8] Each nodding position creates one rawfile so if you record four nodding positions you obtain four raw files, each containing two chop states.

- The radius that maximizes the signal to noise ratio

$$\text{SNR}_{max} = \max_{r=1}^{R} \frac{F(r)}{\sigma\left(F(r)\right)}$$

  is determined.

- The catalog contains a model flux, F_{model}, in unit Jy.

- The sensitivity is then computed in unit mJy at 10σ in 1 hour as

$$10^3 \cdot F_{\text{model}} \cdot \frac{10}{\text{SNR}_{\text{max}}} \cdot \sqrt{t/3600s}$$

  where the exposure time t depends on the type of image.

  The total flux is also computed as:

$$F_{\text{total}} = F(R)$$

  The conversion factor is the reciprocal this value.

10.1.5 Wavelength Calibration

The dispersion relation is approximated by a second degree polynomial, $\lambda(i) = \lambda_0 + i \cdot a + b \cdot i^2, i = 1, 2, \cdots, 1024$, where $\lambda(i)$ is the wavelength at the center of the i'th pixel. Thus in long slit mode, $\lambda(i), i = 512.5$ is the central wavelength. For low resolution mode the pipeline uses following parameters as a first guess:

$$\lambda_0 = 6.8126\mu m \quad a = 0.0092488\mu m \quad b = -2.22766 \cdot 10^{-6}\mu m$$

For high resolution mode the physical model described in *Parameters for setting the VISIR Spectrometer* (VLT-TRE-VIS-14321-5046) includes as dispersion relation a first degree polynomial $\lambda_{ph}(i) = i \cdot \Delta\lambda_{ph} + \lambda_{ph,0}$. For the Aquarius detector $\Delta\lambda_{ph}$ needs to be increased by the ratio in pixel field of view between the old and new detector: $0.127/0.0757$

The parameter λ_0 is optimized to the data in the following way:

1 The field direction in a Half-Cycle frame is collapsed, producing a 1-dimensional spectrum of the atmosphere. See figure 10.1.2 on page 61.

2 A model spectrum is created from a model of the atmospheric emission. See figure 10.1.2 on page 61.

3 The offset, Δi, (in pixels) that maximizes the cross-correlation between these two spectra is used to determine λ_0, since $\lambda_0 - \lambda_{ph,0} = \Delta i \cdot \Delta\lambda_{ph}$. Δi is determined with subpixel accuracy. See figure 10.1.3 on page 62.

For the Aquarius detector in low resolution mode the wavelength calibration has not yet been optimized and an error of about 5 pixels or $0.05\mu m$ must be assumed.

The model spectrum is created as follows:

- The atmospheric emission is assumed to be equivalent to that of a black body at 253 K.
- This emission is multiplied with the emissivity of the atmosphere, see section 7.4 on page 36.
- This is smoothed by convolution with a function that is itself a convolution of two functions:

  A Gaussian with
  $$\sigma = \frac{w_{\text{FWHM}}}{2\sqrt{2\ln 2}},$$
  where the spectral FWHM is defined as $w_{\text{FWHM}} = \lambda L/R$, with L being the Linear dispersion and R being the spectral Resolution as defined by the optical model. With L in unit pixel$/m$ and λ in unit m, w_{FWHM} is in unit pixel.

  A Top-hat with a width of $\frac{w_{slit}}{w_{\text{pfov}}}$, where w_{slit} is the value of the FITS-card with the key `HIERARCH ESO INS SLIT1 WID` which has unit arcseconds and where $w_{\text{pfov}} = 0.127$ arcseconds/pixel is the spectral slit width.

  See figure 10.1.4 on page 63 for an example of such a smoothing function.
- Added to that is the emission of the telescope itself, assumed to be equivalent to that of a black body with the temperature of the main mirror (retrieved from the FITS header) and with an emissivity of 0.12.
- The model spectrum is lastly multiplied by the detector quantum efficiency.

10.1.6 Spectrum Extraction

If no `--apfile` parameter is given, the spectrum is extracted from the combined image with the following optimal extraction method:

- If the spectrum is obtained in long slit mode, each row is supposed to have a mean of zero. In order to ensure this the actual mean of each row is computed and subtracted from each row.
- The standard deviation of the noise in the resulting image is estimated using an iterative σ-clipping (σ=3).
- Each flux, $F(\lambda)$, in the 1D-spectrum is computed as a weighted average of the pixels in the field direction. The weights are the same for all wavelengths, they are obtained by collapsing the spectral dimension of the 2D-spectrum and normalizing the absolute flux of this 1D-image to 1.
- A pixel with an absolute value less than σ=3 times the standard deviation of the noise is considered noise and is excluded from the weighted average. See figure 10.1.5 on page 64.
- Then for each wavelength in the extracted 1D-spectrum the noise, $\sigma(F(\lambda))$ is computed as follows:
  - The above spectrum extraction identifies for each wavelength a number of pixels in the mean-corrected image as being noise. The standard deviation of these pixels is computed.
  - For each wavelength the 2-norm of the spatial weights of the non-noisy pixels is computed.
  - $\sigma(F(\lambda))$ is the product of these two numbers. See figure 10.1.5 on page 64.

If an aperture definitions file is provided via the `--apfile` parameter, then the spectrum is extracted from the combined image and the extraction is performed as specified in the aperture definitions file.

If the aperture definitions file indicates aperture (sum) extraction, then the extraction works as follows:

- A window is extracted from the rectified image of the size defined during object detection.
- The spectrum is collapsed across the columns of the object window. The collapsing is done with a sum.

If the aperture definitions file indicates optimal extraction, then the Horne method is used, as follows (note that this differs slightly from the optimal method performed when no aperture definitions file is given):

1. A 3x3 median filtering is applied to the object window only if the number of rows of the object window is larger than 5.
2. A first extracted spectrum is computed as in aperture extraction.
3. A profile is computed as the 2-D spectrum divided by the current estimation of the extracted spectrum.
4. The profile is smoothed in spectral direction with a median kernel of 32 pixels.
5. For each pixel, a weight is computed as the profile value divided by the variance. The variance is derived from the detector noise model, using the photon noise from the source multiplied by the profile and adding the sky noise in quadrature.
6. The spectrum is extracted using a weighted sum. Steps from point 3 are repeated a predefined number of iterations (currrently 2). Due to their high variance, cosmic rays have low weights, therefore they are usually removed by this step.

10.1.7 Spectral Photometric Calibration

The spectral photometric calibration is carried out as follows:

- The model flux, $F_{\text{model}}(\lambda)$, is obtained from the standard star catalog for the wavelengths in the extracted spectrum.
- The sensitivity in unit mJy at 10σ in 1 hour is then computed for each wavelength λ as

$$\frac{F_{\text{model}}(\lambda) \cdot \sigma(F(\lambda)) \cdot 10 \cdot \sqrt{t/3600s}}{F(\lambda)},$$

where $F(\lambda)$ and $\sigma(F(\lambda))$ is the extracted intensity and its error estimate at wavelength λ (see subsection 10.1.6 on the previous page), and where the exposure time t is DIT · NDIT · NFILES · NCHOP · 2, with the factor 2 due to the Half-Cycle chopping.

10.1.8 Imaging Photometric Calibration

10.1.9 Computation of Strehl Ratio

The computation of the Strehl ratio is carried out on the combined image in a circle with a radius $R = 3$ arcseconds (around the center of the first positive star.

The recipe assumes that the extent of the star is limited by a circle with radius $R_{\text{star}} = 2$ arcseconds.

The Strehl ratio is computed with these steps:

- The background flux, F_{bg} is estimated as the flux of the pixels located between R_{star} and R whose intensities are in the 10th percentile and not in the 90th percentile.
- The flux of the star, F_{star} is computed as the flux within R_{star} corrected for F_{bg}, and $I_{\text{star,max}}$ is the peak intensity of the star.
- The ideal Point Spread Function is computed as the inverse Fourier Transform of the ideal Optical Transfer Function, which is based on the telescope and instrument characteristics. F_{psf} is the flux of the PSF and $I_{\text{psf,max}}$ is its peak intensity.
- The Strehl ratio is then
$$\frac{I_{\text{star,max}}}{F_{\text{star}}} \Big/ \frac{I_{\text{psf,max}}}{F_{\text{psf}}}.$$

The error bound on the Strehl ratio is

$$c \cdot \pi \cdot \sigma_R \cdot w_{\text{pfov}} \cdot R_{\text{star}}^2 / F_{\text{star}},$$

where $c = 0.007/0.0271$ is determined empirically, where w_{pfov} is the imaging pixel field of view (obtained from the FITS card with key `ESO INS PFOV` with unit arcseconds/pixel), and where σ_R is the estimated noise on the pixels located between R_{star} and R, (using the CPL function *cpl_flux_get_noise_ring()*, see [12]).

10.2 Recipe Algorithms

10.2.1 visir_img_ff

This is the algorithm used by this recipe:

- For each flat-field image the median (intensity) is computed.
- For each pixel on the detector, the pixel intensity is plotted against the corresponding median.
- Ideally, all pixels should have an equal gain thus the plots should be straight lines with a slope of 1. Since this is not the case, some pixels have a relative gain greater than 1 while others have a relative gain less than 1. For each pixel this relative gain is stored in the main product of the flat-field recipe.
- Pixels with a relative gain outside the range from 1/5 to 5 are flagged as bad in the produced bad-pixel map.

10.2.2 visir_img_dark

The dark image is computed as the mean of the input dark images.

10.2.3 visir_img_reduce

This is the algorithm used by this recipe:

- The image combination described in subsection 10.1.3 on page 53.

10.2.4 visir_spc_reduce

This is the algorithm used by this recipe:

- The bad pixel detection described in 10.1.1 on page 53.
- The distortion correction described in 10.1.2 on page 53.
- The wavelength calibration described in 10.1.5 on page 56.
- The image combination described in subsection 10.1.3 on page 53.
- The spectrum extraction described in subsection 10.1.6 on page 57.
- The photometric calibration described in subsection 10.1.7 on page 58.

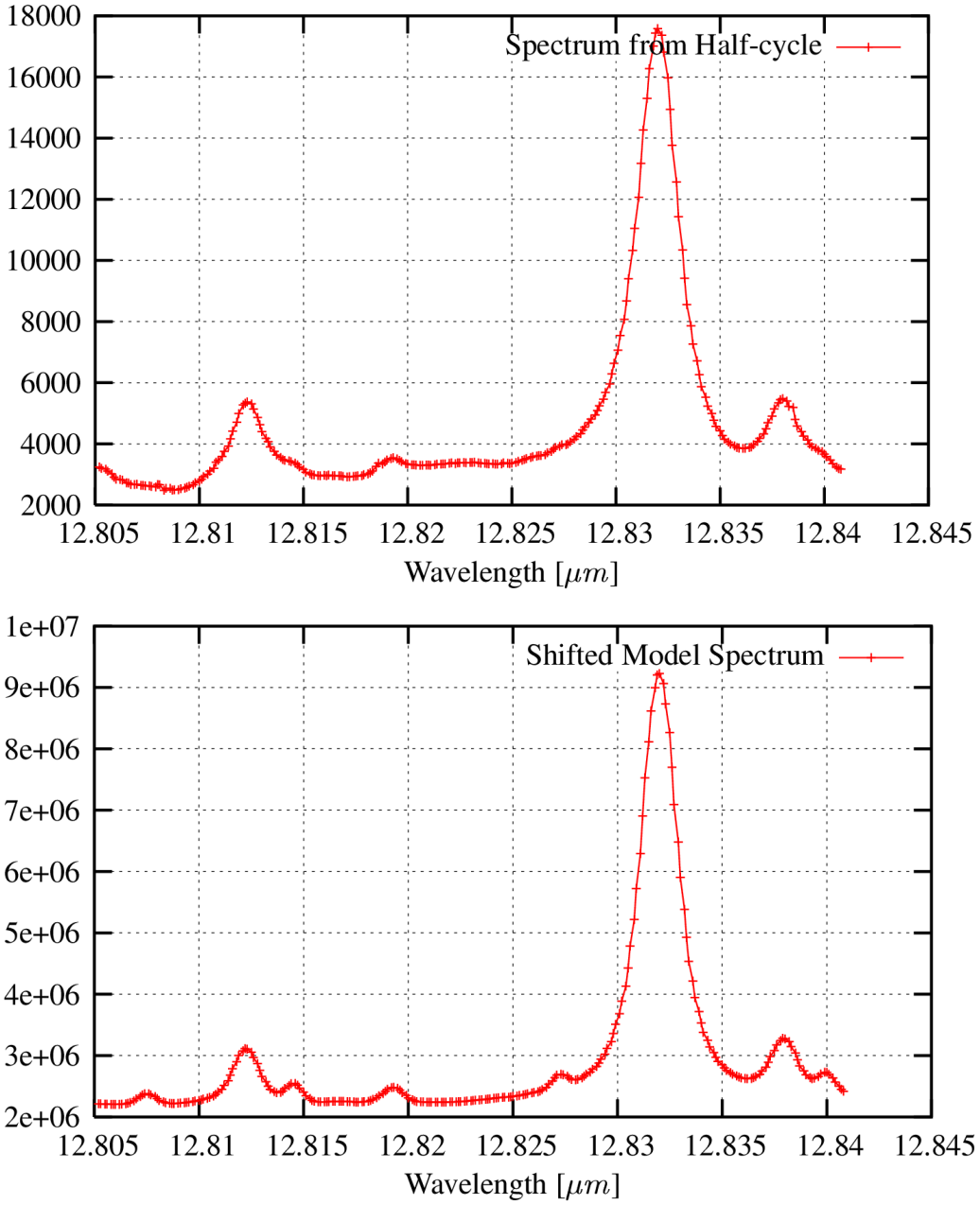


Figure 10.1.2: Example of Atmospheric spectrum from a 1/2-cycle frame and the corresponding, shifted model spectrum that maximizes the cross-correlation with that spectrum.

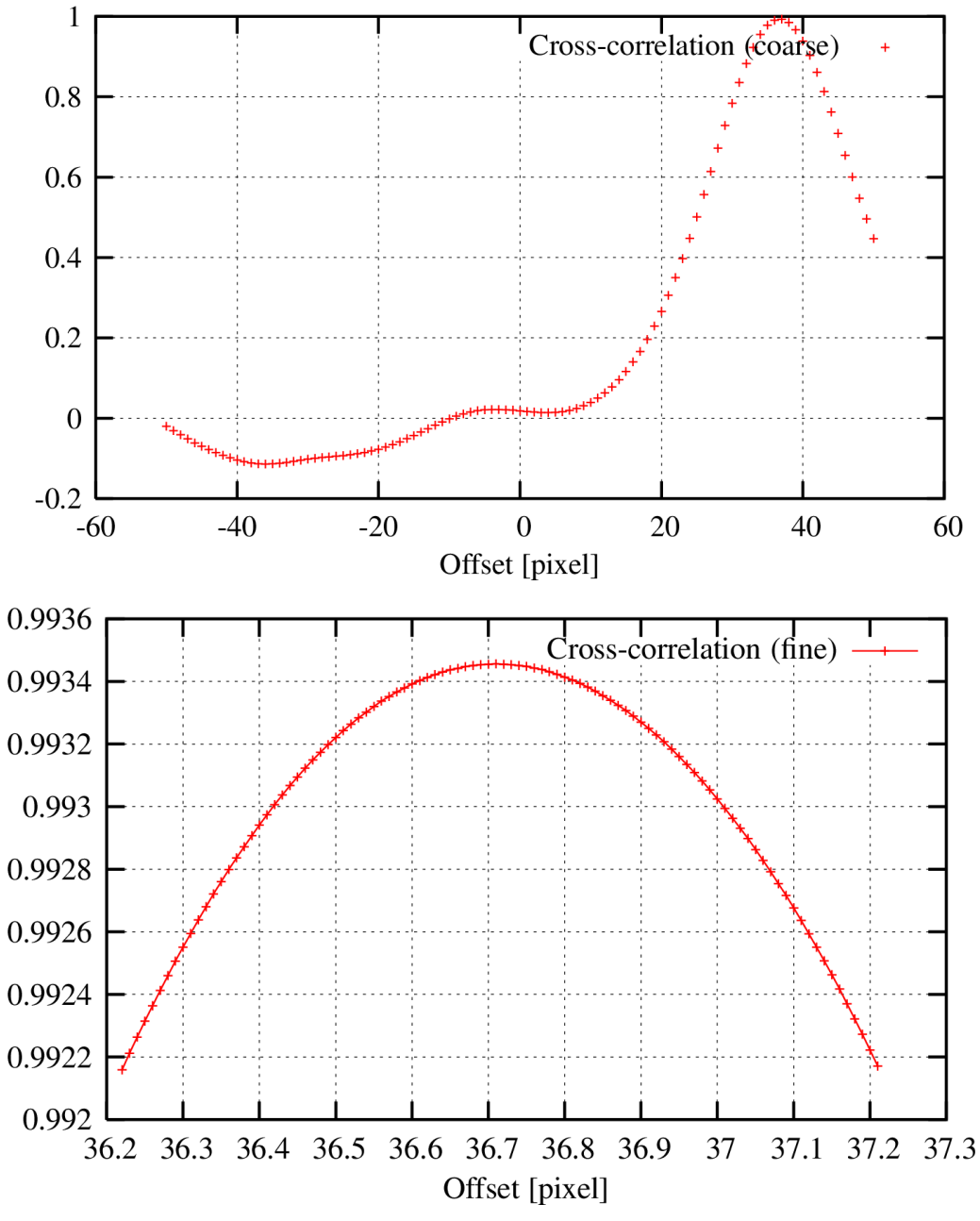


Figure 10.1.3: The cross-correlation (coarse and fine) as a function of pixel-shift.

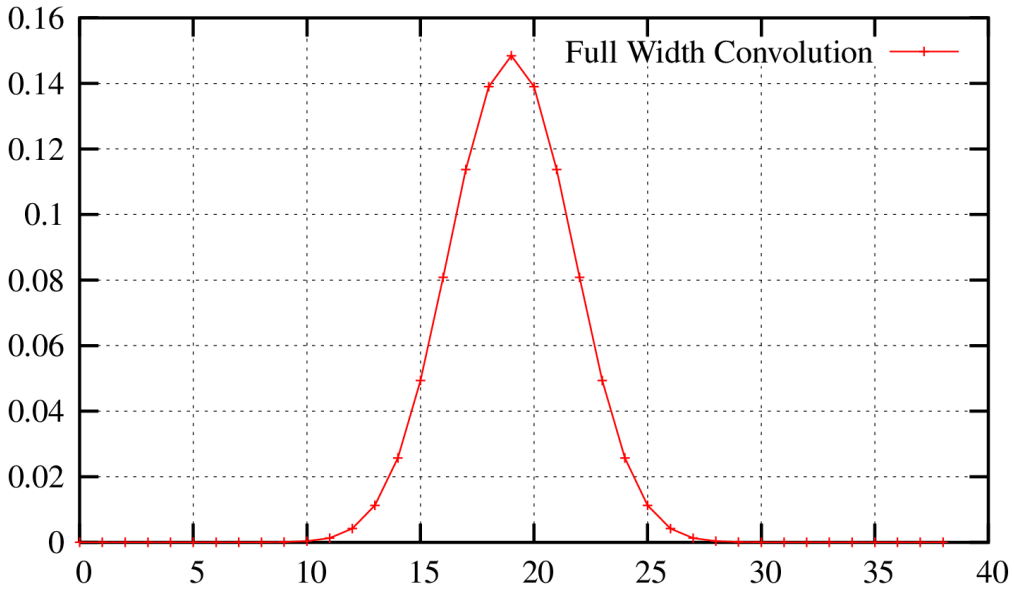


Figure 10.1.4: The symmetric convolution profile used to smooth the model spectrum. The area under the profile is 1. The instrument settings for this example is $\lambda_{central} = 12.818 \mu m$ in High Resolution Long Slit mode, with $w_{slit} = 5.9$pixel and $w_{\mathrm{FWHM}} = 5.25$pixel.

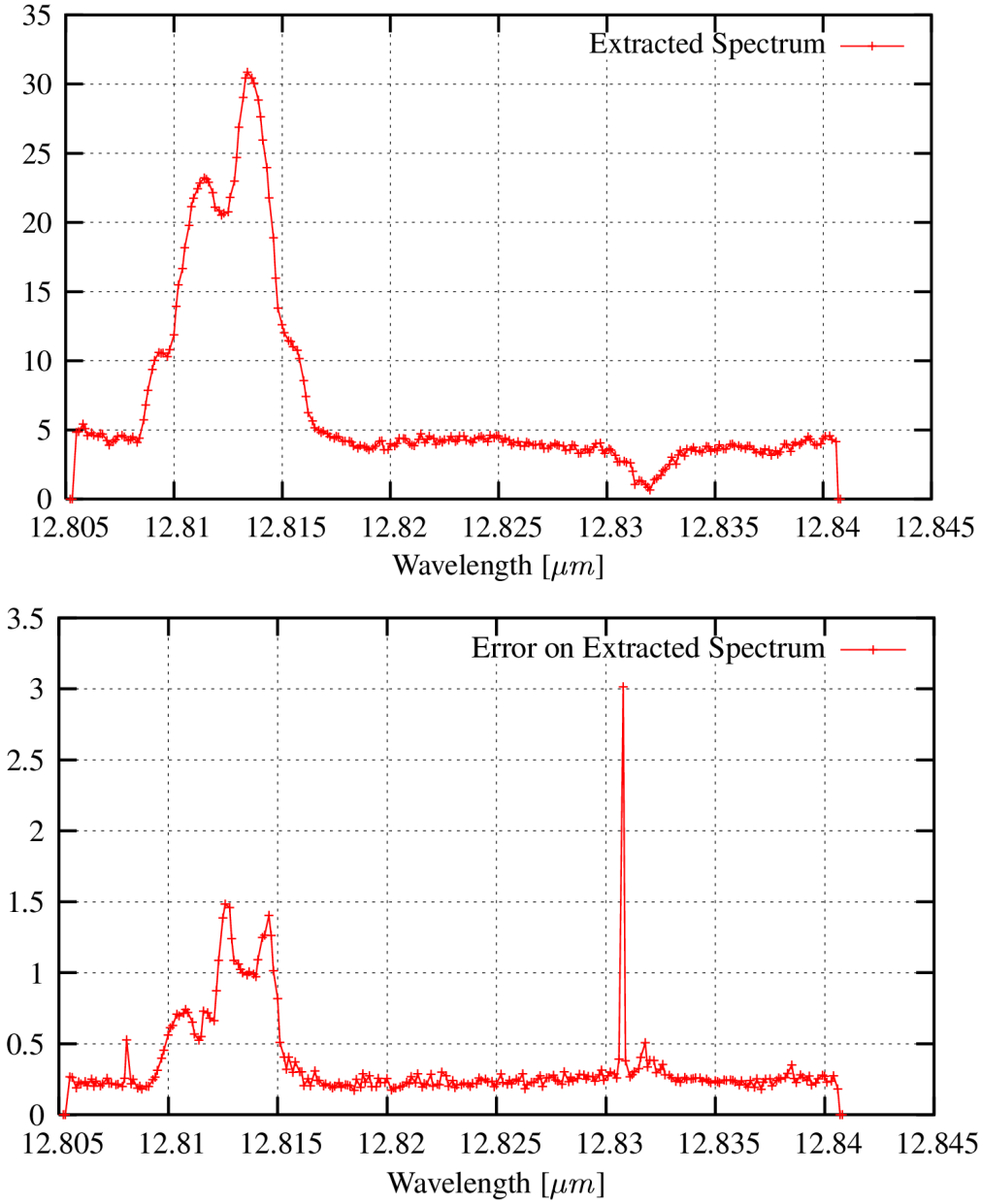


Figure 10.1.5: Example of an extracted spectrum and its error.

A Installation

This chapter gives generic instructions on how to obtain, build and install the VISIR pipeline version 4.4.5. Even if this chapter is kept as up-to-date as much as possible, it may not be fully applicable to future releases. For instructions on installation of future releases, the reader is therefore advised to check the installation instructions delivered with such future releases. The supported platforms are listed in Section A.1. It is recommended reading through Section A.2.2 before starting the installation.

A bundled version of the VISIR pipeline with all the required tools and an installer script is available from *http://www.eso.org/pipelines*, for users who are not familiar with the installation of software packages.

A.1 Supported platforms

The VISIR pipeline version 4.4.5 is verified and supported on the VLT target platforms:

- Scientific Linux 7

The usage of the GNU build tools should allow to build and run the VISIR pipeline on a variety of platforms. As such, the VISIR pipeline version 4.4.5 is known to build and to produce correct output with esorex on these platforms:

- CentOS 7
- Fedora 28 to 35
- MacOS 10.15, 11, and 12

For the most recent list of supported operation systems refer to the VLT Instrument Pipelines page.

A.2 Building the VISIR pipeline

This section shows how to obtain, build and install the VISIR pipeline from the official source distribution.

A.2.1 Requirements

To compile and install the VISIR pipeline one needs:

- one of the C compilers listed above,
- a version of the `tar` file-archiving program,
- the GNU software: `gzip`, `perl`, `make`.

On the above mentioned systems a Java Development Kit (JDK) version 1.7 will additionally allow the usage of Gasgano (part of this distribution).

A.2.2 Compiling and installing the VISIR pipeline

Here is a description of the installation procedure:

1. Change directory to where you want to retrieve the VISIR pipeline recipes 4.4.5 package. It can be any directory of your choice but not:

   ```
   $HOME/gasgano
   $HOME/.esorex
   ```

2. Download from the ESO ftp server, *http://www.eso.org/pipelines*, the latest release of the VISIR pipeline distribution.

3. Unpack using the following command:
   `tar xf visir-kit-4.4.5.tar.gz`

4. Install: after moving to the top installation directory,
   `cd visir-kit-4.4.5`

   it is possible to perform a default installation using the available installer script (*recommended*):

   ```
   ./install_pipeline
   ```

   The execution may take a few minutes.

   By default the script will install the VISIR recipes, *Gasgano*, *EsoRex*, all the necessary libraries, and the static calibration tables, into a directory tree rooted at `$HOME`. A different path may be specified as soon as the script is run.

   The only exception to all this is the *Gasgano* tool, that will always be installed under the directory `$HOME/gasgano`. Note that the installer will move an existing `$HOME/gasgano` directory to `$HOME/gasgano.old` before the new *Gasgano* version is installed.

   Important: the installation script would ensure that any existing *Gasgano* and *EsoRex* setup would be inherited into the newly installed configuration files (avoiding in this way any conflict with other installed instrument pipelines).

Alternatively, it is possible to perform a manual installation (*experienced users only*): the `README` file located in the top installation directory contains more detailed information about a step-by-step installation.

B QC Parameters

This appendix describes the QC Parameters created by the VISIR pipeline:

```
Parameter Name:     QC BACKGD MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               ADU
Comment Field:      Background level from Half-Cycle frames.
Description:        Background level from Half-Cycle frames. This number
                    does not include the offset correction.

Parameter Name:     QC BACKGD SIGMA
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               ADU
Comment Field:      Background noise.
Description:        Background noise determined with 5 iterations of sigma=3
                    clipping.

Parameter Name:     QC CAPA
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %s
Unit:
Comment Field:      The pixel capacity (large, small or problem)
Description:        The pixel capacity (large, small or problem) based on
                    DET VOLT1 DCTA9 and DET VOLT1 DCTB9 (in imaging)
                    DET VOLT2 DCTA9 and DET VOLT2 DCTB9 (in spectroscopy).
                    If the mean of DCTA9 and DCTB9 is less than 1: small
                    If the mean of DCTA9 and DCTB9 exceeds 4.5: large,
                    otherwise problem. DRS detector only.

Parameter Name:     QC CONVER
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
```

```
Unit:               ADU/Jy
Comment Field:      The conversion factor in imaging
Description:        The conversion factor in imaging

Parameter Name:     QC DARKMED
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %f
Unit:               ADU
Comment Field:      Dark current
Description:        Median of the dark produced image

Parameter Name:     QC EXPTIME
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               seconds
Comment Field:      Exposure time
Description:        Exposure time

Parameter Name:     QC EXECTIME
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               seconds
Comment Field:      Execution time
Description:        Execution time of observeration. Approximated from time
                    between start of observation (DATE-OBS) and time of output
                    file written (DATE).

Parameter Name:     QC FILTER
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %20s
Unit:
Comment Field:      The filter used to observe the star.
Description:        The filter used to observe the star.

Parameter Name:     QC FLUXSNR
Class:              header|qc-log
```

```
Context:          process
Type:             double
Value Format:     %e
Unit:             ADU
Comment Field:    Star flux obtained for the best SNR.
Description:      Star flux obtained for the best SNR.

Parameter Name:   QC FLUXSNR NOISE
Class:            header|qc-log
Context:          process
Type:             double
Value Format:     %e
Unit:             ADU
Comment Field:    Noise obtained for the best SNR.
Description:      Noise obtained for the best SNR.

Parameter Name:   QC FLUXSNR RADIUS1
Class:            header|qc-log
Context:          process
Type:             double
Value Format:     %e
Unit:             pixels
Comment Field:    Radius if best SNR appearture.
Description:      Radius if best SNR appearture.

Parameter Name:   QC FLUXTOT
Class:            header|qc-log
Context:          process
Type:             double
Value Format:     %e
Unit:             ADU
Comment Field:    Total flux of the star.
Description:      Total flux of the star.

Parameter Name:   QC FPNOISE
Class:            header|qc-log
Context:          process
Type:             double
Value Format:     %f
Unit:             ADU
Comment Field:    Fixed Pattern Noise
Description:      Largest noise component in raw images

Parameter Name:   QC GAUSSFIT FWHM_MAX
```

```
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixels
Comment Field:      FWHM of major axis of the star
Description:        The Full Width at Half Maximum of the major
                    axis computed via a 2 dimensional Gauss fit.

Parameter Name:     QC GAUSSFIT FWHM_MAX_ERR
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixels
Comment Field:      Error of FWHM of major axis of the star
Description:        The error of Full Width at Half Maximum of the major
                    axis computed via a 2 dimensional Gauss fit.

Parameter Name:     QC GAUSSFIT FWHM_MIN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixels
Comment Field:      FWHM of minor axis of the star
Description:        The Full Width at Half Maximum of the minor
                    axis computed via a 2 dimensional Gauss fit.

Parameter Name:     QC GAUSSFIT PEAK
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               ADU / s
Comment Field:      Peak of Gauss fit
Description:        Peak value of 2 dimensional Gaus fit.

Parameter Name:     QC GAUSSFIT PEAK_ERR
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               ADU / s
```

```
Comment Field:      Error of peak of Gauss fit
Description:        Error of peak value of 2 dimensional Gaus fit.

Parameter Name:     QC GAUSSFIT ANGLE
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               deg
Comment Field:      Angle between X axis and major axis
Description:        Angle between X axis and major axis of ellipse, counted
                    counterclockwise. Computed from 2d Gauss fit.

Parameter Name:     QC GAUSSFIT ANGLE_ERR
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               deg
Comment Field:      Error of angle between X axis and major axis
Description:        Error of angle between X axis and major axis of ellipse,
                    counted counterclockwise. Computed from 2d Gauss fit.

Parameter Name:     QC GAUSSFIT FWHM
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixels
Comment Field:      Median of FWHM of spectrum on the spatial axis
Description:        Median of FWHM of spectrum on the spatial axis.
                    Computed as median from FWHM of 1 dimensional Gauss fits of
                    each pixel of the wavelength axis.

Parameter Name:     QC GAUSSFIT FWHM_ERR
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixels
Comment Field:      Error of median of FWHM of spectrum on the spatial axis
Description:        Error of median of FWHM of spectrum on the spatial axis.
                    Computed as median from FWHM of 1 dimensional Gauss fits of
                    each pixel of the wavelength axis.
```

```
Parameter Name:    QC FWHMX NEG1
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in x of the first negative star
Description:       The Full Width at Half Maximum in x of
                   the first negative star. -1 if not computed.

Parameter Name:    QC FWHMX NEG2
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in x of the second negative star
Description:       The Full Width at Half Maximum in x of
                   the second negative star. -1 if not computed.

Parameter Name:    QC FWHMX POS1
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in x of the first positive star
Description:       The Full Width at Half Maximum in x of
                   the first positive star. -1 if not computed.

Parameter Name:    QC FWHMX POS2
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in x of the second positive star
Description:       The Full Width at Half Maximum in x of
                   the second positive star. -1 if not computed.

Parameter Name:    QC FWHMY NEG1
Class:             header|qc-log
Context:           process
Type:              double
```

```
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in y of the first negative star
Description:       The Full Width at Half Maximum in y of
                   the first negative star. -1 if not computed.

Parameter Name:    QC FWHMY NEG2
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in y of the second negative star
Description:       The Full Width at Half Maximum in y of
                   the second negative star. -1 if not computed.

Parameter Name:    QC FWHMY POS1
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in y of the first positive star
Description:       The Full Width at Half Maximum in y of
                   the first positive star. -1 if not computed.

Parameter Name:    QC FWHMY POS2
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              pixels
Comment Field:     FWHM in y of the second positive star
Description:       The Full Width at Half Maximum in y of
                   the second positive star. -1 if not computed.

Parameter Name:    QC JYVAL
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              Jansky
Comment Field:     Jansky value for a given star in a given band from a
                   catalog.
```

```
Description:        Jansky value for a given star in a given band from a
                    catalog.

Parameter Name:     QC LAMPFLUX
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               ADU/s
Comment Field:      The median flux of the first lampflat difference per DIT.
Description:        The median flux of the difference between the first two
                    lampflats using the central part of the detector, divided
                    by the DIT. Used to monitor the lamp evolution.

Parameter Name:     QC NBBADPIX
Class:              header|qc-log
Context:            process
Type:               integer
Value Format:       %d
Unit:               pixels
Comment Field:      Number of bad pixels.
Description:        Number of bad pixels.

Parameter Name:     QC NBCOLPIX
Class:              header|qc-log
Context:            process
Type:               integer
Value Format:       %d
Unit:               nb of pixels
Comment Field:      Number of cold pixels
Description:        Number of cold pixels

Parameter Name:     QC NBDEVPIX
Class:              header|qc-log
Context:            process
Type:               integer
Value Format:       %d
Unit:               nb of pixels
Comment Field:      Number of deviant pixels
Description:        Number of deviant pixels

Parameter Name:     QC NBHOTPIX
Class:              header|qc-log
Context:            process
```

```
Type:               integer
Value Format:       %d
Unit:               nb of pixels
Comment Field:      Number of hot pixels
Description:        Number of hot pixels

Parameter Name:     QC PHDEGREE
Class:              header|qc-log
Context:            process
Type:               integer
Value Format:       %d
Unit:
Comment Field:      The degree of the model dispersion polynomial
Description:        The degree of the dispersion polynomial from the
                    physical model. It is currently 1 for the DRS detector and 2
                    for the Aquarius detector.

Parameter Name:     QC PHDISPX0
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               meter
Comment Field:      wavelength = PHDISPX0 + i * PHDISPX1, i=1,2,...
Description:        The constant term of the dispersion polynomial from
                    the physical model, wavelength = f(pixel).

Parameter Name:     QC PHDISPX1
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               meter/pixel
Comment Field:      wavelength = PHDISPX0 + i * PHDISPX1, i=1,2,...
Description:        The linear term of the dispersion polynomial from
                    the physical model, wavelength = f(pixel).

Parameter Name:     QC PHDISPX2
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               meter/pixel
Comment Field:      wavelength = PHDISPX0 + i * PHDISPX1, i=1,2,...
```

```
Description:       The quadratic term of the dispersion polynomial from
                   the physical model, wavelength = f(pixel).

Parameter Name:    QC RONi
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %f
Unit:              ADU
Comment Field:     Read-out noise of the ith pair of the set
Description:       Measured read-out noise on the whole array.

Parameter Name:    QC SENSIT
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              mJy/10sigma/1hour
Comment Field:     The sensitivity in imaging
Description:       The sensitivity in imaging

Parameter Name:    QC AREA SENSIT
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              mJy/10sigma/1hour
Comment Field:     The sensitivity in imaging scaled to noise of 1 arcsecond
Description:       The sensitivity in imaging scaled to noise of 1 arcsecond
                   apperature.

Parameter Name:    QC SENS MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %e
Unit:              mJy
Comment Field:     Mean of the spectroscopic sensitivities
Description:       Mean of the spectroscopic sensitivities over the whole
                   spectral range

Parameter Name:    QC SENS MEDIAN
Class:             header|qc-log
Context:           process
```

```
Type:               double
Value Format:       %e
Unit:               mJy
Comment Field:      Median of the spectroscopic sensitivities
Description:        Median of the spectroscopic sensitivities over the whole
                    spectral range

Parameter Name:     QC SENS STDEV
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               mJy
Comment Field:      Standard deviation of the spectroscopic sensitivities
Description:        Standard deviation of the spectroscopic sensitivities
                    over the whole spectral range

Parameter Name:     QC STARNAME
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Unit:
Comment Field:      Standard star name
Description:        Standard star name

Parameter Name:     QC STREHL
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:
Comment Field:      The strehl ratio
Description:        The strehl ratio

Parameter Name:     QC STREHL ERROR
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:
Comment Field:      The error bound on the strehl ratio
Description:        The error bound on the strehl ratio

Parameter Name:     QC XC
```

```
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:
Comment Field:      The cross-correlation, ranging from 0 to 1.
Description:        The cross-correlation between the observed sky spectrum
                    and a model spectrum that has been shifted such that it
                    has maximal cross-correlation with the observed sky
                    spectrum. Range from 0 to 1.

Parameter Name:     QC XCDEGREE
Class:              header|qc-log
Context:            process
Type:               integer
Value Format:       %d
Unit:
Comment Field:      The degree of the calibration dispersion polynomial
Description:        The degree of the dispersion polynomial from the
                    cross-correlation. It is currently 1 for the DRS detector and 2
                    for the Aquarius detector.

Parameter Name:     QC XCDISPX0
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               meter
Comment Field:      wavelength = XCDISPX0 + i * XCDISPX1, i=1,2,...
Description:        The constant term of the dispersion polynomial from
                    the cross-correlation, wavelength = f(pixel).

Parameter Name:     QC XCDISPX1
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               meter/pixel
Comment Field:      wavelength = XCDISPX0 + i * XCDISPX1, i=1,2,...
Description:        The linear term of the dispersion polynomial from
                    the cross-correlation, wavelength = f(pixel). It
                    is currently equal to PHDISPX1.

Parameter Name:     QC XCDISPX2
```

```
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               meter/pixel
Comment Field:      wavelength = XCDISPX0 + i * XCDISPX1, i=1,2,...
Description:        The quadratic term of the dispersion polynomial from
                    the cross-correlation, wavelength = f(pixel). It
                    is currently equal to PHDISPX2.

Parameter Name:     QC XCENTROI
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixel
Comment Field:      The x-centroid of the spectrums brightest object
Description:        The location (centroid) in the field-direction of
                    the brightest object of the spectrum

Parameter Name:     QC XCSHIFT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixel
Comment Field:      The shift in pixels of the model spectrum
Description:        The shift in pixels of the model spectrum that
                    maximizes the cross-correlation between the observed
                    sky spectrum and the model spectrum. A positive number
                    means that the FITS- headers WLEN is too large.
                    The range is bound by the detector size, -256 to 256

Parameter Name:     QC XFWHM
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %e
Unit:               pixel
Comment Field:      The Full Width at Half Maximum of the object at XCENTROI
Description:        The Full Width at Half Maximum of the brightest object
                    of the spectrum, located at XCENTROI
```

C Abbreviations and acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CalibDB	Calibration Database
CPL	Common Pipeline Library
DFO	Data Flow Operations department
DFS	Data Flow System department
DMD	Data Management and Operations Division
DRS	Data Reduction System
ESO	European Southern Observatory
ESOREX	ESO-Recipe Execution tool
FITS	Flexible Image Transport System
FOV	Field Of View
FPN	Fixed Patter Noise
GUI	Graphical User Interface
OB	Observation Block
OCA	Organisation Classification Association
PSO	Paranal Science Operations
QC	Quality Control
RON	Read Out Noise
SOF	Set Of Frames
SQL	Structured Query Language
UT	Unit Telescope
VLT	Very Large Telescope