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

VCAM Pipeline User Manual

VLT-MAN-ESO-19500-XXXX

Issue 1.0

Date 2011-01-14

Prepared:	ESO VCAM Pipeline Team	2011-01-14	
	Name	Date	Signature
Approved:	P.Ballester		
	Name	Date	Signature
Released:	M. Peron		
	Name	Date	Signature

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	14/01/2011	All	First version

This page was intentionally left blank

Contents

1 Introduction

1.1 Purpose

The VCAM pipeline is a subsystem of the *VLT Data Flow System* (DFS). Its target user is ESO *Data Flow Operations* (DFO) in the generation of master calibration data, in the reduction of scientific exposures, and in the data quality control. It should also serve as a quick look tool for *Paranal Science Operations* (PSO).

This manual is a complete description of the data reduction recipes reflecting the status of the VCAM pipeline as of 14.01.2011 (version 1.0.+).

1.2 Scope

This document describes the VCAM 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 [1]. For general information about the current instrument pipelines status we remind the user of [2]. Quality control information are at [3].

Additional information on CFITSIO, the Common Pipeline Library (CPL) and ESOREX can be found respectively at [4], [5], [6]. The Gasgano tool is described in [14]. A description of the instrument is in [7]. The VISTA Data Reduction Library Design document is in [8] and the calibration plan in [9] while results of Science Verifications (SV) are at [?].

1.3 Reference documents

[1]	Vircam Pipeline Users' Manual	VLT-MAN-ESO-?????
	http://www.eso.org/projects/dfs/dfs-shared/web/vlt/vlt-instrument-pipelines.html	
[2]	Current pipeline status	
		http://www.eso.org/observing/dfo/quality/pipeline-status.html
[3]	ESO-Data Flow Operation home page	http://www.eso.org/observing/dfo/quality/
[4]	CFITSIO home page	http://heasarc.nasa.gov/fitsio/fitsio.html
[5]	CPL home page	http://www.eso.org/cpl
[6]	ESOREX home page	http://www.eso.org/cpl/esorex.html
[7]	VISTA home page	http://www.vista.ac.uk
[8]	VISTA Data Reduction Library Design	VIS-SPE-IOA-20000-0010
		http://www.eso.org/instruments/ocam/usermanual.html
[9]	VISTA Infra Red Camera Calibration Plan	VIS-SPE-IOA-20000-0002
[11]	DFS Pipeline & Quality Control – User Manual	VLT-MAN-ESO-19500-1619
[12]	ESO DICB – Data Interface Control Document	GEN-SPE-ESO-00000-0794
[13]	Common Pipeline Library User Manual	VLT-MAN-ESO-19500-2720
[14]	Gasgano User's Manual	VLT-PRO-ESO-19000-1932

2 Overview

In collaboration with instrument consortia, the Pipeline Systems Department (PSD) of the Software Development 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 Vircam jitter stacks; bias-corrected, flat-fielded images). The accuracy of the science products is limited by the quality of the available master calibration products and by the algorithmic implementation of the pipelines themselves. In particular, adopted automatic reduction strategies may not be suitable or optimal for all scientific goals.

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

ESO offers two front-end applications for launching pipeline recipes, *Gasgano* [14] and *EsoRex*, both included in the pipeline distribution (see Appendix **??**, page **??**). These applications can also be downloaded separately from `http://www.eso.org/gasgano` and `http://www.eso.org/cpl/esorex.html`. An illustrated introduction to Gasgano is provided in the "Quick Start" Section of this manual (see page **??**).

The Vircam instrument and the different types of Vircam raw frames and auxilliary data are described in Sections **??**, **??**, and **??**.

A brief introduction to the usage of the available reduction recipes using Gasgano or EsoRex is presented in Section **??**. In section **??** 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 involved in the calibration cascade are provided in section **??**.

More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe are given in section **??**.

More detailed descriptions of the data reduction algorithms used by the individual pipeline recipes can be found in Section **??**.

In Appendix **??** the installation of the Vircam pipeline recipes is described and in Appendix **??** a list of used abbreviations and acronyms is given.

3 Vircam Instrument Description

Vircam has been developed by a consortium of Rutherford Appleton Laboratory, UK Astronomy Technology Centre (UKATC) and University of Durham. The camera will be installed in the VISTA (Visible and Infra Red Survey Telescope for Astronomy) in Paranal. The instrument has been made available to the community and started operations in Paranal on October 1^{st}, 2009.

In this chapter a brief description of the Vircam instrument is given. A more complete documentation can be found in the Vircam User Manual, downloadable from `http://www.eso.org/instruments/ocam/`.

3.1 Instrument overview

The primary goal of Vircam at VISTA is to survey both spatially and over time, through monitoring with good image quality, which is required to enhance sensitivity and resolution for an excellent site. 75% of the VISTA time available to ESO will be available for large scale public surveys and the remaining 25% for smaller proprietary surveys.

Vircam has sixteen 2048x2048 pixel IR detectors (Raytheon VIRGO HgCdTe 0.84-2.5 micron) which are not buttable and are arranged as in the following figure ???? Each camera exposure produces a pawprint consisting of 16 non-contiguous images of the sky. Vircam has a filter wheel which has 8 filter holders, each one containing 16 filters, one for each IR detector. The instrument will be delivered with 4 filter sets (Y, J, H, K) and a further three sets of cold blanks.

The only observing mode is imaging. A target in the sky is observed and tracked and in parallel, the required filter set is placed in the beam. A set of exposures may consist on a number of integrations which usually are jittered by small offsets to remove bad pixels and to determine sky background. This set of exposures is combined in the pipeline to create a single pawprint.

- Pawprint contains 16 non-contiguous images of the sky produced by Vircam with its 16 non-contigous chips. The name is from the similarity to the prints made by the padded paw of an animal.
- Jitter pattern is a pattern of exposures at positions each shifted by a small movement (<30 arcsec) from the reference point. The non-integral part of the shifts is any fractional number of pixels. Each position of a jitter pattern can contain a microstep pattern.
- Mesostep is a sequence of exposures designed to completely sample across the face of the detectors in medium-sized steps, in order to monitor residual systematics in the photometry.
- Microstep pattern is a pattern of exposures at positions each shifted by a very small movement (<3 arcsec) from the reference position. Unlike a jitter, the non-integral part of the shifts are exact fractions of a pixel, which allows the pixels in the series to be interlaced in an effort to increase resolution. A microstep pattern can be contained within each position of a jitter pattern.

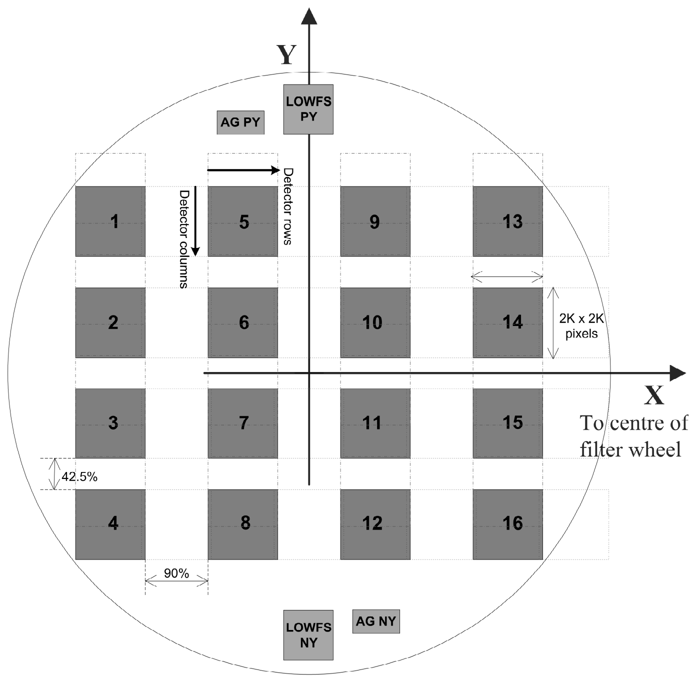

Figure 3.1.0: The Focal plane layout for Vircam.

4 Quick start

This section describes the most immediate usage of the VCAM pipeline recipes. For a complete list of the available recipes, please see Section **??**, page **??**.

4.1 An introduction to Gasgano and EsoRex

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

The process of data classification and association is known as data organisation.

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 tool *Gasgano* .

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.

4.1.1 Using Gasgano

To get familiar with the Vircam 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 system prompt in the following way:

```
gasgano &
```

The *Gasgano* main window will appear. On Figure **??**, a view on a set of Vircam data is shown as an example. *Gasgano* can be pointed to the directories where the data to be handled are located using the navigation panels accessible via the *Add/Remove Files* entry of the *File* menu (shown on the upper left of the figure).

The data are hierarchically organised as preferred by the user. After each file name are shown the classification, the instrument setup id (which indicates the band), the instrument pre-optic (which indicates the camera setting), the template exposure number and the number of exposures in the template, and the value of the DPR.TYPE.

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 for being processed by the appropriate recipe: on Figure **??**, five dark frames are selected and sent to the *vircam_dark_combine* recipe. This will open a *Gasgano* recipe execution window (see Figure **??**), 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 configuration may be opportunely modified on the *Parameters* panel (on top). The window contents might be saved for later use by selecting the *Save Current Settings* entry from the *File* menu, as shown in figure.

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

Please refer to the *Gasgano User's Manual* [7] for a more complete description of the *Gasgano* interface.

TBD

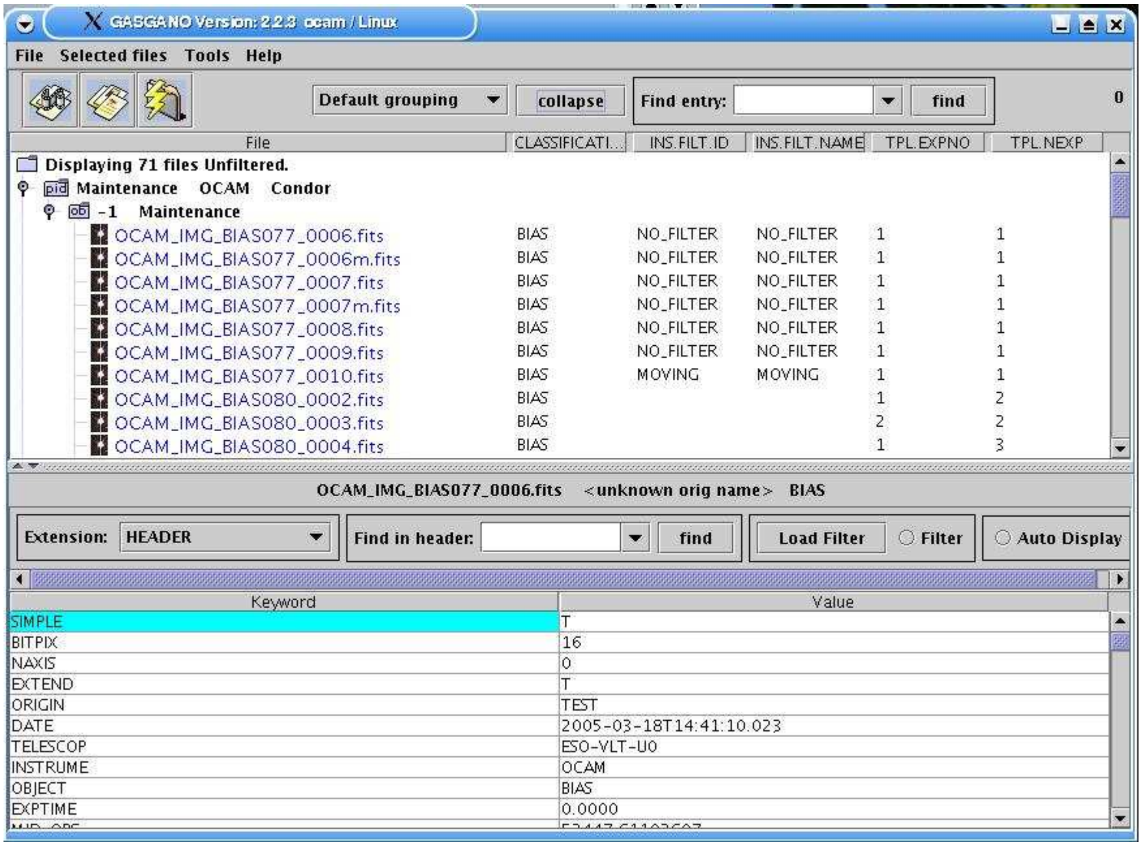

Figure 4.1.1: *The Gasgano main window.*

TBD

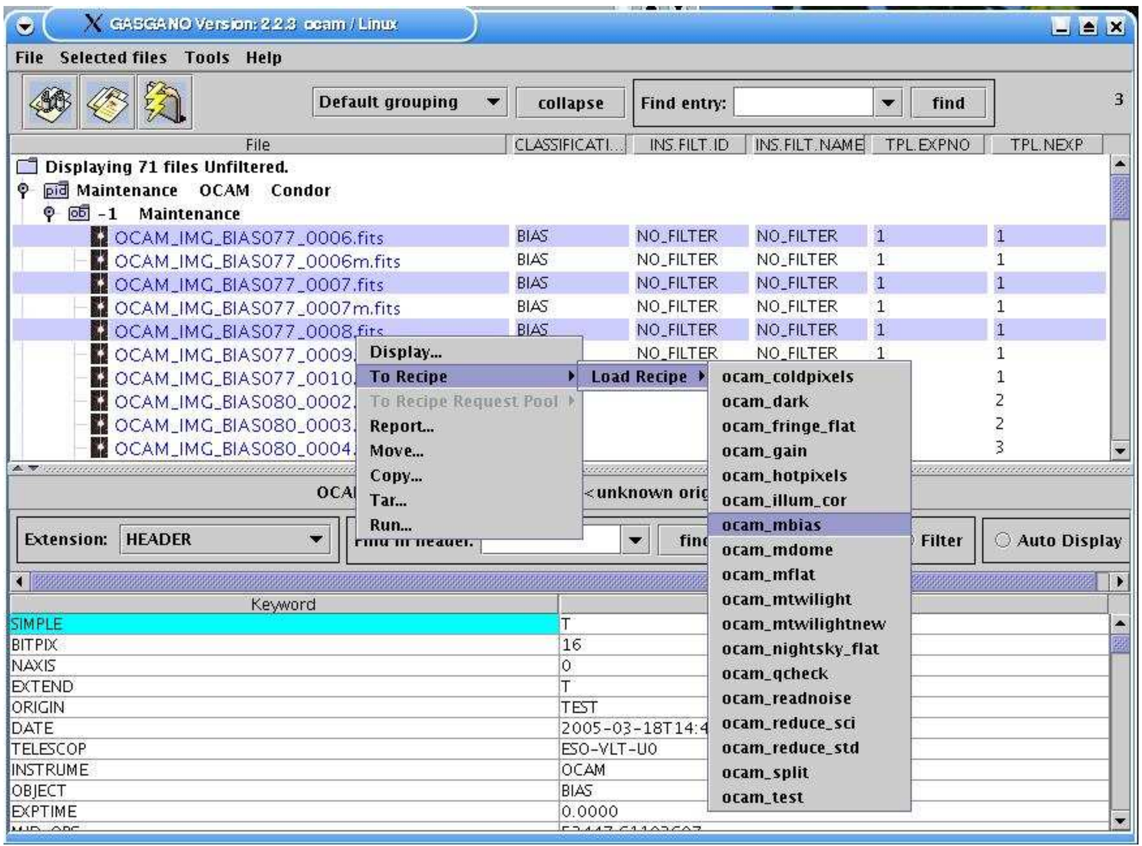

Figure 4.1.2: *Selecting files to be processed by an Vircam pipeline recipe.*

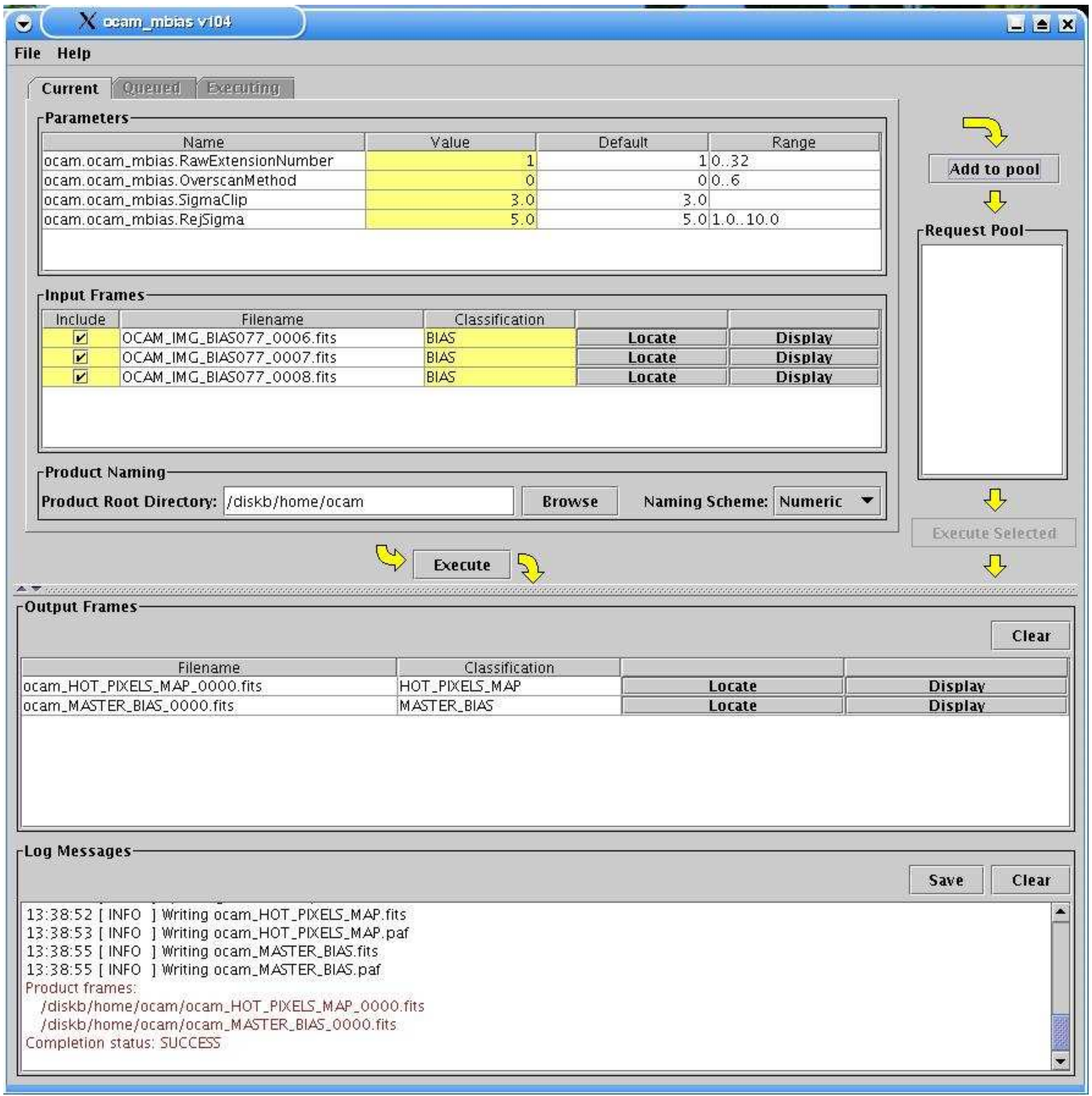

Figure 4.1.3: *The Gasgano recipe execution window.*

4.1.2 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 **??**, page **??**). 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 [1] in an ASCII file, the *set-of-frames* (SOF), that is required when launching a recipe. [2]

Here is an example of SOF, valid for the *vircam_dark_combine* recipe:

```
/file_path/lchantab.fits CHANNEL_TABLE
/file_path/VIRCAM_IMG_DARK159_0001.fits DARK
/file_path/VIRCAM_IMG_DARK159_0002.fits DARK
/file_path/VIRCAM_IMG_DARK159_0003.fits DARK
/file_path/VIRCAM_IMG_DARK159_0004.fits DARK
/file_path/VIRCAM_IMG_DARK159_0005.fits DARK
/file_path/bpm.fits MASTER_BPM
```

It contains for each input frame the full path file name and its DO category. The pipeline recipe will access the listed files when required by the reduction algorithm.

Note that the Vircam pipeline recipes do not verify in any way the correctness of the classification tags specified by the user in the SOF. The reason of this lack of control is that the Vircam recipes are just the DRS 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 **??**, page **??**).

A recipe handling an incorrect SOF may stop or display unclear error messages at best. In the worst cases, the recipe would apparently run without any problem, producing results that may look reasonable, but are actually flawed.

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

esorex [esorex_options] recipe_name [recipe_options] set_of_frames

To get more information on how to customise ESOREX (see also [7]) run the command:

esorex --help

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

esorex --create-config

[1] 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*

[2] The set-of-frames corresponds to the *Input Frames* panel of the *Gasgano* recipe execution window (see Figure **??**, page **??**).

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

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 *vircam_dark_combine* has its *EsoRex* generated configuration file named `vircam_dark_combine.rc`, and is generated with the command:

esorex - -create-config vircam_dark_combine

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

```
# --comb
# Combination method: (1,2)
vircam.vircam_dark_combine.combtype=1
```

In this example, the parameter `vircam.vircam_dark_combine.combtype` is set to the value `1`. 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 (`-comb`).

The recipes provided by the Vircam pipeline are designed to implement a cascade of macro data reduction steps, each controlled by its own parameters. For this reason and to prevent parameter name clashes we specify as parameter prefix not only the instrument name but also the name of the step they refer to. Shorter parameter aliases are made available for use on the command line.

The command

esorex - -create-config recipe_name

generates a default configuration file **recipe_name.rc** in the directory **$HOME/.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

Recipe parameters are provided in section **??** and their role is described in Section **??**.

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.

[3] The *EsoRex* recipe configuration file corresponds to the *Parameters* panel of the *Gasgano* recipe execution window (see Figure **??**, page **??**).

[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 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 *vircam_dark_combine* for processing the files specified in the set-of-frames

`dark1.sof`:

esorex vircam_dark_combine dark1.sof

The recipe parameters can be modifyed 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

vircam_dark_combine recipe *comb* parameter to `2`, the following should be typed:

esorex vircam_dark_combine - -comb=2 dark1.sof

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

4.2 Example of data reduction using EsoRex

A simple, typical data reduction procedure is described here.[5]

We suggest the user to organize the data per type, observed band and camera setting. Dark frames may be grouped per detector DIT, frames to compute distortion and frames to compute detector non linearities may be organized per observed band. The detector DIT is given by the value of the FITS keyword DET DIT[6]. The observed band is indicated by the value of the FITS keyword INS SETUP ID. The camera setting is indicated by the value of INS OPTI1 NAME. In the examples below we suppose the user has data acquired in band V and with the 100 mas pre-optic setting, and DIT=600. In the following examples `/path_raw/` indicates the full path to the source tree directory containing raw data.

Dark Frames: those frames are characterized by DPR.TYPE='DARK',

```
/path_raw/VIRCAM_IMG_DARK159_0001.fits DARK
/path_raw/VIRCAM_IMG_DARK159_0002.fits DARK
/path_raw/VIRCAM_IMG_DARK159_0003.fits DARK
```

Dome flat field frames: those frames are caracterized by DPR.TYPE='FLAT,LAMP'

```
/path_raw/VIRCAM_IMG_FLAT159_0001.fits DOME_FLAT
/path_raw/VIRCAM_IMG_FLAT159_0002.fits DOME_FLAT
/path_raw/VIRCAM_IMG_FLAT159_0003.fits DOME_FLAT
```

[5] The procedure using *Gasgano* is conceptually identical.
[6] We omit here the prefix HIERARCH ESO

5 Instrument Data Description

Vircam data can be separated into *raw* and *product* frames. Raw frames are the unprocessed output of the Vircam instrument observations, while product frames are either the result of the Vircam pipeline processing (as reduced frames, master calibration frames, etc.), or are outsourced (as standard stars catalogs, astrometric catalogs, 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* [7], 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 master bias frame) 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 Vircam 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.

A product frame may be input to more than one Vircam pipeline recipe, but it may be created by just one pipeline recipe (with the same exceptions mentioned above). In the automatic pipeline environment a product data frame alone would not trigger the execution of any recipe.

In the following all raw and product Vircam data frames 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 **??**, page **??**).

The Vircam instrument has only one mode, which is IMAGING. Raw frames, therefore are only distinguished by their observing mode, which can be *DIRECT* and *JITTER*. Their intended use is implicitly defined by the assigned recipe.

- **Linearity:**
  DO category: `FLAT_LAMP_LINEARITY`
  Processed by: `vircam_linearity_analyse`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = FLAT,LAMP,LINEARITY`		
`DPR TECH = IMAGE`		
`INSTRUME = VIRCAM`		

  DO category: `DARK_LINEARITY`
  Processed by: `vircam_linearity_analyse`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = DARK,LINEARITY`		
`DPR TECH = IMAGE`		
`INSTRUME = VIRCAM`		

  DO category: `DARK_CHECK`
  Processed by: `vircam_linearity_analyse`

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB		
DPR TYPE = DARK,CHECK		
DPR TECH = IMAGE		
INSTRUME = VIRCAM		

DO category: FLAT_LAMP_CHECK
Processed by: vircam_linearity_analyse

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB		
DPR TYPE = FLAT,LAMP,CHECK		
DPR TECH = IMAGE		
INSTRUME = VIRCAM		

- **Dark:**

  DO category: DARK
  Processed by: vircam_dark_combine

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB	INSTRUME = VIRCAM	Instrument used
DPR TYPE = DARK		
DPR TECH = IMAGE		
INSTRUME = VIRCAM		

- **Reset:**

  DO category: BIAS
  Processed by: vircam_reset_combine

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB		
DPR TYPE = BIAS	INSTRUME = VIRCAM	Instrument used
DPR TECH = IMAGE		
INSTRUME = VIRCAM		

- **Dark Current:**

  DO category: DARK_CURRENT
  Processed by: vircam_darkcurrent

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB		
DPR TYPE = DARKCURRENT		
DPR TECH = IMAGE		
INSTRUME = VIRCAM		

- **Detector Noise:**
  DO category: DARK_GAIN
  Processed by: vircam_detector_noise

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB		
DPR TYPE = DARK,GAIN		
DPR TECH = IMAGE		
INSTRUME = VIRCAM		

  DO category: FLAT_LAMP_GAIN
  Processed by: vircam_detector_noise

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB		
DPR TYPE = FLAT,LAMP,GAIN		
DPR TECH = IMAGE		
INSTRUME = VIRCAM		

- **Dome Flat Field:**
  DO category: FLAT_LAMP
  Processed by: vircam_dome_flat_combine

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB	INSTRUME = VIRCAM	Instrument used
DPR TYPE = FLAT,LAMP	INS FILT[1-4] ID	Filter unique ID
DPR TECH = IMAGE	DET DIT	actual subintegration time
INSTRUME = VIRCAM	DET NDIT	actual subintegration times

- **Twilight flat field:**
  DO category: FLAT_TWILIGHT
  Processed by: vircam_twilight_flat_combine

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB	INSTRUME = VIRCAM	Instrument used
DPR TYPE = FLAT,TWILIGHT	INS FILT[1-4] ID	Filter unique ID
DPR TECH = IMAGE	DET DIT	actual subintegration time
INSTRUME = VIRCAM	DET NDIT	actual subintegration times

- **Standard Stars:**
  DO category: STD_FLUX
  Processed by: vircam_standard_process

Classification keywords:	Association keywords:	Note:
DPR CATG = CALIB	INSTRUME = VIRCAM	Instrument used
DPR TYPE = STD,FLUX	INS FILT[1-4] ID	Filter unique ID
DPR TECH = IMAGE,JITTER	DET DIT	actual subintegration time
INSTRUME = VIRCAM	DET NDIT	actual subintegration times

- **Illumination:**

  DO category: `STD_ILLUMINATION`
  Processed by: `vircam_mesostep_analyse`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`	`INSTRUME = VIRCAM`	Instrument used
`DPR TYPE = STD,ILLUMINATION`	`INS FILT[1-4] ID`	Filter [1-4] on each beam
`DPR TECH = IMAGE,JITTER`	`DET DIT`	actual subintegration time

\> `DET NDIT` `INSTRUME = VIRCAM` actual subintegration times

- **Science:** DO category: `OBJECT`
  Processed by: `vircam_jitter_microstep_process`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`	`INSTRUME = VIRCAM`	Instrument used
`DPR TYPE = OBJECT`	`INS FILT[1-4] ID`	Filter [1-4] on each beam
`DPR TECH = IMAGE,JITTER`	`DET DIT`	actual subintegration time
`INSTRUME = VIRCAM`	`DET NDIT`	actual subintegration times

DO category: `OBJECT_EXTENDED`
Processed by: `vircam_jitter_microstep_process`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`	`INSTRUME = VIRCAM`	Instrument used
`DPR TYPE = OBJECT,EXTENDED`	`INS FILT[1-4] ID`	Filter [1-4] on each beam
`DPR TECH = IMAGE,JITTER`	`DET DIT`	actual subintegration time
`INSTRUME = VIRCAM`	`DET NDIT`	actual subintegration times

6 Static Calibration Data

In the following, all static calibration frames used in the VCAM pipeline are listed, together with the keywords used for their correct association. The indicated *PRO.CATG* is a label assigned to any product data type, which is then used to identify the calibration frames listed in the *Set of Frames* (see Section **??**, page **??**).

6.1 Photometric calibration table

DO category: `PHOT_TABLE`

This is a table used to define the transformation from instrumental to standard magnitudes.

Name	Type	Description
$filter$	char	The name of the filter
$extinction$	float	The extinction coefficient for airmass of unity for the given filter
$offset$	float	A pedestal value to be added to the instrumental magnitude
$columns$	char	The standard magnitude columns from the matched standards catalogue to be used
$coleq$	char	The colour equation coefficients

6.2 Channel table

DO category: `CHANNEL_TABLE_INIT`

Each VIRCAM detector is split into 16 different data channels, each with its own electronics. This means that some reduction tasks will rely on knowing the location and readout timing information for each data channel. The location and linearity information will be provided by the âĂĲchannel tableâĂ§. The information will be stored in a multi-extension FITS file with each extension being a FITS binary table (one for each detector). Each of the tables will contain the columns listed in below (although perhaps not in this order). The extension name should match the extension names for the input images. It is worth remembering here that there is no zeroth order coefficient, so the number of coefficient columns is the same as the polynomial order. Two types of channel tables will be employed as shown in Table 5-1 and Table 5-3. A recipe which requests a table of type CHANNEL_TABLE_INIT will only make use of the positional information for the channels. Any information about the linearity and the readout parameters will be ignored. If a MASTER_CHANNEL_TABLE is requested, then the recipe will need the linearity information. All channel table files must have in each extension an estimate of the saturation level in ADUs in the keyword ESO DET SATURATION.

Name	Type	Units	Description
$channum$	int		Number of the data channel
$ixmin$	int	pixels	Lower left X coordinate
$ixmax$	int	pixels	Upper right left X coordinate
$iymin$	int	pixels	Lower left Y coordinate
$iymax$	int	pixels	Upper right left Y coordinate
$dcrpix1$	int	pixels	First pixel on readout, X coordinate
$dcrpix2$	int	pixels	First pixel on readout, Y coordinate
$dcd1_1$	int		Partial derivative, fast readout axis, X axis
$dcd1_2$	int		Partial derivative, fast readout axis, Y axis
$dcd2_1$	int		Partial derivative, slow readout axis, X axis
$dcd2_2$	int		Partial derivative, slow readout axis, Y axis
$lin_1 0000$	double		Non-linearity at a level of 10000 ADU
$lin_1 0000_e rr$	double		Error of the above
$norder$	int		Order of the polynomial fit
$coeff_1$	double		First coefficient of the fit, always 1
$coeff_n$	double		n-th coefficient of the fit, n=norder

7 Data Reduction

7.1 VCAM pipeline recipes

The current VCAM pipeline is based on a set of 12 stand-alone recipes involved in the data reduction cascade:

vircam_reset_combine provides stability information on the pedestal and pixel to pixel variation.

vircam_dark_combine calculates the variation in the reset anomaly structure and scale.

vircam_dark_current calculates the accumulated counts from the thermal noise.

vircam_dome_flat_combine monitors instrument performance, image structure and confidence maps.

vircam_detector_noise measures the readout noise and gain of each chip.

vircam_linearity_analyse determines the linearity curve of each detector.

vircam_twilight_flat_combine removes pixel-to-pixel gain variations and the instrumental vignetting profile. It also provides a global gain correction.

vircam_mesotep_analyse for standard stars field data reduction and the determination of

vircam_persistence_analyse removes image persistence from images.

vircam_crosstalk_analyse remove the cross-talk if eventual alterations happen to the electrical environment.

vircam_jitter_microstep_analyse processes a sequence of science data.

vircam_standard_process processes a sequence of photometric standard data.

8 Pipeline Recipes Interface

In this Section a detailed description of the FORS pipeline recipes interfaces is given, with a complete specification of the recipes usage, their input, output, and configuration parameters.

8.1 vircam_reset_combine

This recipe combines a sequence of reset frames to form a mean frame. It also compares to a library reset frame to provide information on the stability of the pedestal and reset structure.

8.1.1 Input files

BIAS: *required* list of reset frames

REFERENCE_BIAS: *optional* library mean reset frame

MASTER_CHANNEL_TABLE: *optional* channel table

CHANNEL_TABLE_INIT: *optional* channel table

MASTER_BPM: *optional* library bad pixel map

MASTER_CONF: *optional* library confidence map

8.1.2 Output files

MASTER_BIAS: new master reset frame

DIFFIMG_BIAS: difference image

DIFFIMG_STATS_BIAS: reset difference image statistics table

8.1.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_reset_combine* recipe will process the input frames.

--*combtype:* Determines the type of combination that is done to form the output map. Can take the following values:

1. The output pixels are medians of the input pixels.

1. The output pixels are means of the input pixels.

--*scaletype:* Determines how the input data are scaled or offset before they are combined. Can take the following values:

0. No scaling or biasing.

1. All input frames are biased additively to bring their backgrounds to a common median level.

2. All input frames are biased multiplicatively to bring their backgrounds to a common median level.

3. All input frames are scaled to a uniform exposure time and then additively corrected to bring their backgrounds to a common median level.

--*xrej:* If set, then an extra rejection cycle will be run.

--*thresh:* The rejection threshold in numbers of background sigmas.

--*ncells:* If a difference image statistics table is being done, then this is the number of cells in which to divide each readout channel. The value must be a power of 2, up to 64.

8.1.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_reset_combine* recipe.

QC RESETMED: Reset median level. *Units*: ADU

QC RESETRMS: Reset standard deviation. *Units*: ADU

QC RESETDIFF_MED: Reset difference image median level. *Units*: ADU

QC RESETDIFF_RMS: Reset difference image standard deviation. *Units*: ADU

8.2 vircam_dark_combine

This recipe combines a series of dark frames taken with a particular integration and exposure time combination. It compares with a similarly observed master dark frame and calculates the variation in the reset anomaly structure and scale.

8.2.1 Input files

DARK: *required* list of dark frames

REFERENCE_DARK: *optional* library mean dark frame

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

MASTER_CHANNEL_TABLE: *optional* channel table

CHANNEL_TABLE_INIT: *optional* channel table

8.2.2 Output files

MASTER_DARK: new master dark frame

DIFFIMG_DARK: dark frame difference image

DIFFIMG_STATS_DARK: dark difference image statistics table

8.2.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_dark_combine* recipe will process the input frames.

--*combtype:* Determines the type of combination that is done to form the output map. Can take the following values:

- **1.** The output pixels are medians of the input pixels.
- **1.** The output pixels are means of the input pixels.

--*scaletype:* Determines how the input data are scaled or offset before they are combined. Can take the following values:

- **0.** No scaling or biasing.
- **1.** All input frames are biased additively to bring their backgrounds to a common median level.
- **2.** All input frames are biased multiplicatively to bring their backgrounds to a common median level.

3. All input frames are scaled to a uniform exposure time and then additively corrected to bring their backgrounds to a common median level.

--*xrej:* If set, then an extra rejection cycle will be run.

--*thresh:* The rejection threshold in numbers of background sigmas.

--*ncells:* If a difference image statistics table is being done, then this is the number of cells in which to divide each readout channel. The value must be a power of 2, up to 64.

8.2.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_dark_combine* recipe.

QC DARKMED: Dark median level. *Units*: ADU

QC DARKRMS: Dark standard deviation. *Units*: ADU

QC DARKDIFF_MED: Reset difference image median level. *Units*: ADU

QC DARKDIFF_RMS: Reset difference image standard deviation. *Units*: ADU

QC PARTICLE_RATE: Particle rate.

QC STRIPERMS: Stripe standard deviation.

QC NHOTPIX: Number of hot pixels.

QC HOTFRAC:

QC RON12:

8.3 vircam_dark_current

This recipes calculates the dark current of a detector using a series of dark exposures with varying exposure times.

8.3.1 Input files

DARK_DARKCURRENT: *required* series of dark exposures at a variety of different exposure times.

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

8.3.2 Output files

MASTER_DARK_CURRENT: Dark current map

8.3.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_dark_current* recipe will process the input frames.

--*thresh:* The threshold in units of background sigma above or below the local mean value. This defines whether a data point in the fit is bad or not.

8.3.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_dark_current* recipe.

QC DARKCURRENT: Value of the dark current. *Units*: ADU

8.4 vircam_dome_flat_combine

This recipe combines a series of dome flat images to create a mean dome flat. It compares with a similarly observed master dome flat frame.

8.4.1 Input files

FLAT_LAMP: *required* list of dome flat exposures all taken with the same exposure parameters.

MASTER_DARK: *required* master dark frame of the same exposure parameters as above.

REFERENCE_DOME_FLAT: *optional* master mean dome flat.

MASTER_CHANNEL_TABLE: *optional* channel table

CHANNEL_TABLE_INIT: *optional* channel table

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

8.4.2 Output files

MASTER_DOME_FLAT: New master dome flat

RATIOIMG_DOME_FLAT: Ratio image

RATIOIMG_STATS_DOME_FLAT: Ratio image statistics table

8.4.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_dome_flat_combine* recipe will process the input frames.

--*lthr:* Any input flat with a mean value of less than this will be excluded as being underexposed.

--*hthr:* Any input flat with a mean value of more than this will be excluded as being overexposed.

--*combtype:* Determines the type of combination that is done to form the output map. Can take the following values:

- **1.** The output pixels are medians of the input pixels.
- **1.** The output pixels are means of the input pixels.

--*scaletype:* Determines how the input data are scaled or offset before they are combined. Can take the following values:

- **0.** No scaling or biasing.
- **1.** All input frames are biased additively to bring their backgrounds to a common median level.
- **2.** All input frames are biased multiplicatively to bring their backgrounds to a common median level.
- **3.** All input frames are scaled to a uniform exposure time and then additively corrected to bring their backgrounds to a common median level.

--*xrej:* If set, then an extra rejection cycle will be run.

--*thresh:* The rejection threshold in numbers of background sigmas.

--*ncells:* If a difference image statistics table is being done, then this is the number of cells in which to divide each readout channel. The value must be a power of 2, up to 64.

8.4.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_dome_flat_combine* recipe.

QC FLATRMS:

QC FLATRATIO_MED:

QC FLATRATIO_RMS:

QC FLATMIN:

QC FLATMAX:

QC FLATAVG:

QC FLAGRNG:

8.5 vircam_detector_noise

This recipe measures the detector readout noise and gain.

8.5.1 Input files

FLAT_LAMP_GAIN: *required* two dome flat frames taken with the same exposure parameters.

DARK_GAIN: *required* two dome flat frames taken with the same exposure parameters as the dome flats.

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

8.5.2 Output files

MASTER_READGAIN_TABLE: Read noise and gain estimates for each extension written to a detector noise table.

8.5.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_detector_noise* recipe will process the input frames.

--*thresh:* The threshold in units of background sigma above or below the local mean value. This is used during the statistical analyses of the input images and difference images.

8.5.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_detector_noise* recipe.

QC READNOISE:

QC CONAD:

QC COUNTS:

QC LAMPFLUX:

8.6 vircam_linearity_analyse

This recipe creates the detector channel linearity curves and bad pixel maps.

8.6.1 Input files

FLAT_LAMP_LINEARITY: *required* a series of dome flat exposures taken under constant illumination with varying integration times.

CHANNEL_TABLE_INIT: *required* channel table

DARK_LINEARITY: *required* a list of raw dark frames containing a series of dark exposures with the same exposure parameters for each of the input dome flat exposures.

FLAT_LAMP_CHECK: *optional* a series of dome flat exposures taken at a constant exposure time and done in between the linearity sequence exposures. These are used to monitor drift in the dome lamps.

DARK_CHECK: *optional* a series of raw dark frames taken with the same exposure parameters as the monitor dome flats.

8.6.2 Output files

MASTER_CHANNEL_TABLE: Output channel table with new linearity information.

MASTER_BPM: Output bad pixel mask.

LINEARITY_SEQ_DIAG: Output diagnostic curves for the linearity sequence.

LINEARITY_CHECK_DIAG: Output diagnostic curves for monitor sequence.

8.6.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_linearity_analyse* recipe will process the input frames.

--*nord:* The order of the polynomial to be fit to the linearity curve of each channel.

--*lthr:* The lower threshold in the ratio maps to define a pixel as bad. Units are in background sigma of the ratio map.

--*hthr:* The upper threshold in the ratio maps to define a pixel as bad. Units are in background sigma of the ratio map.

--*maxbpmfr:* The bad pixel mask computation may not need to use all of the input flat frames. This parameter defines the size of the subset to use.

8.6.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_linearity_analyse* recipe.

QC LINEARITY:

QC LINERROR:

QC SCREEN_TOTAL:

QC SCREEN_STEP:

QC BAD_PIXEL_STAT:

QC BAD_PIXEL_NUM:

8.7 vircam_twilight_flat_combine

This recipe combines a series of twilight flat images to create a mean twilight flat and initial confidence map. Compare with a similarly observed master twilight flat frame.

8.7.1 Input files

FLAT_TWILIGHT: *required* list of twilight flat exposures all taken with the same exposure parameters.

MASTER_DARK: *required* master dark frame of the same exposure parameters as above

REFERENCE_TWILIGHT_FLAT: master mean twilight flat

MASTER_CHANNEL_TABLE: *optional* channel table

CHANNEL_TABLE_INIT: *optional* channel table

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

8.7.2 Output files

MASTER_TWILIGHT_FLAT: master twilight flat

RATIOIMG_TWILIGHT_FLAT: ratio image

RATIOIMG_STATS_TWILIGHT_FLAT: ratio image statistics table

MASTER_CONF: new master confidence map

8.7.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_twilight_flat_combine* recipe will process the input frames.

--*lthr:* Any input flat with a mean value of less than this will be excluded as being underexposed.

--*hthr:* Any input flat with a mean value of more than this will be excluded as being overexposed.

--*combtype:* Determines the type of combination that is done to form the output map. Can take the following values:

1. The output pixels are medians of the input pixels.
1. The output pixels are means of the input pixels.

--*scaletype:* Determines how the input data are scaled or offset before they are combined. Can take the following values:

0. No scaling or biasing.
1. All input frames are biased additively to bring their backgrounds to a common median level.
2. All input frames are biased multiplicatively to bring their backgrounds to a common median level.
3. All input frames are scaled to a uniform exposure time and then additively corrected to bring their backgrounds to a common median level.

--*xrej:* If set, then an extra rejection cycle will be run.

--*thresh:* The rejection threshold in numbers of background sigmas.

--*ncells:* If a difference image statistics table is being done, then this is the number of cells in which to divide each readout channel. The value must be a power of 2, up to 64.

8.7.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_twilight_flat_combine* recipe.

QC FLATRMS:

QC FLATRATIO_MED:

QC FLATRATIO_RMS:

QC FLATMIN:

QC FLATMAX:

QC FLATAVG:

QC FLATRNG:

8.8 vircam_mesostep_analyse

This recipe creates a map of illumination corrections using a mesostep sequence of a standard stars.

8.8.1 Input files

STD_ILLUMINATION: *required* a series of exposures of a sparse secondary standard field that has been offset in a regular raster

MASTER_DARK: *required* library master dark frame for the given exposure and integration time

MASTER_TWILIGHT_FLAT: library master flat field for the given passband

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

MASTER_CHANNEL_TABLE: *optional* channel table

PHOTCAL_TAB: *required* photometric calibration table

MASTER_2MASS_CATALOGUE: *required* master 2MASS index

8.8.2 Output files

ILLCOR_TAB_MES: illumination correction table

8.8.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_mesostep_analyse* recipe will process the input frames.

--*ipix:* The minimum size of an object in pixels in order for that object not to be considered spurious.

--*tthr:* The detection threshold measured in units of the mean background noise.

--*icrowd:* If set, then the function will attempt to de-blend merged objects.

--*rcore:* The core radius in pixels for the default profile fit.

--*nb:* The size in pixels of the grid squares used for background estimation.

--*destripe:* If this is set, then the input images will be de-striped. Not recommended for images that are likely to contain very large extended objects.

--*skycor:* If this is set, then the input images are stacked with rejection to form a mean background map. This is normalised to zero median and subtracted off the input images. This is not recommended for images that are likely to contain extended sources.

--*nord:* The order of the polynomial surface to be fit.

8.8.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_mesostep_analyse* recipe.

QC ILLUMCOR_RMS:

8.9 vircam_jitter_microstep_process

This science recipe processes a sequence of target data that may have been both jittered and microstepped.

8.9.1 Input files

OBJECT or OBJECT_EXTENDED: *required* A jittered and/or microstepped sequence of exposures of a target region.

SKY_OFFSET: *optional* optional offset sky exposures

MASTER_DARK: *required* library mean dark frame for the given exposure and integration time.

MASTER_TWILIGHT_FLAT: library master flat field for the given passband

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

MASTER_SKY: *optional* optional master sky background frame

MASTER_CHANNEL_TABLE: *optional* channel table

MASTER_READGAIN_TABLE: readnoise/gain table

PHOTCAL_TAB: photometric calibration table

MASTER_2MASS_CATALOGUE: *required* master 2MASS index

8.9.2 Output files

SIMPLE_IMAGE_SCI: Single exposure images that are corrected for linearity, dark current, flat field, stripes and sky. A full WCS will appear in the header.

SIMPLE_IMAGE_SKY: As above for any offset sky exposures that are included in the input data. These will not have been processed past the point of background subtraction and hence will not have a calibrated WCS in the header.

INTERLEAVED_IMAGE_SCI: Interleaved super-frame images from the above if microstepping has been done as part of the observing sequence.

JITTERED_IMAGE_SCI: Stacked jitter images from the super-frames. Full WCS and photometric zero point will appear in the FITS header.

MEAN_OFFSET_SKY: Mean sky frame if a set of offset sky exposures have been included.

CONFIDENCE_MAP_INTER_SCI and CONFIDENCE_MAP_SCI: Associated confidence maps for each of the above output images.

OBJECT_CATALOGUE_SCI: Object catalogue in the form of a FITS table if the savecat parameter has been set.

8.9.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_jitter_microstep_process* recipe will process the input frames.

--*ipix:* The minimum size of an object in pixels in order for that object not to be considered spurious.

--*thr:* The detection threshold measured in units of the mean background noise.

--*icrowd:* If set, then the function will attempt to de-blend merged objects.

--*rcore:* The core radius in pixels for the default profile fit.

--*nb:* The size in pixels of the grid squares used for background estimation.

--*destripe:* If this is set, then the input images will be de-striped. Not recommended for images that are likely to contain very large extended objects.

--*skycor:* If this is set, then the input images are stacked with rejection to form a mean background map. This is normalised to zero median and subtracted off the input images. This is not recommended for images that are likely to contain extended sources.

--*savecat:* If set, then the catalogue generated during the astrometric and photometric calibration will be saved.

--*savesimple:* If set then the simple images will be saved on output. Otherwise they will be thrown away after they are stacked.

8.9.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_jitter_microstep_process* recipe.

QC SATURATION:

QC MEAN_SKY:

QC SKY_NOISE:

QC NOISE_OBJ:

QC IMAGE_SIZE:

QC APERTURE_CORR:

QC ELLIPTICITY:

QC MAGZPT:

QC MAGZERR:

QC MAGNZPT:

QC MAGNCUT:

QC LIMITING_MAG:

QC WCS_DCRVAL1:

QC WCS_DCRVAL2:

QC WCS_DTHETA:

QC WCS_SCALE:

QC WCS_SHEAR:

QC WCS_RMS:

8.10 vircam_standard_process

This science recipe processes a sequence of photometric standard data that may have been both jittered and microstepped.

8.10.1 Input files

STD_FLUX: *required* a jittered and/or microstepped sequence of exposures of a target region

MASTER_DARK: *required* library mean dark frame for the given exposure and integration time.

MASTER_TWILIGHT_FLAT: library master flat field for the given passband

MASTER_BPM: *optional* library bad pixel mask

MASTER_CONF: *optional* library confidence map

MASTER_CHANNEL_TABLE: *optional* channel table

MASTER_READGAIN_TABLE: readnoise/gain table

PHOTCAL_TAB: photometric calibration table

MASTER_2MASS_CATALOGUE: *required* master 2MASS index

MASTER_REFERENCE_CATALOGUE: *required* master standard star index

8.10.2 Output files

SIMPLE_IMAGE_STD: Single exposure images that are corrected for linearity, dark current, flat field, stripes and sky. A full WCS will appear in the header.

ILLCOR_TAB_SKY: illumination correction table

INTERLEAVED_IMAGE_STD: Interleaved super-frame images from the above if microstepping has been done as part of the observing sequence.

JITTERED_IMAGE_STD: Stacked jitter images from the super-frames. Full WCS and photometric zero point will appear in the FITS header.

CONFIDENCE_MAP_INTER_STD and CONFIDENCE_MAP_STD: Associated confidence maps for each of the above output images.

OBJECT_CATALOGUE_STD: Object catalogue in the form of a FITS table if the savecat parameter has been set.

8.10.3 Configuration parameters

The configuration parameters setting determines the way the *vircam_jitter_microstep_process* recipe will process the input frames.

--*ipix:* The minimum size of an object in pixels in order for that object not to be considered spurious.

--*thr:* The detection threshold measured in units of the mean background noise.

--*icrowd:* If set, then the function will attempt to de-blend merged objects.

--*rcore:* The core radius in pixels for the default profile fit.

--*nb:* The size in pixels of the grid squares used for background estimation.

--*destripe:* If this is set, then the input images will be de-striped. Not recommended for images that are likely to contain very large extended objects.

--*skycor:* If this is set, then the input images are stacked with rejection to form a mean background map. This is normalised to zero median and subtracted off the input images. This is not recommended for images that are likely to contain extended sources.

--*savecat:* If set, then the catalogue generated during the astrometric and photometric calibration will be saved.

--*savesimple:* If set then the simple images will be saved on output. Otherwise they will be thrown away after they are stacked.

8.10.4 Quality control parameters

Currently the following QC parameters, used by PSO and DPD, are evaluated by the *vircam_jitter_microstep_process* recipe.

QC SATURATION:

QC MEAN_SKY:

QC SKY_NOISE:

QC NOISE_OBJ:

QC IMAGE_SIZE:

QC APERTURE_CORR:

QC ELLIPTICITY:

QC MAGZPT:

QC MAGZERR:

QC MAGNZPT:

QC MAGNCUT:

QC LIMITING_MAG:

QC ILLUMCOR_RMS:

QC WCS_DCRVAL1:

QC WCS_DCRVAL2:

QC WCS_DTHETA:

QC WCS_SCALE:

QC WCS_SHEAR:

QC WCS_RMS:

QC ZPT_2MASS:

QC ZPT_STDS:

QC ZPT_STDS_CAT:

9 Algorithms

9.1 Reset Correction

As with most electronic detectors infrared detectors are given a pedestal bias level by the driving electronics. As such the first step in any reduction of such data is to remove that bias. For VIRCAM this will be done in the DAS. This removes the need for explicit bias removal in the pipeline.

9.2 Non Linearity

The Calibration Plan [AD3] lays out the necessity and the methodology for calibrating and correcting for the expected non-linearity in the response of the detector system to incident radiation.

9.2.1 Correcting for non linearity

In default CDS reset-read-read (RRR) mode, downstream of the data acquisition system (DAS) the output that we see is

$$\Delta I' = I'_2 - I'_1 = f(I_2) - f(I_1)$$

where I'_1 and I'_2 denote the non-linear first (i.e. the reset-frame) and second readouts respectively and I_1 and I_2 the desired linear quantities. The non-linear function $f(I)$ maps the distortion of the desired linear ounts to the non-linear system I'. If we define the inverse transform $g(I')$ that maps measured counts I' to linearized counts I as the inverse operator $g() = f^{-1}()$ then

$$I = g(I') \text{ and } I_1 = g(I'_1) I_2 = g(I'_2)$$

If I'_1 and I'_2 were directly available this is a one-to-one mapping and can be done efficiently and accurately using Look Up Tables (LUT). This is the conventional way of implementing the correction prior to other image manipulation operations. However, if I'_1 and I'_2 are not separately available and all we have to work from is the difference $\Delta I'$ then a simple LUT transformation is not possible.

For example, taking the simplest case where the illumination level across the detector has not changed during the course of the RRR and no on-board co-addition is happening then, in principle given only ΔI and knowledge of the timing of the RRR operations we can deduce I_1 and I_2 by using the effective integration time for each to estimate their scaling to the measured difference ΔI such that,

$$I_1 = k\Delta I \text{ and } I_2 = (1 + k)\Delta I$$

Unfortunately, the ratio k will not be constant for the non-linear quantities I'_1 and I'_2 forcing us to adopt a scheme along the following lines.

Given $\Delta I'$ and defining the non-linear operator $f()$ as a polynomial with coefficients a_m (typically up to quartic) we have

$$\Delta I' = \sum a_m (I_2^m - I_1^m) = \sum a_m [(1+k)^m \Delta I^m - k^m \Delta I^m]$$

The quantity we want ΔI is buried in the non-linearity of the RHS and we have to solve an equation like this for every pixel. This is possible, and relatively simple to program using something like a Gauss-Seidel iterative scheme, but is more inefficient than a direct mapping.

If we wanted to use a completely general LUT approach we would require a 2D LUT for all possible values of I_1 and I_2 i.e. 65k x 65k in size, or 4.3 x 2 Gbytes. Most likely we would need a different correction for each âĂIJchannelâĂİ making a total of 256 x 8.6 Gbytes = 2.2 Tbytes of LUT for the VIRCAM! Of course if the range of values of k is limited via exposure time quantisation this decreases the size of the total

number of LUTs required considerably for the constant illumination case, but would be an ugly and possibly impractical solution.

Practical considerations (e.g. data volume), suggest two alternative solutions for nonlinearity correction: either correct the individual frames directly in the DAS by measuring and downloading the appropriate LUTs, or polynomial coefficients, to the DAS; or use a non-linear inversion on the reset-corrected frames as outlined here. This methodology is not generally applicable, e.g. to multi-NDR/gradient fitting readouts, but is directly applicable to co-added (or co-averaged) frames of the same exposure times, assuming constant illumination over the series.

For reset-corrected data, the non-linear inversion is competitive with complex operations on LUTs and much simpler to implement. It also has the added advantage of removing all aspects of the non-linearity correction from the DAS. The main disadvantages are the method is restricted to CDS RRR mode, and if the illumination level is rapidly varying (e.g. twilight) the effective scale factors ki may be hard to compute accurately - although for all realistic practical situations the knock-on effect is likely to be negligible.

9.2.2 Measuring non linearity

If all that is available are reset-corrected data from say a time series of dome flats, it is still feasible to compute the non-linearity coefficients directly. Given a series of measurements i of $\Delta I_i'$ and using the previous notation and polynomial model

$$\Delta I_i' = \sum a_m (I_2^m - I_1^m) = \sum a_m \Delta I_i^m [(1+k_i)^m - k_i^m]$$

where k_i are the exposure ratios under the constant illumination. In general $\Delta I_i = st_i$ where ti is the exposure time of the ith reset-corrected frame in the series and s is a fixed (for the series) unknown scale factor. The k_i are computable from a knowledge of the exposure times and the reset-read overhead, t_i and $\Delta I_i'$ are measured quantities leaving the polynomial coefficients a_m and the scaling s to be determined. Thus the model is defined by

$$\Delta I_i' = \sum a_m (I_2^m - I_1^m) = \sum a_m s^m t_i^m [(1+k_i)^m - k_i^m]$$

and can be readily solved by standard linear least-squares methods using the following sleight-of-hand. Since the scaling s and hence the polynomial solution a_m are coupled, by simply (and logically) requiring in the final solution $a_1 = 1$, computation of s can be completely avoided.

Rewriting the previous equation in the following form makes this more apparent

$$\Delta I'_i = \sum (a_m s^m) t_i^m [(1+k_i)^m - k_i^m] = \sum b_m t_i^m [(1+k_i)^m - k_i^m]$$

where now b_m are the coefficients to be solved for. The final step is to note that

$$a_m = b_m / s^m = b_m / b_1^m$$

since by definition $a_1 = 1$.

One final simplification can be made when it is realised that the readout time and the reset time for the VIRCAM detectors are the same. This means that the value of k_i is constant for all the pixels in a given image and is given by:

$$k_i = r/t_i$$

where r is the readout/reset time. Substituting this into equation 2.7 above and reorganising the terms, we can show that this reduces to a simple power series fit:

$$\Delta I'_i = \sum c_m t_i^m$$

The relationship between the coefficient vectors $\boldsymbol{c}$ and $\boldsymbol{b}$ is defined by an intermediate matrix $\boldsymbol{M}$ so that:

$$\boldsymbol{c} = \boldsymbol{M}\boldsymbol{b}$$

An example of $\boldsymbol{M}$ for a 4th order fit is:

$$\begin{pmatrix} 1 & 2r & 3r^2 & 4r^3 \\ 0 & 1 & 3r & 6r^2 \\ 0 & 0 & 1 & 4r \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This can be solved using a straight forward Gaussian elimination algorithm and the values of b_m and then a_m can be recovered.

9.3 Gain Correction

In the case of a single detector camera the mean flat field image is normalised to a value of 1. This ensures that when the flat field correction is done the average counts in the output image is the same as in the input. For multi-detector instruments, we normalise the mean flat field image for each detector by:

$$V = \frac{\sum \langle I \rangle_i}{n}$$

where $\langle I \rangle_i$ is a robust estimate of the average flux in the combined flat field image for the ith detector. Normalising in this way ensures that when doing flat field correction we also include a factor that removes the differences in mean gain of each detector.

9.4 Measurement of Read Noise and Gain

The read noise and gain can be measured using two dome flat frames of similar illumination and two similarly observed (in terms of exposure and integration times) dark frames. Forming the difference of the two flat frames gives a variance for the difference frame σ_f^2. Doing the same for the two dark frames yields σ_d^2. If the background means of the flat and dark frames are: m_{f1}, m_{f2} and m_{d1}, m_{d2} the local gain in electrons per ADU is:

$$\varepsilon = ((m_{f1} + m_{f2}) - (m_{d1} + m_{d2}))/(\sigma_f^2 - \sigma_d^2)$$

and the readout noise in electrons is

$$\sigma_{ro} = \varepsilon \sigma_d \sqrt{2}$$

9.5 Dark correction, flat fielding and sky correction

If the fringe spatial pattern is stable and if flat fields can be generated without fringing present, it is possible to decouple sky correction and fringe correction and apply a defringing method similar to the one we have developed for optical imaging [RD 11]. This involves creating a series of master fringe frames which are scaled by a suitable factor for each object frame. The scale factors are adjusted to minimise the fringe pattern in the processed frame.

Standard NIR processing recipes often subtract sky first and then flat-field. We can see why this can be advantageous compared with dark-correcting, flat-fielding and sky-correcting by considering the following encapsulation of the problem

$$D(x, y) = ff(x, y)[S(x, y) + F(x, y) + O(x, y) + T(x, y) + dc(x, y)]$$

where $D(x, y)$ is observed, $ff(x, y)$ is the flat-field function, $S(x, y)$ is the sky illumination, $F(x, y)$ is the fringe contribution, $O(x, y)$ is the object contribution, $T(x, y)$ is the thermal contribution, $dc(x, y)$ is the dark current, and without loss of generality we have excluded any explicit wavelength and time-dependence for clarity. Stacking a series of dithered object frames with rejection produces an estimate of the terms

$$\hat{I}(x, y) = ff(x, y)[S(x, y) + F(x, y) + T(x, y)] + dc(x, y)$$

therefore,

$$D(x,y) - \hat{I}(x,y) = ff(x,y)O(x,y)$$

ËĘ

obviating the need for dark-correcting and fringe removal as both separate data gathering requirements and as separate data processing steps; and minimising the effect of systematic and random errors in the flat-field function by removing the largest potential error terms.

In the event that the dark correction stage fails to remove the reset anomaly completely, the residual background variation is analogous to the problem of dealing with short-term variations in sky structure and can be dealt with using the methodology above.

The caveats here of course are that this method may well remove parts of large extended objects, large area nebulosity, and large low surface brightness objects and so on, unless suitable offset skies are used in the sky frame construction. Unfortunately this then opens the door for spatial and temporal variability of the sky background, leaving residual patterns.

The optimal strategy to use depends on the stability of the flat-fields, and the time constants for sky fringe pattern variations, and will be dealt with by assessing these characteristics during commissioning and then invoking suitable processing recipes.

The alternative is to treat the dark correction $dc(x,y)$, flat field $ff(x,y)$, and fringe pattern $F(x,y)$, as accurately known master library frames, in which case data processing involves solving the following variant of the problem

$$D(x,y) = ff(x,y)[S(x,y) + kF(x,y) + O(x,y) + T(x,y)] + dc(x,y)$$

where k is a scale factor to be determined by the fringe-removing algorithm. In this case applying the master frames leads to

$$D'(x,y) = S(x,y) + O(x,y) + T(x,y)$$

reducing the problem to one of detecting astronomical objects on an additive, slowly spatially varying, background. This could be the method of choice for analysing large scale astronomical surface brightness variations.

In the main science recipe it will be possible to define some input frames to act as âĂđoffset skyâĂ§ images for the science images. This is useful in the instance that the science object is a large extended source where a standard âĂđshift and combine with rejectionâĂ§ algorithm would fail to remove flux arising from science objects in the maps to be used in background correction.

9.6 Stripe removal

Data from the VIRCAM detectors show a low level medium frequency stripe pattern. The stripes are perpendicular to the readout direction and are the same for all channels in a detector. The pattern also is repeated on each of the 4 detectors in a line on the focal plane. This means that they can be modelled out by calculating the median of each row (ignoring any bad or object pixels) to form a one dimensional stripe profile. The stripe profile can be averaged over the 4 detectors is normalised to zero median. This ensures that once the stripes are subtracted the median background level with remain the same. Each point in the normalised stripe profile is used to correct the relevant row in the input data. Although there is this four-fold redundancy in the pattern over the detectors in a line, it wonâĂ§t be possible to use this in the summit pipeline as this will be designed to deal with only a single image at time. The RMS of the stripe pattern should be quite small (< 2 counts). If it is significantly higher than this, it is likely the pattern is being affected by the presence of a large extended object. In this case a large scale median filter is applied to the profile to smooth out any large scale variations that have arisen because of such an object.

9.7 Astrometric Calibration

From the optical design studies of VISTA we know that, to a good approximation, the astrometric distortion shows negligible variation with wavelength and is well described by a radially symmetric polynomial distortion model of the form

$$r_{true} = k_1 r + k_3 r^3 + k_5 r^5 + ...$$

where r_{true} is an idealised angular distance from the optical axis, r is the measured distance, and k_1 is the scale at the centre of the field, usually quoted in arcsec/mm. VISTA will have a central field scale, i.e. k_1 value of roughly 17.09 arcsec/mm. It has been determined that the value of k_5 is in fact significant in the case of VIRCAM. Rearranging the preceding equation to a more convenient form gives

$$r_{true} = r'(1 + \frac{k_3}{k_1^3} r'^2 + \frac{k_5}{k_1^5} r'^4) = K_1 r' + K_3 r'^3 + k_5 r'^5$$

where r' is the measured distance from the optical axis in arcsec using the k_1 scale and $K_1 = 1$. If we convert all units to radians the coefficient K_3 is conveniently scaled (in units of radians/radian3) and has a theoretical value of around 42 for VISTA. Measured values place this nearer to 44 with no dependence on wavelength. Analysis of world coordinate results also show a value for $K_5 \approx 10300$.

Although this type of distortion generally presents no problem for accurate calibration of individual pointings, it can lead to various complications when stacking data taken at various locations, e.g. dither sequences. This is caused by the differential non-linear distortions across individual detectors being comparable to, or larger than, the pixel size of the detector. In these cases stacking involves resampling and interpolation of some form. While these are inevitable in combining pointings to form contiguous tiled regions, they may be avoided at earlier stages, such as stacking individual detector dither sequences, by suitably limiting dither offsets and thereby both simplify and speed up the data processing.

The effective scale due to the radial distortion is given by

$$dr_{true}/dr' = 1 + 3K_3 r'^2 + 5K_5 r'^4$$

which describes the local change in relative pixel scale as a function of radial distance. For example, for VISTA at 0.8 degree radius, the differential distortion term is about 2.5in the centre corresponds to a 10.25 arcsec shift at the outer corners of the arrays. For the outer detectors a large fraction of this distortion occurs across individual detectors.

In anticipation of this problem, we will implement a range of interpolation schemes that offer a trade off between maintaining independent pixel noise and resolution degradation.

For further information see the report at

http://www.ast.cam.ac.uk/vdfs/docs/reports/astrom/.

9.8 World Coordinate System

We intend, at least initially, to characterise the WCS using the ZPN projection [RD 7] and [RD 8], i.e. ARC + polynomial distortion, using a 3rd order parameterisation (equation 2.22). The coefficients for this are encoded in the FITS header using the keywords PV2_1, PV2_3 and PV2_5. These are equivalent to the values of K_1 , K_3 , K_5 in equation 2-26.

9.9 Effect of Scale Change in Photometry

In addition to astrometric effects the change in scale as a function of radius also has photometric implications. The aim of conventional flat fielding is to create a flat background by normalising out perceived variations from (assumed) uniformly illuminated frames. If the sky area per pixel changes then this is reflected in a systematic error in the derived photometry.

However, since it much simpler to deal with âĂIJflatâĂİ backgrounds, this problem is either usually ignored or corrected during later processing stages, together with other systematic photometry effects. The effect is simplest to envisage by considering what happens to the area of an annulus on sky when projected onto the detector focal plane. The sky annulus of $2\pi sds$ becomes $2\pi r'dr'$ on the detector, which using a lower order approximation leads to a relative area of

$$(1 + K_3 r'^2)(1 + 3K_3 r'^2) \approx (1 + 4K_3 r'^2)$$

or in other words roughly 4 x the linear scale distortion.

However, since other more unpredictable factors, such as scattered light, will also play a significant role, it is simpler procedurally to bundle all the effects together and correct all the photometric systematics in one operation. The VDFS calibration plan [AD3] describes a procedure for achieving this as an illumination correction.

9.10 Bad Pixel Masks

A bad pixel mask is simply an image that is used to track the location of cosmetic defects on a detector. This is done by creating a data array of zeros for all locations where there is a good pixel and non-zero values where the pixel is know to be bad for whatever reason. We can define the location of bad pixels by taking a series of dome flat field exposures with a range of flux values. We then combine all the dark corrected exposures to form a single master flat. Each input frame can be divided by the master flat in turn and normalised to a median of unity. Any pixel which is above or below an input threshold value (in sigma) can be marked as bad for that image. If a pixel is bad for more than a certain percentage of the input images, then it is defined as a bad pixel.

9.11 Confidence Maps

We define a confidence map c_{ij} as a normalised i.e. $\langle c_{i,j} \rangle_j = 1$ inverse variance weight map denoting the âĂIJconfidenceâĂİ associated with the flux value in each pixel j of frame i. This has the advantage that the map is always finite and can also be used to encode for hot, bad or dead pixels, by assigning zero confidence. Furthermore, after image stacking the confidence map also encodes the effective relative exposure time for each pixel, thereby preserving all the relevant intra-pixel information for further optimal weighting.

The initial confidence map for each frame is derived from regular analysis of the master calibration flat-field and dark frames and is unique for each filter/detector combination due to the normalisation. As such it also encodes individual pixel sensitivities and also allows, for example, vignetted regions to be correctly weighted when combining frames. To use the confidence maps for weighted co-addition of frames then simply requires an overall estimate of the average noise properties of the frame. This can readily be derived from the measured sky noise, in the Poisson noise- limited case, or from a combination of this and the known system characteristics (e.g. gain and readout noise).

All processed frames (stacked individual detectors, tiled mosaiced regions) have an associated derived confidence map which is propagated through the processing chain in the following manner.

Defining the signal s_i in frame i with respect to some reference signal level s_{ref} as $s_i = f_i s_{ref}$, where f_i denotes the relative throughput (which in photometric conditions would be ∞ exposure time), the optimum weight to use for combining the jth pixel of (suitably aligned) frames in order to maximise the signal:-to-noise of sky-limited objects is defined by

$$x'_j = frac \sum w_{ij} x_{ij} \sum w_{ij} w_{ij} = c_{ij} f_i / \sigma_i^2$$

where σ_i^2 is the average noise variance in frame i, x_{ij} is the flux in pixel j on the ith frame and x'_j is the combined output flux. The effective exposure time is that of s_{ref}.

The output confidence map, is therefore given by j

$$c'_j = frac \sum c_{ij} f_i / {\sigma_i^2}^2 {}^2 \sum c_{ij} f_i^2 / \sigma_i^2$$

Special cases of this occur when $f_i = 1$, e.g. equal length exposures in stable photometric conditions, or the more general Poisson noise limited case, when $f_i / \sigma_i^2 = 1$, and the special variant of this when $f_i = 1$. These cases are given below, prior to renormalisation.

$$c'_j = \sum c_{ij} / \sigma_i^2$$

$$c'_j = \frac{\sum {c_{ij}}^2}{\sum c_{ij} f_i}$$

$$c'_j = \sum c_i j$$

9.12 Catalogue Generation

In order to provide quality control, and astrometric and photometric calibration information, it is necessary to generate detected object (i.e. stars, galaxies) catalogues for each target frame. The catalogue generation software (e.g. [RD 12], [RD 9]) will make direct use of the confidence maps for object detection and parameterisation, and will produce the requisite information via the use of standard object descriptors. For completeness we give here a brief description of how this will be accomplished by use of the following steps:

estimate the local sky background over the field and track any variations at adequate resolution to eventually remove them;

detect objects/blends of objects and keep a list of pixels belonging to each blend for further analysis;

parameterise the detected objects, i.e. perform astrometry, photometry and some sort of shape analysis.

9.12.1 Background analysis and object detection

The possibly-varying sky background is estimated automatically, prior to object detection, using a combination of robust iteratively-clipped estimators.

Any variation in sky level over the frame will be dealt with by forming a coarsely sampled background map grid. Within each background grid pixel, typically equal to 64 x 64 image pixels, an iteratively k-sigma clipped median value of "sky" will be computed based on the histogram of flux values within the grid pixel zone. A robust estimate of sigma can be computed using the Median of the Absolute Deviation (MAD) from the median (e.g. [RD 13]). This will then be further processed to form the frame background map (e.g. [RD 9]).

After removing the, possibly, varying background component, a similar robust estimate of the average sky level and sky noise per pixel can be made. This forms part of the quality control measures and also helps to robustly determine the detection threshold for object analysis.

Individual objects will be detected using a standard matched filter approach (e.g. [RD 12]). Since the only images difficult to locate are those marginally above the sky noise, assuming constant noise is a good approximation (after factoring in the confidence map information) and the majority of these objects will have a shape dominated by the point spread function (PSF), which thereby defines the filter to use.

9.12.2 Image parameterisation

The following image parameters can be computed efficiently and are directly used as part of the image quality control and calibration analysis.

Isophotal Intensity - the integrated flux within the boundary defined by the threshold level; i.e. the 0th object moment

$$I_{iso} = \sum I(x_i, y_i$$

For Gaussian images, this is related to the total intensity by the factor $(1 - I_t/I_p)^(- 1)$, where I_p is the peak flux and I_t the threshold level (all relative to sky).

Position - computed as an intensity-weighted centre of gravity; i.e. 1st moment

$$x_0 = \sum x_i I(x_i, y_i) / \sum I(x_i, y_i)$$

$$y_0 = \sum y_i I(x_i, y_i) / \sum I(x_i, y_i)$$

Covariance Matrix - the triad of intensity-weighted 2nd moments is used to estimate the eccentricity/ellipticity, position angle and intensity-weighted size of an image

$$\sigma_{xx} = \sum (x_i - x_0)^2 I(x_i, y_i) / \sum I(x_i, y_i)$$

$$\sigma_{xy} = \sum (x_i - x_0)(y_i - y_0) I(x_i, y_i) / \sum I(x_i, y_i)$$

$$\sigma_{yy} = \sum (y_i - y_0)^2 I(x_i, y_i) / \sum I(x_i, y_i)$$

The simplest way to derive the ellipse parameters from the 2nd moments is to equate them to an elliptical Gaussian function having the same 2nd moments. It is then straightforward to show (e.g. [RD 14]) that the scale size, $\sqrt{\sigma_{rr}}$, is given by $\sigma_{rr} = \sigma_{xx} + \sigma_{yy}$; the eccentricity,

$$ecc = \frac{\sqrt{(\sigma_{xx} - \sigma_{yy})^2 + 4\sigma_{xy}^2}}{\sigma_{rr}}$$

and the position angle, θ, is defined by

$$tan(2\theta) = 2\sigma_{xy} / (\sigma_{yy} - \sigma_{xx})$$

The ellipticity, e, which is simpler to interpret for estimating potential image distortions (e.g. trailing), is related to the eccentricity by

$$e = 1 - \sqrt{(1 - ecc)/(1 + ecc)}$$

Areal Profile - a variant on the radial profile, which measures the area of an image at various intensity levels. Unlike a radial profile, which needs a prior estimate of the image centre, the areal profile provides a single pass estimate of the profile

$$ArealProfile \rightarrow T + p_1, T + p_2, T + p_3, ...T + p_m$$

where $p_j; j = 1, ...m$ are intensity levels relative to the threshold, T, usually spaced logarithmically to give even sampling.

The peak height, I_p , is a useful related addition to the areal profile information and is defined as

$$I_p = max[I(xi, yi)]_i$$

or alternatively measured by extrapolation from the areal profile if the image is saturated. The areal profile provides a direct method to estimate the seeing of objects in an image by enabling the average area of stellar images (point sources) at half the peak height, $\langle A \rangle$, to be estimated. The seeing, or FWHM, is then given by $FWHM = 2\sqrt{\langle A \rangle / \pi}$.

Finally a series of aperture fluxes are required for object morphological classification (see below).

Aperture flux is defined as the integrated flux within some radius r of the object centre

$$I_{ap}(r) = \sum I_i - Nxsky$$

Where boundary pixels are weighted pro-rata (soft-edged aperture photometry). A series of these is used to define the curve-of-growth, $I_{ap}(r) - v - r$, for each object.

9.12.3 Morphological classification

The object detection software will produce a series of background-corrected flux measures for each object in a set of âĂIJsoft-edgedâĂİ apertures of radius $r/2, r/\sqrt{2}, r, \sqrt{2}r, 2r......12r$, where r is typically fixed as the median seeing for the site+telescope+camera. The average curve-of-growth for stellar images is used to define automatically an aperture correction for each aperture used and also forms the basis for object morphological classification (required for isolating stellar images for seeing and trailing quality control).

The curve-of-growth of the flux for each object is compared with that derived from the (self-defining) locus of stellar objects, and combined with information on the ellipticity of each object, to generate the classification statistic. This statistic is designed to preserve information on the âĂIJsharpnessâĂİ of the object profile and is re- normalised, as a function of magnitude, to produce the equivalent of an $N(0, 1)$ measure, i.e. a normalised Gaussian of zero-mean and unit variance. Objects lying within $2 - 3\sigma$ are generally flagged as stellar images, those below 3σ (i.e. sharper) as noise-like, and those above $2 - 3\sigma$ as non-stellar.

A by-product of the curve-of-growth analysis is the estimate of the average PSF aperture correction for each detector.

9.13 Photometric Zeropoint

For the purposes of quality control (e.g. sky transparency and system performance) a primary photometric zeropoint will be determined for each observation by direct comparison of instrumental magnitudes with the magnitudes of 2MASS stars. An alternative cross-check on the photometric calibration will be applied retrospectively given a complete night of observations including regular exposures in VISTA photometric standard fields.

The internal gain-correction, applied at the flat-fielding stage, should place all the detectors on a common zeropoint system (at least to first order i.e. ignoring colour equation variations between the detectors), and given a stable instrumental setup, the apparent variation of zeropoint then directly measures the change in âĂIJextinctionâĂİ without the need to rely solely on extensive standard field coverage over a range in airmass. Therefore for any given observation of a star in a particular passband:

$$m^{cal} = m^{inst} + ZP - \kappa(X - 1 = m^{std} + ce^{std} + \varepsilon)$$

where ZP is the zeropoint in that passband (i.e. the magnitude at airmass unity which gives 1 count/second at the detector), m^{cal} is the calibrated instrumental magnitude, m^{inst} is the measured instrumental magnitude ($-2.5log10[counts/sec]$), κ is the extinction coefficient, X is the airmass of the observation, ce^{std} is the colour term to convert to the instrumental system, and ε is an error term. This assumes that the second-order extinction term and colour-dependency of κ are both negligible. By robustly averaging the zeropoints for all the matching stars on the frame an overall zeropoint for the observation can be obtained.

Typically, the zeropoint of the instrument + telescope system should be stable throughout the night. Long-term decreases in the sensitivity of the instrument, and hence a decreasing ZP, could be caused by for example the accumulation of dust on the primary mirror.

On photometric nights the extinction coefficient Îž should be constant in each passband. The extinction Îž can be monitored through each night either by assuming the true instrumental zeropoint only varies slowly as a function of time (and using the individual 2MASS calibrations to monitor it) or by making measurements over a range of airmass.

9.14 Illumination Correction

The two methods of determination of illumination correction differ in that the first described below requires either a rich standard star field or a series of fields with known photometry, but the second can be used before such information is available.

9.14.1 Standard star fields

Errors in the large scale structure of the illumination of the flat fields used in signature removal can cause position dependent systematic errors in photometry. This can be a result of a varying scattered light profile between twilight (nominally when the flat field exposures would have been made) and the time when the observation was done. We can map this out by first dividing an observation of a rich photometric standard field into cells or by dividing a series of calibrator fields from, for example, 2MASS into cells. For each cell we define a median zero point of all the stars in that cell:

$$zp_j = \langle m^{cal} - m^{inst} \rangle$$

(It is safe to ignore the extinction term for this exercise.) The illumination correction is then defined for each cell as:

$$ic_j = \langle zp \rangle - zp_j$$

where $\langle zp \rangle$ is the median value of zp_j over all the cells. This is defined such that a star in the jth cell is calibrated by:

$$m^{cal} = m^{inst} + ZP + \kappa(X - 1) - ic_j$$

9.14.2 Mesostep analysis

We assume that the spatial sensitivity of each detector can be approximated by a polynomial surface, i.e. a magnitude offset as a function of (x, y) measured from the centre of the detector, e.g.

$$ZP(x, y) = \sum a_{hk} x^h y^k$$

For example, in quadratic form, at positions i and j:

$$ZP(x_i, y_i) = a_{00} + a_{10}x_i + a01y_i + a_{20}x_i^2 + a_{11}x^i y^i + a_{02}y_i^2$$

$$ZP(x_j, y_j) = a_{00} + a_{10}x_j + a01y_j + a_{20}x_j^2 + a_{11}x^j y^j + a_{02}y_j^2$$

The difference in sensitivity/zeropoint between two positions i and j is then:

$$\Delta ZP(x_i, x_j, y_i, y_j) = a_{10}(x_i - x_j) + a_{01}(y_i - y_j) + a_{20}(x_i^2 - x_j^2) + a_{11}(x_i y_i - x_j y_j) + a_{02}(y_i^2 - y_j^2)$$

If we make two observations of the same star at offset positions $i(x_i, y_i)$ and $j(x_j, y_j)$, we sample this function such that the difference in magnitude measured is Δm_{ij} then:

$$\Delta m_{ij} = \Delta ZP(x_i, x_j, y_i, y_j)$$

In the simplest case, observing the same star in a number of different places would allow one to measure the Δm_{ij} as a function of (x_i, y_i) and (x_j, y_j). One could then fit a polynomial using least-squares and solve for the a_{hk}. The multiple observations of multiple stars in a grid across the array ensure we can solve for the polynomial coefficients accurately.

A Installation

This chapter gives generic instructions on how to obtain, build and install the OmegaCAM pipeline. Even if this chapter is kept up-to-date as much as possible, it may not be fully applicable to a particular release. This might especially happen for patch releases. One is therefore advised to read the installation instructions delivered with the OmegaCAM pipeline distribution. These release-specific instructions can be found in the file `README` located in the top-level directory of the unpacked OmegaCAM pipeline source tree. The supported platforms are listed in Section **??**. It is recommended reading through Section **??** before starting the installation.

A bundled version of the OmegaCAM 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 utilisation of the GNU build tools should allow to build and install the OmegaCAM pipeline on a variety of UNIX platforms, but it has only been verified on the VLT target platforms:

- Linux (glibc 2.1 or later),

using the GNU C compiler (version 3.2 or newer).

A.2 Building the OmegaCAM pipeline

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

A.2.1 Requirements

To compile and install the OmegaCAM pipeline one needs:

- the GNU C compiler (version 3.2 or later),
- the GNU `gzip` data compression program,
- a version of the `tar` file-archiving program, and,
- the GNU `make` utility.

An installation of the Common Pipeline library (CPL) must also be available on the system. Currently the CPL version 2.0 or newer is required. The CPL distribution can be obtained from `http://www.eso.org/cpl` .

Please note that CPL itself depends on an existing cfitsio installation. The cfitsio sources are available from the CPL download page or directly from the cfitsio homepage at `http://heasarc.nasa.gov/fitsio/fitsio.html` . In conjunction with CPL 4.0 cfitsio 2.5.10 must be used.

In order to run the OmegaCAM pipeline recipes a front-end application is also required. Currently there are two such applications available, a command-line tool called *EsoRex* and the Java based data file organizer, *Gasgano* , which provides an intuitive graphical user interface. At least one of them must be installed. The *EsoRex* and *Gasgano* packages are available at `http://www.eso.org/cpl/esorex.html` and `http://www.eso.org/gasgano` respectively.

For installation instructions of any of the additional packages mentioned before please refer to the documentation of these packages.

A.2.2 Downloading the OmegaCAM pipeline source distribution

From the ESO ftp server, `ftp://ftp.eso.org/pub/cpl/ocam` , the latest release of the OmegaCAM pipeline distribution is available.

The OmegaCAM pipeline sources are distributed as a gzipped tar archive named like `ocam-X.Y.Z.tar.gz` , where `X`and `Y` are the major and minor release numbers, and `Z` indicates the patch level (which might be missing if no patch has been released).

A.2.3 Compiling and installing the OmegaCAM pipeline

It is recommended to read through this section before starting with the installation.

The OmegaCAM pipeline distribution kit 1.0 contains:

ocam-manual-1.0.pdf	The OmegaCAM pipeline manual
cpl-4.1.0.tar.gz	CPL 2.1.1
esorex-3.7.0.tar.gz	esorex 3.5.1
gasgano-2.2.3-Linux.tar.gz	GASGANO 2.2.3 for Linux
gasgano-2.2.3-SunOS.tar.gz	GASGANO 2.2.3 for SunOS
ocam-0.5.0.tar.gz	OmegaCAM pipeline 0.1.3
ocam-calib-0.1.3.tar.gz	OmegaCAM calibration files 0.1.3

Here is a description of the installation procedure:

1. First, if an appropriate version of CPL (c.f. section **??**) does not already exist on the system, compile and install the CPL libraries and their dependencies. For detailed instructions on how to install the CPL libraries please refer to the CPL documentation.

2. Unpack the OmegaCAM pipeline sources in a choosen directory using

   ```
   $ zcat -d ocam-X.Y.Z.tar.gz | tar -xf -
   ```

   at the system prompt. This will create a directory called `ocam-X.Y.Z` containing the source tree.

3. Run the automatic installation script `setup` which is located in the top-level directory of the OmegaCAM pipeline source tree.

   If the CPL has not been installed into one of the system's standard directories, the configuration script must be told where the dependent libraries can be found. This is done by defining the environment variables `CPLDIR`.

   For example, if the CPL has been installed into `/somewhere` , (and therefore the CPL header files are located in `/somewhere/include` and the libraries are in `/somewhere/lib`) , the path assigned to the variable `CPLDIR` must be `/somewhere` , *i.e.*, the root directory of the CPL installation. The same is true for the environment variable corresponding to qfits.

   The `setup` script takes one argument which is the location where the OmegaCAM pipeline should be installed in your system. All its components will be located in the directory tree rooted at this path.

   The following example assumes that the variable `CPLDIR` is properly set or the CPL has been installed into one of the system's standard directories. This should keep the following example commands as simple as possible.

   Please note also that the usage of the default installation prefixes in the example below is just for demonstration purposes. Any directory for which one has write access can be used, although it is **not recommended** to use the distribution's source directory as the installation's target directory.

   The simplest way to set up the package is to run the following command at the system prompt from the source-tree's top-level directory:

   ```
   ./setup /usr/local/pipeline
   ```

   This script will install the OmegaCAM pipeline and all other necessary external programs and configuration files under the path given in the command line. It will create a `/usr/local/pipeline/bin` , and a `/usr/local/pipeline/config` directories containing the executables and configuration files for these programs.

After the installation has been completed the source tree is no longer needed and can be removed.

A.3 Configuring the pipeline recipe front-end applications

In this section an outline is given how to set up the recipe front-ends *EsoRex* and *Gasgano* so that the just installed OmegaCAM pipeline recipes can be executed by these applications.

For detailed instructions on how to configure the two currently available front-end applications, *EsoRex* and *Gasgano* , please refer to their documentation, available at `http://www.eso.org/cpl/esorex.html` and `http://www.eso.org/gasgano` respectively.

In the following it is assumed that the OmegaCAM pipeline was installed as described in Section **??**, *i.e.* , that the OmegaCAM recipes have been copied into `/usr/local/pipeline/lib/ocam/plugins/ocam-X.Y.Z` (`X`, `Y` and `Z` indicate the version number of the recipes).

A.3.1 Setting up the EsoRex command-line tool

The general syntax for the *EsoRex* front-end is the following:

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

In order to execute a recipe, *EsoRex* must be told where the recipes can be found. This location can be passed to the tool using its command line option `--recipe-dir` followed by the complete path to the recipes. In this scenario the command to be executed at the shell's prompt will look like:

```
$ esorex --recipe-dir \
> /usr/local/pipeline/lib/ocam/plugins/ocam-X.Y.Z
```

However, the path to the recipe location(s) can also be set in the *EsoRex* configuration file. If an *EsoRex* configuration file does not already exist, it can be created by executing the command

```
$ esorex --create-config --recipe-dir \
> /usr/local/pipeline/lib/ocam/plugins/ocam-X.Y.Z
```

In addition to just creating the configuration file in its standard location `$HOME/.esorex/esorex.rc` the path to the recipes is also added by this command. This way for updating the configuration, can be repeated whenever a new version of the recipes is installed. It will replace the path to the recipes in the *EsoRex* configuration file.

If *EsoRex* has already been used for running recipes from other instruments, the path to the OmegaCAM recipes must be appended to the existing configuration file. To do this edit the configuration file `$HOME/.esorex/esorex.rc` . Go to the entry starting with `esorex.caller.recipe-dir` . This is a colon separated list of directories searched by *EsoRex* for recipes. To add the OmegaCAM recipes just append the OmegaCAM recipe installation directory, separated by a colon (:), to the end of this list.

To verify the updated configuration execute the following command at the shell's prompt:

```
esorex --recipes
```

This should display a list of the available recipes on the terminal screen.

At the *EsoRex* homepage, `http://www.eso.org/cpl/esorex.html` , a detailed description of the application can be found.

A.3.2 Setting up Gasgano

The OmegaCAM recipe set can be incorporated into *Gasgano*'s configuration using the *Preferences* dialog from the *File* menu. Select the tab labeled *Recipe Configuration* and press the *Add Recipe* button. A file selection dialog pops up where the recipes to add can be selected. The selection has to be confirmed and *Gasgano* must be restarted to activate the new recipe configuration. Now the recipes are seamlessly integrated into the application and the files to process can be passed to the recipes using drag'n drop (for details please have a look into the *Gasgano* User Manual [11]).

B Abbreviations and acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CalibDB	Calibration Database
CFITSIO	FITS IO Library
CPL	Common Pipeline Library
DFO	Data Flow Operations department
DFS	Data Flow System
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
PSO	Paranal Science Operations
QC	Quality Control
RON	Read Out Noise
SOF	Set Of Frames
SDD	Software Development Division
UT	Unit Telescope
VLT	Very Large Telescope

C Troubleshooting Guide