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

HAWK-I Pipeline User Manual

VLT-MAN-ESO-19500-4407

Issue 1.6

Date March 2011

Prepared:	Y. Jung, C. E. García Dabó	March 2011	
	Name	Date	Signature
Approved:	P.Ballester		
	Name	Date	Signature
Released:	M. Peron		
	Name	Date	Signature

No content on this page

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	01/04/2008	All	For HAWK-I 1.2.0
1.1	30/10/2008	All	Changes up to HAWK-I 1.4.2
1.2	20/07/2009	All	Changes up to HAWK-I 1.5.3
1.3	02/05/2010	All	Changes up to HAWK-I 1.7.0
1.4	11/05/2010	Quickstart	Fixed step by step procedure

No content on this page

Contents

1 Introduction

1.1 Purpose

The HAWK-I pipeline is a subsystem of the *VLT Data Flow System* (DFS). It is used in three different environments. First, it is used within ESO in *Data Flow Operations* (DFO) for the validation of scientific exposures, data quality control and the generation of master calibration data. Second, the *Paranal Science Operations* (PSO) uses it for quick-look assessment of data and the reduction of scientific exposures. Finally, the HAWK-I pipeline recipes are made public to the user community, to allow a more fine-tuned processing of the data. The purpose of this document is to describe a typical HAWK-I data reduction procedure using the HAWK-I pipeline.

This manual is a complete description of the data reduction recipes implemented by the HAWK-I pipeline, reflecting the status of the HAWK-I pipeline version 1.8.1.

1.2 Acknowledgments

Nancy Ageorges, Markus Kissler-Patig, Fernando Selman and Giovanni Carraro, instrument scientists of HAWK-I, and John Pritchard, QC group contact person have provided valuable feed-back on both the pipeline itself and its documentation, and made it possible to release it publicly.

L. R. Bedin has provided very precise distortion measurements that are used by the pipeline to correct the distortion, as well as very useful support during the commissioning phases. Clear documentation of the distortion measurements can be found in this document.

Thanks to W. Freudling for his input regarding distortion, background computation and many other insightful suggestions.

P. Ballester, W. Freudling, M. Romaniello and D. M. Bramich have provided comments on the documentation.

1.3 Scope

Updated versions of the present document may be found on [12]. Quality control information are at [2].

Additional information on the Common Pipeline Library (CPL) and *EsoRex* can be found respectively at [7], [10]. The Gasgano tool is described in [11]. A description of the instrument is in [3]. The HAWK-I instrument user manual is in [4].

1.4 Reference and applicable documents

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

[2] ESO/DMO/DFO, *http://www.eso.org/observing/dfo/quality/pipeline-status.html*. *HAWK-I Pipeline Current Status*. 13

[3] ESO/INS, *http://www.eso.org/sci/facilities/paranal/instruments/hawki*. *HAWK-I instrument home page*. 13, 16

[4] ESO/INS, *http://www.eso.org/sci/facilities/paranal/instruments/hawki/doc*. *HAWK-I User Manual*. 13

[5] ESO/SDD/DFI, *http://www.eso.org/sci/data-processing/software/sampo/reflex//*. *Reflex User Manual*. VLT-SPE-XXX-XXX. 15

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

[7] ESO/SDD/DFS, *http://www.eso.org/cpl/*. *CPL home page*. 13

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

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

[10] ESO/SDD/DFS, *http://www.eso.org/cpl/esorex.html*. *ESOREX home page*. 13, 15, 22

[11] ESO/SDD/DFS, *http://www.eso.org/gasgano/*. *Gasgano User's Manual*. VLT-PRO-ESO-19000-1932. 13, 15, 19, 29

[12] ESO/SDD/DFS, *http://www.eso.org/pipelines*. *HAWK-I Pipeline Web Page*. 13, 95

[13] IRAF, *http://iraf.noao.edu/iraf/ftp/iraf/extern-v212/xdimsum/xdimsum.readme*. *Xdimsum IRAF package*. 35

[14] Cushing M. C. Vacca W. D. and Rayner J. T. *Nonlinearity Corrections and Statistical Uncertainties Associated with Near-Infrared Arrays*. PASP 116:352-361 2004. 64

2 Overview

In collaboration with instrument consortia, the Pipeline Department (PSD) of the Software Development Division (SDD) is implementing data reduction pipelines that have the following three main purposes:

Science product creation: using pipeline-generated master calibration products, science products are produced for the supported instrument modes (*e.g.*, combined HAWK-I jitter stacks; bias-corrected, flat-fielded FORS images, wavelength-calibrated UVES spectra). The HAWK-I pipeline aims to provide scientific grade data products, although there are still on-going developments in the areas of distortion and sky computation (see D).

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).

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, from *Reflex* or from *Gasgano*.

ESO offers three front-end applications for launching pipeline recipes, *Reflex* [5], *Gasgano* [11] and *EsoRex* [10]. The two latter applications are included in the pipeline distribution (see Appendix C). These applications can also be downloaded separately from `http://www.eso.org/reflex`, `http://www.eso.org/gasgano` and `http://www.eso.org/cpl/esorex.html`, respectively. An illustrated introduction to *Reflex* and *Gasgano* is provided in the "Quick Start" Section of this manual (see Section **??**).

The HAWK-I instrument and the different types of HAWK-I raw frames and auxiliary data are described in Sections 3, 5, and 6.

A brief introduction to the usage of the available reduction recipes using *Gasgano* or *EsoRex* is presented in Section **??**.

An overview of the data reduction, the relevant input data, and the recipes involved in the calibration cascade is provided in Section 7.

More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe are given in Section 8.

More detailed descriptions of the data reduction algorithms used by the individual pipeline recipes can be found in Section 9.

In Appendix C the installation of the HAWK-I pipeline recipes is described and in Appendix E we supply a list of the abbreviations and acronyms used in this document.

3 HAWK-I Instrument Description

HAWK-I is a cryogenic wide-field imager installed at the Nasmyth A focus of UT4. The on-sky field of view is 7.5x7.5 arc minutes, with a cross-shaped gap of 15 arc seconds between the four HAWAII 2RG 2048x2048 pixels detectors. The pixel scale is $0.106\ arcsec\ pix^{-1}$. The instrument is offered with 10 observing filters placed in two filter wheels: 4 broad band filters (Y(Z), J, H and K) and 6 narrow band filters (Bracket gamma, CH4, H2, 1.061 microns, 1.187 microns and 2.090 microns).

Figure 3.0.1 shows a picture of the instrument mounted on the fourth unit telescope on the VLT.

Figure 3.0.1: *Picture of HAWK-I.*

Please refer to the instrument web page [3] for more detailed informations about the instrument.

4 Quick start

This section describes how to make immediate usage of the HAWK-I pipeline recipes.

4.1 HAWK-I pipeline recipes

The current HAWK-I pipeline is based on a set of 7 calibration recipes, 1 science monolithic recipe, 9 modular science recipes and 2 utility recipes.

The 7 calibration recipes are:

hawki_cal_dark: Recipe for creation of master darks.

hawki_cal_flat: Recipe for creation of master twilight flat.

hawki_cal_zpoint: Compute zero point solution from standard stars.

hawki_cal_illum: Recipe for the computation of illumination correction for accurate photometry.

hawki_cal_distortion: Recipe to compute the distortion solution of the instrument using autocalibration images.

hawki_cal_lingain: Gets a characterization of the HAWK-I detector.

hawki_tec_filtchk: Filter monitoring recipe (only for observatory use).

The science monolithic recipe is:

hawki_sci_jitter: Recipe to reduce jitter observations up to the final reduction stage.

The 9 modular science recipes are:

hawki_step_basic_calibration: Dark, flat and bad pixel mask calibration recipe.

hawki_step_compute_bkg: Recipe to compute background of a series of jitter science images.

hawki_step_subtract_bkg: Subtraction of background recipe.

hawki_step_apply_dist: Recipe to correct science images from distortion of the instrument.

hawki_step_refine_offsets: Recipe to compute accurate offsets of a jitter series using correlation points.

hawki_step_combine: Recipe to combine (stack) a series of jitter science images.

hawki_step_detect_obj: Recipe to detect objects in an image.

hawki_step_stitch: Recipe to stitch together the four HAWK-I detectors in a single image.

hawki_step_stats: Basic statistics computation recipe.

The 2 utility recipes are:

hawki_util_stdstars: Standard stars catalog creation

hawki_util_gendist: Distortion map creation

4.2 An introduction to *Gasgano* and *EsoRex*

Before being able to call pipeline recipes to process a set of data, the data must be correctly classified, and associated with the appropriate calibrations. *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. Since all the required information is stored in the FITS headers, data association is based on a set of header keywords (called "association keywords") and the process is specific to each type of calibration.

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

An instrument pipeline consists of a set of data processing modules that can be called from different host applications, namely:

- *Gasgano* is a data management tool that simplifies the data organization 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.
- *EsoRex* is a command line tool used to run the pipeline recipes. *EsoRex* commands can be easily scripted.
- The Paranal observatory implements automatic data management tools that trigger the execution of pipeline recipes. This aspect is not covered in this manual.

4.2.1 Using *Gasgano*

Another convenient tool useful for familiarising oneself with the HAWK-I pipeline recipes and their usage is the graphical user interface *Gasgano*. It provides a complete graphical user interface for data browsing, classification and association, and offers several other utilities such as easy access to pipeline recipes, documentation and the preferred data display tools.

Gasgano can be started from the system prompt in the following way:

```
gasgano &
```

The *Gasgano* main window will appear. In Figure **??**, a view of a set of HAWK-I data is shown as an example. *Gasgano* can be configured to point 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).

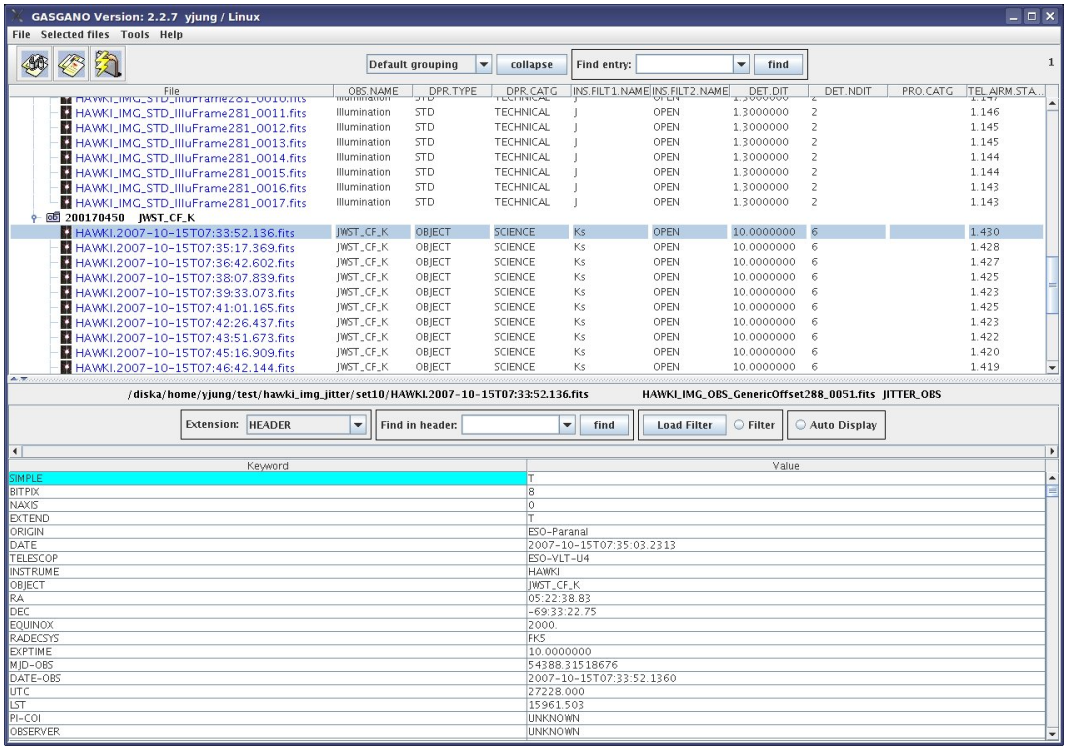


Figure 4.2.1: *The Gasgano main window.*

More information about a single frame can be obtained by clicking on its name: the corresponding FITS file header will be displayed in 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 processing by the appropriate 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 modified on the *Parameters* panel (on top). The window contents may be saved for later use by selecting the *Save Current Settings* entry from the *File* menu.

At this point the recipe can be launched by clicking 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 in the *Output Frames* panel, where they can be easily viewed and located back in the *Gasgano* main window.

Please refer to the *Gasgano User's Manual* [11] for a more complete description of the *Gasgano* interface.

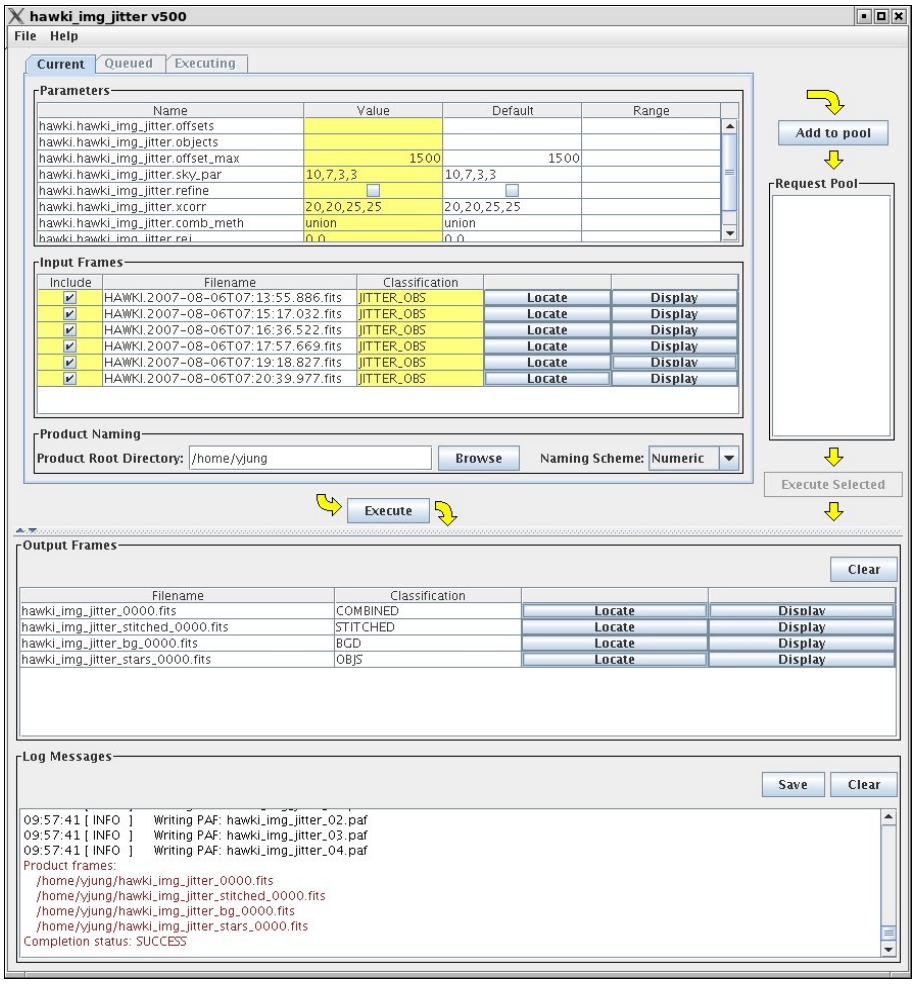


Figure 4.2.2: *The Gasgano recipe execution window.*

4.2.2 Using *EsoRex*

EsoRex is a command line utility for running pipeline recipes. It may be embedded by users into data reduction scripts for the automation of processing tasks. A disadvantage of *EsoRex* is that it does not 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 **??**). The user is also responsible for defining the input set-of-frames and the appropriate configuration parameters for each recipe run:

The set-of-frames: Each pipeline recipe is run on a set of input FITS data files. When using *EsoRex*, the file names must be listed together with their DO category in an ASCII file called the *set-of-frames* (SOF), which is required when launching a recipe [1].

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

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

```
HAWKI_IMG_STD_StandardStar281_0027.fits ZPOINT
HAWKI_IMG_STD_StandardStar281_0028.fits ZPOINT
HAWKI_IMG_STD_StandardStar281_0029.fits ZPOINT
HAWKI_IMG_STD_StandardStar281_0030.fits ZPOINT
/home/user/hawki-calib/ima/HI_GSSC_081103A_stdstars_cat.fits STDSTARS_CATS
```

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

Using *Gasgano* as an interface to the pipeline recipes will always ensure a correct classification of all the data frames and the appropriate assignation of the DO category to each one (see Section 4.2.1).

EsoRex syntax: The basic syntax for using *EsoRex* is as follows:

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

To obtain more information on how to customise *EsoRex*, run the command:

```
esorex --help
```

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

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

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

```
esorex --recipes
```

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

```
esorex --params recipe_name
```

To obtain a brief description of each parameter for a specific pipeline recipe, execute the command:

```
esorex --help recipe_name
```

To obtain more details about a given recipe, issue the following 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.[2] The configuration files are normally generated in the directory `$HOME/.esorex`, and have the same name as the recipe to which they are related, with the file name extension `.rc`.

The command

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

generates a default configuration file **recipe_name.rc** in the directory `$HOME/.esorex`[3].

For instance, *EsoRex* uses configuration file named `hawki_sci_jitter.rc` to set the default parameters for the recipe *hawki_sci_jitter*. This file can be generated with the command:

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

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

[2] The *EsoRex* recipe configuration file corresponds to the *Parameters* panel of the *Gasgano* recipe execution window (see Figure **??**).
[3] If a number of recipe parameters are specified on the command line, the given values will be used in the created configuration file.

```
# --xcorr
# Cross correlation search and measure sizes.
hawki.hawki_sci_jitter.xcorr=20,20,25,25
```

In this example, the parameter `hawki.hawki_sci_jitter.xcorr` is set to the value `20,20,25,25`. In the configuration file generated by *EsoRex*, one or more comment lines are added containing information about the possible values of the parameter, and an alias that could be used as a command line option.

The recipes provided by the HAWK-I 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 a 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.

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

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

```
esorex hawki_sci_jitter hawki_sci_jitter.sof
```

The recipe parameters may be modified either by directly editing the 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 *hawki_sci_jitter* recipe `--xcorr` parameter to `10,10,25,25`, the following command should be entered:

```
esorex hawki_sci_jitter --xcorr="10,10,25,25" hawki_sci_jitter.sof
```

For more information on *EsoRex*, see [10].

4.3 Data demonstration package

The following sections will guide the user step-by-step through the reduction of the data in the demonstration package provided alongside the pipeline. In this Section, we will briefly describe the contents of the demonstration package. See Section **??** for information about how to obtain and install the the demo data package.

The directory structure is as follows:

- `raw`. This directory contains raw data from the telescope and has three subdirectories:
  - `DARK`. This directory contains the test dark frames
  - `FLAT`. This directory contains the test flat frames

  - `JITTER`. This directory contains the test science data taken in jitter mode.
  - `ZPOINT`. This directory contains the test standard star observations that will be used to calculate the zero point of the instrument.

- `ref`. This directory includes the reference files required for the reduction. It includes the distortion solutions and a subset of the 2MASS catalogue.

- `sof`. This directory contains files that are used in this quick start guide to run the pipeline using *EsoRex*.

The demonstration package contains three science data sets from the Bullet Cluster observed with narrow band filter NB1060 in December 2008 under program 181.A-0485(B). The three jitter data sets, without calibrations, are:

- `HAWKI.2008-12-02T05:54:33.406.fits HAWKI.2008-12-02T05:59:51.444.fits`
  `HAWKI.2008-12-02T06:05:09.478.fits HAWKI.2008-12-02T06:10:27.521.fits`
  `HAWKI.2008-12-02T06:15:45.577.fits HAWKI.2008-12-02T06:21:03.628.fits`
  `HAWKI.2008-12-02T06:26:21.691.fits HAWKI.2008-12-02T06:31:39.737.fits`
  `HAWKI.2008-12-02T06:36:57.808.fits HAWKI.2008-12-02T06:42:15.893.fits`
  `HAWKI.2008-12-02T06:47:33.956.fits`

- `HAWKI.2008-12-03T04:33:33.722.fits HAWKI.2008-12-03T04:38:51.789.fits`
  `HAWKI.2008-12-03T04:44:09.857.fits HAWKI.2008-12-03T04:49:27.884.fits`
  `HAWKI.2008-12-03T04:54:45.910.fits HAWKI.2008-12-03T05:00:03.946.fits`
  `HAWKI.2008-12-03T05:05:21.970.fits HAWKI.2008-12-03T05:10:40.034.fits`
  `HAWKI.2008-12-03T05:15:59.321.fits HAWKI.2008-12-03T05:21:17.349.fits`
  `HAWKI.2008-12-03T05:26:35.377.fits`

- `HAWKI.2008-12-03T05:33:00.248.fits HAWKI.2008-12-03T05:38:19.574.fits`
  `HAWKI.2008-12-03T05:43:37.640.fits HAWKI.2008-12-03T05:48:55.665.fits`
  `HAWKI.2008-12-03T05:54:13.691.fits HAWKI.2008-12-03T05:59:31.755.fits`
  `HAWKI.2008-12-03T06:04:49.781.fits HAWKI.2008-12-03T06:10:07.849.fits`
  `HAWKI.2008-12-03T06:15:25.877.fits HAWKI.2008-12-03T06:20:43.929.fits`
  `HAWKI.2008-12-03T06:26:03.205.fits`

In the following examples we will show the user how to reduce just the first data set.

4.4 Example of jitter data modular reduction using Esorex

Jittered science observations can be reduced using the set of modular HAWK-I recipes, which allow in depth control of the recipes as well as inspection of intermediate products. If you are using *EsoRex* and your goal is to obtain science grade data products we recommend employing this kind of reduction. Here we will present a step-by-step walkthrough of how to reduce data with the modular recipes.

The procedure implements a two-pass reduction to improve the estimation of the background. In the second pass the objects detected in the combined image are masked before computing the background.

1. Follow the instructions in section **??** to install the HAWK-I pipeline and the test data packages.

2. Make sure that esorex keeps the pipeline names using a configuration variable (it is also possible to edit `esorex.rc` or use a command line option:

```
export ESOREX_SUPPRESS_PREFIX=TRUE   #(bash, ksh, zsh)
setenv ESOREX_SUPPRESS_PREFIX TRUE  #(csh, tcsh)
```

3. Point the `$HAWKI_DATA` variable to the location of the test data. Use one of the following commands, depending on your shell:

```
export HAWKI_DATA=/path/hawki-demo-0.2   #(bash, ksh, zsh)
setenv HAWKI_DATA  /path/hawki-demo-0.2  #(csh, tcsh)
```

4. Create a directory to store the products and change to the top level directory:

```
user@host# mkdir $HAWKI_DATA/products
user@host# cd $HAWKI_DATA/
```

5. Create the master dark. One might want to have a look to the input frames first:

```
cat  $HAWKI_DATA/sof/hawki_cal_dark.sof
raw/DARK/HAWKI.2008-12-02T09:19:24.807.fits DARK
raw/DARK/HAWKI.2008-12-02T09:14:16.855.fits DARK
raw/DARK/HAWKI.2008-12-02T09:09:08.909.fits DARK
```

   And then execute the *hawki_cal_dark* recipe:

```
esorex --output-dir=products hawki_cal_dark \
   sof/hawki_cal_dark.sof
```

   The most important files created are:

   - `products/hawki_cal_dark.fits`. The master dark frame.
   - `products/hawki_cal_dark_bpmdark.fits`. The bad pixel mask created from the dark frames.

6. Create the master flat.

```
esorex --output-dir=products hawki_cal_flat \
   sof/hawki_cal_flat.sof
```

The most important files created are:

- `products/hawki_cal_flat_set01.fits`. The master flat frame.
- `products/hawki_cal_flat_bpm_set01.fits`. The bad pixel mask using the information from both the dark and flat frames.

7. Reduce the standard star frames.

```
esorex --output-dir=products hawki_cal_zpoint \
   sof/hawki_cal_zpoint.sof
```

The most important file created is:

- `products/hawki_cal_zpoint.fits`. This file contains the photometric solution for each detector. Check column `ZPOINT` in each extension. See **??** for details.

8. Apply the basic calibrations: dark, flat and bad pixel correction to the science jittered images

```
esorex --output-dir=products hawki_step_basic_calib \
   sof/hawki_step_basic_calib.sof
```

One product is created for each input science jitter frame:

- `products/hawki_step_basic_calib_obj_xxx.fits`. This is the calibrated frame corresponding to the xxx raw jittered in the sof (numbers follow the sequential order in the sof)

9. Compute the first pass of background computation.

```
esorex --output-dir=products hawki_step_compute_bkg \
   sof/hawki_step_compute_bkg_1.sof
```

One important product is created for each input science jitter frame:

- `products/hawki_step_compute_bkg_xxx.fits`. This is the background image corresponding to the xxx basic calibrated image in the sof.

10. Subtract background (first pass)

```
esorex --output-dir=products hawki_step_subtract_bkg \
   sof/hawki_step_subtract_bkg_1.sof
```

One product is created for each input science jitter frame:

- `products/hawki_step_subtract_bkg_xxx.fits`. This is the science image with the background subtracted.

11. Combine the jittered stack (first pass).

```
esorex --output-dir=products hawki_step_combine \
   sof/hawki_step_combine_1.sof
```

Only one product is created:

- `products/hawki_step_combine.fits`. This is the combined image of the jitter stack. It has 4 extensions, one per detector (see **??**).

12. Detect objects in the combined image (first pass).

```
esorex --output-dir=products hawki_step_detect_obj \
   sof/hawki_step_detect_obj_1.sof
```

The most important product here is the mask:

- `products/hawki_step_detect_obj_mask.fits`. This frame is a mask that contains values of 1 where an object was present in the combined image.

13. Compute the second pass of background computation. In this case the mask of objects is also taken into account.

```
esorex --output-dir=products hawki_step_compute_bkg \
   sof/hawki_step_compute_bkg_2.sof
```

One important product is created for each input science jitter frame (which will overwrite the previous background images created in the first pass)

- `products/hawki_step_compute_bkg_xxx.fits`. This is the second pass background image corresponding to the xxx basic calibrated image in the sof.

14. Subtract background (second pass).

```
esorex --output-dir=products hawki_step_subtract_bkg \
   sof/hawki_step_subtract_bkg_2.sof
```

One product is created for each input science jitter frame (which will overwrite the previous background subtracted images created in the first pass).

- `products/hawki_step_subtract_bkg_xxx.fits`. This is the second pass science image with the background subtracted.

15. Correct from distortion. This is optional, but it is recommended in case that the combined images show double stars.

```
esorex --output-dir=products hawki_step_apply_dist \
   sof/hawki_step_apply_dist.sof
```

One product is created for each input science jitter frame.

- `products/hawki_step_apply_dist_xxx.fits`. This is the distortion corrected image.

16. Refine the offsets between jittered images. This is optional, but it improves the alignment of all the jittered stack.

```
esorex --output-dir=products hawki_step_refine_offsets \
   sof/hawki_step_refine_offsets_1.sof
```

One product is created that contains the refined offsets.

- `products/hawki_step_refine_offsets.fits`. This is a 4 extension table that contains for each extension a list of offsets, one per input jittered image.

17. Combine the jittered stack (second pass). Combine again the jittered images, now with better background subtracted images, better offsets and distortion corrected images. If the last two steps were not apply, use `sof/hawki_step_combine_1.sof` instead.

```
esorex --output-dir=products hawki_step_combine \
   sof/hawki_step_combine_2.sof
```

Only one product is created:

- `products/hawki_step_combine.fits`. This is the combined image of the jitter stack. It has 4 extensions, one per detector (see **??**).

4.5 Example of jitter data monolithic reduction using *EsoRex*

Jittered science observations can be easily reduced by using the monolithic recipe *hawki_sci_jitter*. Here we will present step-by-step walkthrough of how to reduce data with the monolithic recipe. Note that for a more flexible reduction and better control of the intermediate steps and products the modular reduction can be used instead (4.4). The reduction procedure used in the Paranal observatory is similar to the one presented here.

1. Follow steps 1. to 6. of section 4.4.

2. Reduce the science jitter observations.

```
esorex --output-dir=products hawki_sci_jitter \
   sof/hawki_sci_jitter.sof
```

The most important files created are:

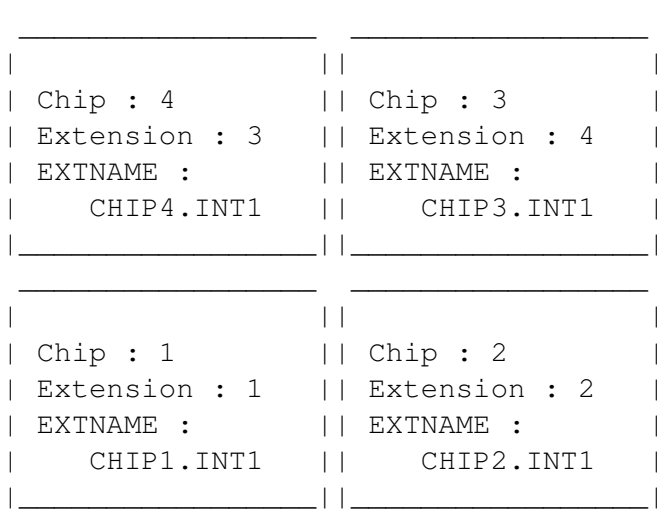
- `products/hawki_sci_jitter.fits`. This is the final combined image using all the jittered images.
- `products/hawki_sci_jitter_stich.fits`. This is also the combined image, but here all the chips have also been combined into the frame. Note that this product might not be suitable for science purposes (see Section **??**).

5 HAWK-I Data Description

This section gives a description of the raw data produced by HAWK-I.

The HAWK-I fits file contains a primary unit without data, and 4 extensions for the 4 chips. The EXTNAME keyword identifies which chip is stored in the current extension (CHIP3.INT1 for chip number 3). Each chip is a 2048x2048 pixels image. The gap between the chips corresponds to approximately 145 pixels.

Beware that the order of extensions in the RAW files do not match the order of the chips: extension 1 contains chip 1, extension 2 contains chip 2, extension 3 contains chip 4 and extension 4 contains chip 3. The same structure is kept in the pipeline product files.



Any raw 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* [11], 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 HAWK-I pipeline recipe, *i.e.*, the recipe assigned to the reduction of that specific frame type. In the pipeline environment this recipe would be launched automatically.

In the following, all HAWK-I raw 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 **??**).

5.1 Calibration frames

- **Dark calibration:**

  DO category: `DARK`
  Processed by: *hawki_cal_dark*

  Classification keywords: Association keywords:

```
DPR CATG = CALIB
DPR TYPE = DARK
DPR TECH = IMAGE
```

- **Flat field calibration:**

  DO category: `FLAT`
  Processed by: *hawki_cal_flat*

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = FLAT
  DPR TECH = IMAGE
  ```
  Association keywords:

- **Zero point:**

  DO category: `ZPOINT`
  Processed by: *hawki_cal_zpoint*

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = STD
  DPR TECH = IMAGE
  ```
  Association keywords:

- **Illumination frame:**

  DO category: `TEC_STD`
  Processed by: *hawki_cal_illum*

  Classification keywords:
  ```
  DPR CATG = TECHNICAL
  DPR TYPE = STD
  DPR TECH = IMAGE
  ```
  Association keywords:

- **Dark image for non-linearity calibration:**

  DO category: `DETLIN_DARK`
  Processed by: *hawki_cal_lingain*

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = IMAGE
  DPR TECH = LINEARITY,DARK,DETCHAR
  ```
  Association keywords:

- **Lamp image for non-linearity calibration:**

  DO category: `DETLIN_LAMP`
  Processed by: *hawki_cal_lingain*

  Classification keywords:
  ```
  DPR CATG = CALIB
  DPR TYPE = IMAGE
  DPR TECH = LINEARITY,FLAT,DETCHAR
  ```
  Association keywords:

5.2 Science frames

- **Science observation of the sky:**
  DO category: `JITTER_SKY`
  Processed by: *hawki_sci_jitter*

  Classification keywords:
  `DPR CATG = SCIENCE`
  `DPR TYPE = SKY`
  `DPR TECH = IMAGE`

  Association keywords:

- **Science observation of an object:**
  DO category: `JITTER_OBS`
  Processed by: *hawki_sci_jitter*

  Classification keywords:
  `DPR CATG = SCIENCE`
  `DPR TYPE = OBJECT`
  `DPR TECH = IMAGE`

  Association keywords:

5.3 Technical frames

- **Filter position verification :**
  DO category: `TEC_FLAT`
  Processed by: *hawki_tec_filtchk*

  Classification keywords:
  `DPR CATG = TECHNICAL`
  `DPR TYPE = FLAT`
  `DPR TECH = IMAGE`

  Association keywords:

6 Static Calibration Data

For HAWK-I, the static calibration data are the standard stars catalogs, and the 2 dimensional map of the distortion.

All these catalogs and maps can be found in the HAWK-I pipeline source distribution as ASCII files in hawkip/catalogs:

```
ls hawkip/catalogs/*

hawkip/catalogs/dist_map_chip1.txt  hawkip/catalogs/dist_map_chip3.txt
hawkip/catalogs/dist_map_chip2.txt  hawkip/catalogs/dist_map_chip4.txt

hawkip/catalogs/stdstars:
HAWKI_Persson_2008-10-30.txt  HAWKI_UKIRT_2008-10-30.txt
```

From these ASCII files, it is possible to generate the FITS calibration tables needed by the recipes by using the following utilities:

- *hawki_util_stdstars* : Standard stars catalog creation
- *hawki_util_gendist* : Distortion maps creation

For example, the following:

```
$ more IN
/home/yjung/hawkip/catalogs/stdstars/HAWKI_Persson_2008-10-30.txt STDSTAR_CAT
/home/yjung/hawkip/catalogs/stdstars/HAWKI_UKIRT_2008-10-30.txt STDSTAR_CAT

$ esorex hawki_util_stdstars IN
```

will create the static calibration FITS file needed by the recipes.

In the same way,

```
$ more IN
/home/yjung/hawkip/catalogs/dist_map_chip1.txt DIST_MAP
/home/yjung/hawkip/catalogs/dist_map_chip2.txt DIST_MAP
/home/yjung/hawkip/catalogs/dist_map_chip3.txt DIST_MAP
/home/yjung/hawkip/catalogs/dist_map_chip4.txt DIST_MAP

$ esorex hawki_util_gendist IN
```

will create the distortion calibration maps.

6.1 Standard Star Catalogs

The standard stars catalogs are used by the *hawki_cal_zpoint* recipe to get the magnitude of the observed standard star to compute the Zero Point.

Stars are currently taken from the following catalogs:

- HAWKI_Persson_2008-10-30.txt (33 entries)
- HAWKI_UKIRT_2008-10-30.txt (75 entries)

Each entry in the catalog correspond to one standard star. They all contain the following information:

- The name of the star
- The position (RA / DEC) of the star
- The spectral type of the star
- The different magnitudes in bands J, H, K, Y

See `http://w4.hq.eso.org/observing/dfo/quality/HAWKI/pipeline/pipe_stdstar_cat.html` for more information.

6.2 Distortion map

The distortion map describes chip by chip the distortion in the image. It contains a grid of x and y correction shifts to apply locally on the image to correct for the distortion.

The pipeline includes the *nominal* distortion map computed at comissioning. However, recent versions of the pipeline include the recipe *hawki_cal_distortion* to compute the distortion solution using calibration data. If you have distortion calibration data we recommend to use it rather than the static distortion map. This data is still no taken regularly as part of the calibration plan, but it will be in the near future.

```
$ more dist_map_chip1.txt
#DXGC        DYGC         I      J
2.84390      2.45460      0      0
2.23870      1.98040    256      0
1.63340      1.60680    512      0
1.06490      1.10210    768      0
0.82980      0.80000   1024      0
0.55210      0.42790   1280      0
0.37440      0.17950   1536      0
0.16430     -0.01240   1792      0
0.14150     -0.26860   2048      0
2.48740      2.11620      0    256
```

```
1.96600     1.66470   256   256
1.37510     1.22280   512   256
...
```

The distortion calibration FITS files (created with hawki_util_gendist) are used by the *hawki_sci_jitter* recipe and the *hawki_step_apply_dist* utility.

7 Data Reduction

In this section, after an overview of the main problems the data reduction needs to solve, we list the required data and the recipes which allow to solve them, giving the data reduction sequence necessary to reduce calibration and science data.

The data reduction is strongly connected with the observation strategy which in turn depends on the nature of the observed object and the science goals of the observations. The Pipeline only treats the most general cases and the user is strongly recommended to read the HAWK-I user manual for detailed discussion of the observing strategies and their optimizations.

7.1 Data reduction overview

The most used observation technique is the jittering. Small shifts are applied between successive frames. This way, with a set of a sufficient number of frames, it is possible to make a precise estimation of the sky for all the pixels of the detector, the sky estimation being the most important and difficult part usually in IR.

7.2 Monolithic and modular reduction

A new feature introduced since release 1.4.0 is the concept of modular reduction. The modular reduction mode has been implemented for the jitter technique and allows more control in the steps taken during the reduction as well as intermediate products inspection.

The modular recipes were created to provide more fine-grained control of the reduction process, interactivity and intermediate product saving. It is worth noting that in particular sky subtraction is much improved with respect to the *hawki_sci_jitter* recipe due to a *double pass* scheme. This is the standard IR reduction approach similar to other well proven packages (see for example the xdimsum package [13]).

The modular recipes have been designed with a standard workflow in mind in order to create coherent products. In the common case of several jitter images and no sky images the workflow should be run as follows:

1. *hawki_step_stats*: create raw statistics (optional).
2. *hawki_step_basic_calib*: basic calibration of raw science images.
3. *hawki_step_compute_bkg*: first background computation
4. *hawki_step_subtract_bkg*: background subtraction
5. *hawki_step_combine*: combination of background subtracted images. No distortion or offsets refinement is applied.
6. *hawki_step_detect_obj*: detect objects in the combined image and get the object mask.
7. *hawki_step_compute_bkg*: second background computation, using the object mask computed above.
8. *hawki_step_subtract_bkg*: background subtraction

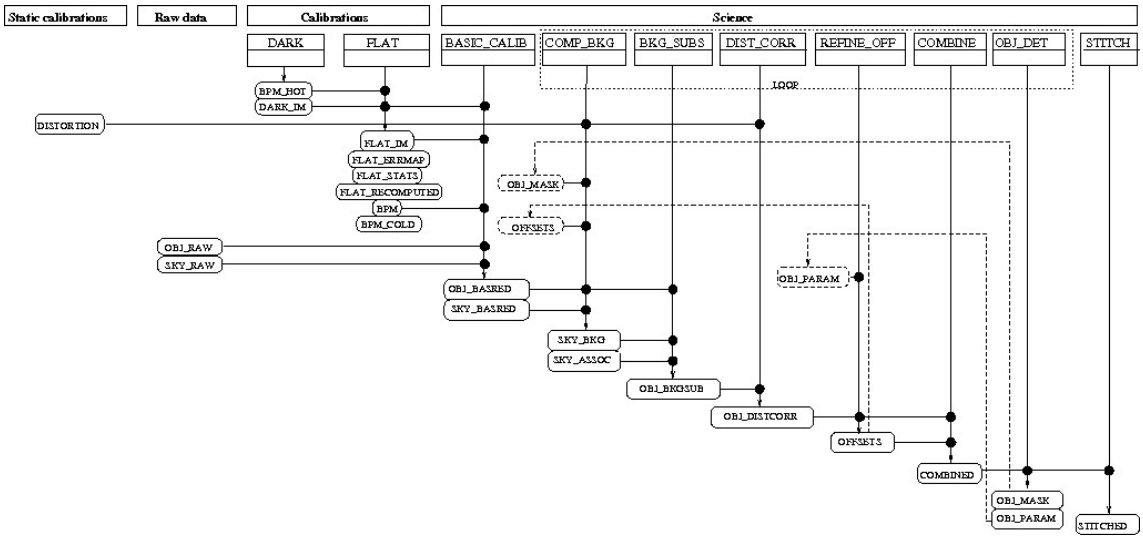


Figure 7.2.1: *HAWK-I Modular recipes Association Map.*

9. *hawki_step_apply_dist*: correct all the images from distortion
10. *hawki_step_refine_offsets*: use the detected objects to correlate the images and refine the offsets.
11. *hawki_step_combine*: combine the raw images using the refined offsets and distortion-corrected images.
12. *hawki_step_detect_obj*: detect the objects for accurate photometry (optional).
13. *hawki_step_stitch*: stitch all HAWK-I detectors together (optional).

Figure 7.2.1 shows the interdependency between all the modular recipes and allows to understand the input and products for each recipe.

Different recipe workflows of the recipes may be constructed, however there is limited support for running the modular recipes in non-standard combinations. See the individual recipe descriptions for details.

7.3 Required input data

To be able to reduce science data one needs to use raw, product data and pipeline recipes in a given sequence which provides all the input necessary to each pipeline recipe. We call this sequence a data reduction cascade.

Calibration data products can be generated from raw data using the pipeline recipes. Alternatively the user may use calibration products obtained from the ESO archive or from the ESO Data Flow Operation department.

7.4 Reduction Cascade

The reduction cascade is described on figure 7.4.1.

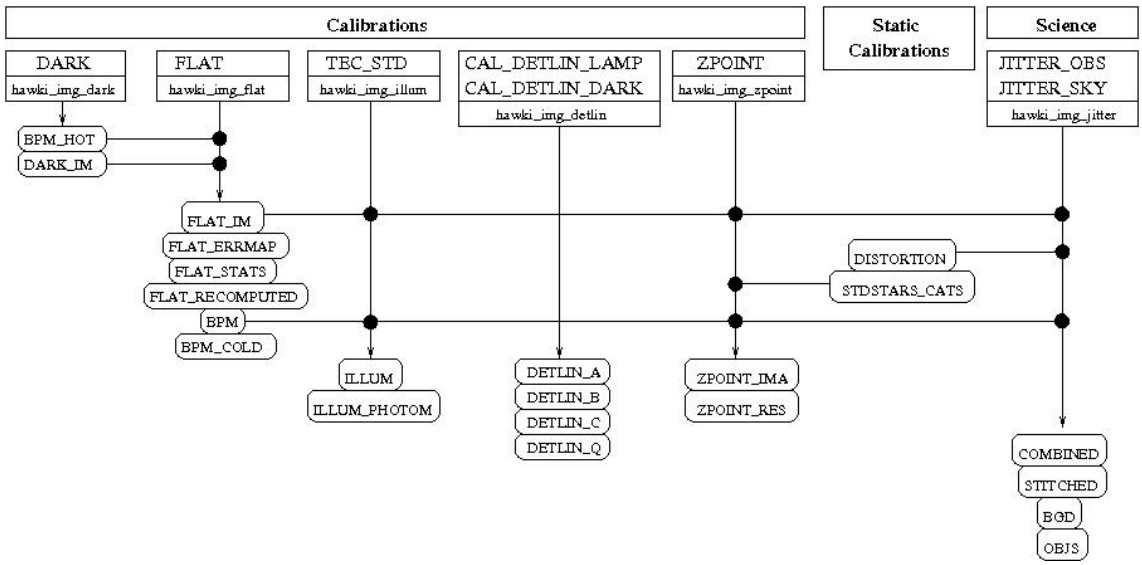


Figure 7.4.1: *HAWK-I Association Map.*

8 Pipeline Recipe Interfaces

In this section we provide for each recipe examples of the required input data.

We also provide a list of the pipeline products for each recipe, indicating their default recipe name, the value of the FITS keyword HIERARCH ESO PRO CATG (in short PRO.CATG) and a short description. Note that depending on the *EsoRex* or *Gasgano* options the final names of the products may differ.

For each recipe we also list the input parameters (as they appear in the recipe configuration file), the corresponding aliases for the command line usage, and their default values. Also quality control parameters are listed. Those are stored in relevant pipeline products. More information on instrument quality control can be found on http://www.eso.org/qc

All the products of the recipes are multiextension FITS images or tables except in the cases indicated below. Each extension contains data regarding on of the detectors, following the same numbering scheme as the raw data (see section 5).

In addition to the products mentioned below, all recipes produce some PAF (VLT parameter file) which is an intermediate pipeline ASCII data file containing quality control parameter values.

8.1 hawki_cal_dark

This recipe creates a master dark image, and computes the Read-Out Noise of the detectors along with several statistics.

8.1.1 Input

This recipe expects at least 3 input frames classified as DARK.

8.1.2 Products

- A master dark image named `hawki_cal_dark.fits` (PRO CATG = DARK_IM) and a hot pixels map named `hawki_cal_dark_hotpix.fits` (PRO CATG = BPM_HOT) are created.
- A statistics table named `hawki_cal_dark_stats.fits` (PRO CATG = DARK_STATS). For each raw input frame the following statistics values are tabulated: MIN, MAX, MEAN, MED, STDEV along with a field that states whether that raw image has been used for the final master dark or not (USED).
- If error propagation is requested, a product with the error map of the master dark is created (PRO CATG = DARK_ERR).

8.1.3 Quality control

The quality control parameters are computed for each detector. Within each detector, the successive pairs are used to compute RON values.

- QC DARK MEAN: Mean of the median values of the different input images
- QC DARK MED: Median of the median values of the different input images
- QC DARK STDEV: Standard deviation of the median values of the different input images
- QC RONi: Read Out Noise value computed on pair number i
- QC RON MEAN: The mean Read Out noise
- QC RON MED: The median Read Out noise
- QC RON STDEV: The standard deviation of Read Out noises
- QC DARK NBADPIX: Number of hot pixels computed
- QC DARK VCi MEAN: Mean value computed in the Video Channel number i in the computed master dark image
- QC DARK VCi MED: Median value computed in the Video Channel number i in the computed master dark image
- QC DARK VCi STDEV: Standard deviation value computed in the Video Channel number i in the computed master dark image
- QC DATANCOM: Number of frames used for the reduction
- QC RAW [MAXIMUM, MINIMUM, MEAN, MEDIAN, RMS] [MEAN, MEDIAN, MAXIMUM, MINIMUM, STDEV]: Statistics (2nd part of the keyword) of the raw frame statistics (firs part of the keyword). For instance QC RAW MEAN STDEV is the standard deviation of all the mean values of the raw frames.

8.1.4 Parameters

- *--sigma*: Sigma value used for hot pixels detection (default is 10.0)
- *--nsamples*: Number of samples used to compute the RON (default is 100)
- *--hsize*: Half-size of the boxes used to compute the RON (default is 6)
- *--zone*: Region definition for the medians computation (default is 512,512,1536,1536)
- *--gain*: Detector nominal gain in e-/ADU (default is -1.0)
- *--ron*: Detector nominal RON for a single readout in ADU (default is -1.0)

8.2 hawki_cal_flat

This recipe creates a master flat field from twilight observations, a bad pixels map, and various statistics.

8.2.1 Input

This recipe expects input frames classified as FLAT, optionally a dark frame classified as DARK_IM and/or a hot pixels map classified as BPM_HOT.

The frames are first classified by the recipe by different filters and different DITs, each group is then reduced separately.

8.2.2 Products

For each setting, the following products are created (xx identifies the group or setting number):

- A master flat image named `hawki_cal_flat_setxx.fits` (PRO CATG = FLAT_IM).
- The map identifying the recomputed pixels in the second pass named `hawki_cal_flat_recomputed_setxx.` (PRO CATG = FLAT_RECOMPUTED).
- An image with the error bar of the fit for each pixel named `hawki_cal_flat_err_setxx.fits` (PRO CATG = FLAT_ERRMAP).
- A cold pixels map named `hawki_cal_flat_coldpix_setxx.fits` (PRO CATG = BPM_COLD). Despite its name (for historical reasons), it does not only contain cold pixels, but it includes all the pixels regarded as bad by the recipe.
- Optionally, a bad pixels map created from the cold pixels map and a passed hot pixels map named `hawki_cal_flat_bpm_setxx.fits` (PRO CATG = BPM).
- A statistics table named `hawki_cal_flat_stats_setxx.fits` (PRO CATG = FLAT_STATS). For each raw input frame the following statistics values are tabulated: MIN, MAX, MEAN, MED, STDEV along with a field that states whether that raw image has been used for the final master flat or not (USED).
- If the flag extra_stats is set, a statistics table named `hawki_cal_flat_stats_ec_setxx.fits` (PRO CATG = FLAT_STATS_EVEN_COL) which contains statistics for each even column.
- If the flag extra_stats is set, a statistics table named `hawki_cal_flat_stats_oc_setxx.fits` (PRO CATG = FLAT_STATS_ODD_COL) which contains statistics for each odd column.
- If the flag extra_stats is set, a statistics table named `hawki_cal_flat_stats_er_setxx.fits` (PRO CATG = FLAT_STATS_EVEN_ROW) which contains statistics for each even row.
- If the flag - *-extra_stats* is set, a statistics table named `hawki_cal_flat_stats_or_setxx.fits` (PRO CATG = FLAT_STATS_ODD_ROW) which contains statistics for each odd row.

8.2.3 Quality control

- QC FLAT MEDMIN: The minimum value of the medians of the input flat images
- QC FLAT MEDMAX: The maximum value of the medians of the input flat images

- QC FLAT MEDMEAN: The average value of the medians of the input flat images
- QC FLAT MEDSTDEV: The standard deviation of the medians of the input flat images
- QC FLAT NBADPIX: The number of bad pixels detected
- QC FLAT NORM: The normalisation factor applied
- QC.DATANCOM: Number of frames used for the reduction
- QC RAW [MAXIMUM, MINIMUM, MEAN, MEDIAN, RMS] [MEAN, MEDIAN, MAXIMUM, MINIMUM, STDEV]: Statistics (2nd part of the keyword) of the raw frame statistics (firs part of the keyword). For example QC RAW MEAN STDEV is the standard deviation of all the mean values of the raw frames.

8.2.4 Parameters

- *--zone*: Zone where the statistics are computed (default is "1,1,2048,2048")
- *--normalise*: Flag to apply the final normalisation (default is FALSE)
- *--second_pass*: Flag to apply the second pass (default is TRUE)
- *--sigma_badres*: Sigma to identify the pixels that need a second pass (default is 1.0)
- *--sigma_bpm*: Sigma for the bad pixels detection (default is 10.0)
- *--lowval_bpm*: Pixels in the flat below this value will be included in the bpm. It is in units of final flat (normalized if normalise is on (default is .1)
- *--highval_bpm*: Pixels in the flat above this value will be included in the bpm. It is in units of final flat (normalized if normalise is on (default is 10.)
- *--select_auto*: Flag to automatically select the frames used for the fit (default is TRUE)
- *--select_auto_max_bins*: Number of maximum bins to use if *--select_auto* is set. This in turn will be the maximum number of flats used (default is 10). WARNING: The default value has been selected based on the Paranal Observatory experience. If you have more flats at your disposal, you can increase this number to achieve better S/N (probably you will have to play with the other *--select-xxx* parameters).
- *--select_min_level*: Minimum mean value of the frames used for the fit (default is -1.0)
- *--select_max_level*: Maximum mean value of the frames used for the fit (default is 25000)
- *--select_max_rms*: Maximum RMS value of the frames used for the fit (default is 4000)
- *--select_min_nframes*: Minimum number of frames (default is 3)
- *--extra_stats*: Request for even/odd column/rows statistics (default is FALSE)

8.3 hawki_cal_illum

This recipe computes the response of the detectors from their illumination using a standard star moved across those detectors.

NOTE: This recipe will be soon deprecated and there is no longer support. A new recipe to compute accurate illumination correction will be delivered in coming releases

8.3.1 Input

This recipe expects input frames classified as TEC_STD for the illumination frames. It also accepts bad pixels map (BPM) and a flat field (FLAT_IM). Each illumination frame is a standard star at a given position on one of the detectors.

8.3.2 Products

The two products are the image that gives the polynomial illumination on the whole detector named `hawki_cal_illum.` (PRO CATG = ILLUM) and a table named `hawki_cal_illum_photom.fits` containing the flux and the position of the standard stars in the input frames (PRO CATG = ILLUM_PHOTOM) as shown in Figure 8.3.1.

```
#
# file           hawki_img_illum_photom.fits
# extensions     4
# ---------------------------------------------
# XTENSION       2
# Number of columns 3
#
      POSX|      POSY|      FLUX
       435|      1702|1.15724e+06
       435|      1306|1.12054e+06
       434|       910|1.17831e+06
       434|       518|1.25785e+06
       433|       123|1.24589e+06
       830|      1699|1.03413e+06
       828|      1305|1.20793e+06
       827|       911|1.22662e+06
       827|       518|1.20964e+06
       828|       121|1.14135e+06
      1223|      1699|1.17492e+06
      1222|      1304|1.20316e+06
      1222|       911|1.24219e+06
      1222|       515|1.24538e+06
      1222|       121|1.26832e+06
      1616|      1698|1.06467e+06
      1617|      1304|1.23185e+06
      1617|       910|1.26573e+06
      1616|       516|1.22662e+06
      1617|       120|1.2361e+06
      2012|      1698|1.20083e+06
      2012|      1303|1.19246e+06
      2011|       909|1.15647e+06
      2012|       514|1.21567e+06
      2013|       118|1.24561e+06
```

Figure 8.3.1: *Table created by the illumination recipe for detector 2.*

8.3.3 Quality control

- QC.DATANCOM: Number of frames used for the reduction

8.3.4 Parameters

- *--subtract*: Flag to subtract the next image to clean the background (default is TRUE).
- *--degree*: The polynomial degree for the fit (default is 3).
- *--star_r*: The star radius in pixels (default is -1.0)
- *--bg_r1*: The internal radius in pixels of the ring used for the background computation (default is -1.0)
- *--bg_r2*: The external radius in pixels of the ring used for the background computation (default is -1.0)
- *--s_hx*: X half size in pixels of the box used for the star detection (default is 50).
- *--s_hy*: Y half size in pixels of the box used for the star detection (default is 50).

8.4 hawki_cal_zpoint

This recipe computes the Zero Point values of the detectors.

8.4.1 Input

The recipe expects a set of 4 frames with a standard star exposure for each detector. The standard star must appear around the center. Those frames are tagged with ZPOINT. The recipe also expects the FITS calibration file (STDSTARS_CATS) containing the list of known standard stars with their positions (J2000) and magnitudes in the different bands.

The recipe also accepts a bad pixels map(tagged with BPM) or a flat field (tagged with FLAT_IM).

8.4.2 Products

Three files are produced.

- The first is a table named `hawki_cal_zpoint.fits` containing for each detector the standard star photometry computed as shown in Figure 8.4.1 (PRO CATG = ZPOINT_RES).

```
#
# file          hawki_img_zpoint.fits
# extensions    1
# ----------------------------------------------
# XTENSION      1
# Number of columns 14
#
   CHIP|      POSX|      POSY|    ZPOINT|      ATXO|      FLUX|      PEAK|       BGD|     FWHMX|     FWHMY|      FWHM| FWHMX_AS| FWHMY_AS|  FWHM_AS
      4|  1078.16|  1044.02|  25.4844|  25.5597|  203787|  3097.23|  0.866577|  6.67497|  6.43001|  6.55134|  0.710417|  0.684346|  0.69726
      3|  1053.66|  1039.29|  25.7454|  25.8208|  259180|  3788.67|  -11.4667|  6.77799|  6.26358|  6.51571|  0.721382|  0.666633|  0.693467
      2|  1059.19|  1049.49|  25.5047|  25.5801|  207647|  2215.07|  -2.69995|  7.62674|  6.96529|  7.28851|  0.811714|  0.741315|  0.775716
      1|  1083.71|  1055.26|  25.7122|  25.7876|  251375|  2387.03|  18.7666|  8.06087|  7.11262|  7.57191|  0.857918|  0.756996|  0.805879
```

Figure 8.4.1: *Example of the output produced by hawki_cal_zpoint.*

- The second one is an image named `hawki_cal_zpoint_check.fits` with the extracted stars used to verify that the proper stars have been used for the computation (PRO CATG = ZPOINT_IMA).

- A statistics table named `hawki_cal_zpoint_stats.fits` (PRO CATG = ZPOINT_STATS). For each raw input frame the following statistics values are tabulated: MIN, MAX, MEAN, MED, STDEV.

8.4.3 Quality control

- QC FILTER OBS: The name of the filter used in the header
- QC FILTER REF: The name of the filter actually used to get the information from the catalog
- QC AMBI RHUM AVG: The humidity average
- QC AIRMASS: The airmass
- QC STDNAME: The name of the standard star
- QC SPECTYPE: The spectral type of the standard star
- QC STARMAG: The magnitude of the standard star
- QC CATNAME: The catalog name where the star has been found
- QC.DATANCOM: Number of frames used for the reduction
- QC ZPOINT MEAN: The mean computed zero point for all the chips
- QC ATX0 MEAN: The computed mean zero point with the extinction correction for all the chips.
- QC ZPOINT: The computed zero point (one per each chip)
- QC ATX0: The computed zero point with the extinction correction (one per each chip)
- QC ZPOINT POSX: X position of the standard star
- QC ZPOINT POSY: Y position of the standard star
- QC ZPOINT FLUX: Flux of the standard star
- QC ZPOINT PEAK: Peak value of the standard star
- QC ZPOINT BGD: Background around the standard star
- QC ZPOINT FWHMX: FWHM in X in pixels of the standard star
- QC ZPOINT FWHMY: FWHM in Y in pixels of the standard star
- QC ZPOINT FWHM: FWHM in pixels of the standard star
- QC ZPOINT FWHMX_AS: FWHM in X in arcsecs of the standard star
- QC ZPOINT FWHMY_AS: FWHM in Y in arcsecs of the standard star
- QC ZPOINT FWHM_AS: FWHM in arcsecs of the standard star
- QC RAW [MAXIMUM, MINIMUM, MEAN, MEDIAN, RMS] [MEAN, MEDIAN, MAXIMUM, MINIMUM, STDEV]: Statistics (2nd part of the keyword) of the raw frame statistics (firs part of the keyword). For example QC RAW MEAN STDEV is the standard deviation of all the mean values of the raw frames

8.4.4 Parameters

- *--detect_sigma*: Sigma value for object detection (default is 7.0).
- *--star_r*: Star radius in pixels (default is -1.0)
- *--bg_r1*: Background ring internal radius in pixels (default is -1.0)
- *--bg_r2*: Background ring external radius in pixels (default is -1.0)
- *--ra*: RA position in J2000 (default is 999.0 for unknown)
- *--dec*: DEC position in J2000 (default is 999.0 for unknown)
- *--mag*: star magnitude (default is 99.0 for unknown)
- *--sx*: Half-size in the X direction of the search box (default is 100)
- *--sy*: Half-size in the Y direction of the search box (default is 100)
- *--xcoord*: Coordinates in X axe where the standard star is located in each detector, using the format $[x_{det1}, x_{det2}, x_{det3}, x_{det4}]$. If a value of -1 is found, then the position of the star is computed from the WCS. Default is [-1., -1., -1., -1.].
- *--ycoord*: Coordinates in Y axe where the standard star is located in each detector, using the format $[y_{det1}, y_{det2}, y_{det3}, y_{det4}]$. If a value of -1 is found, then the position of the star is computed from the WCS. Default is [-1., -1., -1., -1.].

8.5 hawki_cal_distortion

This recipe computes the distortion solution using the autocalibration method described in A. This method does not require an external reference catalog to obtain the star positions.

8.5.1 Input

The recipe expects frames marked with DISTOR_OBS for distortion field frames and DISTOR_SKY for sky frames.

8.5.2 Products

The recipe produces two files:

- `hawki_cal_distortion_x.fits` (PRO CATG = DISTORTION_X). Contains for each pixel of the detector which is the distortion correction in the X axis (Δ x(x,y)).
- `hawki_cal_distortion_y.fits` (PRO CATG = DISTORTION_Y). Contains for each pixel of the detector which is the distortion correction in the Y axis (Δ y(x,y)).

8.5.3 Quality control

- QC DIST NMATCHED: Number of object pairs used to compute the solution.
- QC DIST TOTAL RMS: The sum of all the rms in object positions.

8.5.4 Parameters

- *--sigma_det*: Detection level above the sigma of the sky to detect objects in the individual images (default: 6.0).
- *--grid_points*: Number of points in distortion grid (default: 9).
- *--borders*: Number of pixels to trim at the borders (default: 6).
- *--subtract_linear*: Subtract a linear term to the solution (default: TRUE).

8.6 hawki_sci_jitter

This recipe is the science recipe in imaging. It reduces the data to create clean combined images and an additional stitched image.

8.6.1 Input

The recipe expects frames marked with JITTER_OBS for object frames and optionally JITTER_SKY for sky frames

It also accepts a bad pixels map (BPM), a flat field (FLAT_IM) and/or distortion calibration files (DISTORTION_X, DISTORTION_Y).

8.6.2 Products

- The produced combined images are stored in a 4 extensions file named `hawki_sci_jitter.fits` (PRO CATG = COMBINED).
- The 4 combined images are stitched together and saved in a file named `hawki_sci_jitter_stitched.fits` (PRO CATG = STITCHED) which contains a single extension. Note that this stitched image is not meant for scientific exploitation.
- Also, a 4 tables file named `hawki_sci_jitter_bkg_stats.fits` with the background values (PRO CATG = BKG_STATS).
- Additionally, a 4 tables file named `hawki_sci_jitter_stars.fits` (PRO CATG = OBJ_PARAM) with the detected stars and their photometry are produced (see Figure 8.6.1). This table contains the following fields:

- POS_X: The X coordinate of the position of the star in the image.
- POS_Y: The Y coordinate of the position of the star in the image.
- FWHM_MAJAX: The major axis of the fitted ellipse.
- FWHM_MINAX: The major axis of the fitted ellipse.
- ELLIP: 1 - MINAX / MAJAX.
- ANGLE: The angle which forms the major axis in the image.

```
# ------------------------------------------
# XTENSION       2
# Number of columns 6
#
     POS_X|     POS_Y|     ANGLE|    FWHM_X|    FWHM_Y|       FLUX
   1602.19|   28.4864|   15.8558|   6.99954|   6.49589|     174440
   928.153|   140.334|   33.8561|   6.82794|   6.03887|    69988.4
   1593.77|    188.93|   31.3773|   6.80129|   6.10976|    81180.4
   1637.75|   228.775|   30.2647|   6.82564|   6.05041|    85860.8
    539.23|   351.752|   120.359|      13.2|   14.0402|7.02129e+06
   790.854|   367.584|   34.0468|   6.81881|   5.82144|    32805.8
   1296.24|   370.477|   30.3191|   6.82401|   5.96493|    76841.1
   5.06546|    443.61|         0|        -1|        -1|     185451
   1923.02|   454.172|   35.7089|   8.13245|   6.92263|3.40373e+06
   1196.11|   534.419|   32.6417|   6.84994|   5.83174|    31813.2
   401.031|   644.007|   32.7036|    6.8232|   5.83532|    14842.2
   306.534|   695.278|   35.3008|   6.84022|   5.82132|     170218
   1796.18|    743.55|   30.8872|   7.12347|   5.90291|     166201
   2009.14|      929|   32.8318|   7.05978|    5.9244|     145811
   1176.65|   960.077|   24.8725|   6.88831|   5.78043|    53869.7
   1627.99|   1085.97|   30.7632|   7.17036|   5.88044|      83114
   540.974|    1100.5|   24.8918|   6.76779|   5.74589|    19856.5
   881.857|   1313.45|   25.5285|   6.90481|   5.76319|     328225
   52.5517|   1432.39|   33.5231|   6.72978|   5.75664|     591868
   2023.16|   1434.39|   28.1107|   7.09338|   5.98466|    68870.1
   1937.42|   1491.36|   118.134|   6.01742|   7.30635|     138957
   1431.43|   1522.42|   29.5498|   6.91271|   5.88518|     113428
   69.9942|   1535.29|   23.8548|   6.68564|   5.60075|     317531
   29.8535|   1539.08|    30.545|   6.61058|   5.68275|     102236
   1274.81|   1650.18|   26.8458|   7.00156|   5.84987|     165916
   371.425|   1727.02|   17.2297|   6.93561|   6.04637|     126739
   1685.04|    1738.5|   26.8847|   7.17469|   5.93549|    35001.8
   1397.76|   1770.65|   148.952|   12.6418|   15.6566|3.62012e+06
   1972.06|   1784.57|   26.8097|   7.06895|   5.96048|     126874
   72.9671|   1812.97|   33.7944|   6.58278|   5.67468|    13909.7
   75.5757|   1885.12|   31.7022|   6.63513|   5.65242|     333436
   110.397|   1888.41|   30.7502|   6.66608|   5.62139|     135668
   60.5917|   1892.41|   38.7337|   6.71596|   5.75184|    24426.6
   912.029|   1966.01|   26.6893|   6.84692|   5.81161|    14032.8
   66.0808|   1990.96|   33.2794|   7.24035|   6.24312|1.71898e+06
    142.23|   2000.97|   29.3097|   6.89809|   5.91925|     695851
```

Figure 8.6.1: *Photometry for detector 2 from hawki_sci_jitter_stars.fits.*

- A statistics table named `hawki_sci_jitter_stats.fits` (PRO CATG = JITTER_STATS). For each raw input frame the following statistics values are tabulated: MIN, MAX, MEAN, MED, STDEV.
- Finally, a product which contains the telescope pointing and atmospheric conditions at each of the raw images as well as statistics on these quantities is stored in file `hawki_sci_jitter_pcs.fits` (PRO CATG = SCIENCE_PCS).

8.6.3 Quality control

- QC BACKGD MEAN: The mean of the background values

- QC BACKGD STDEV: The standard deviation of the background values
- QC BACKGD INSTMAG: The background magnitude
- QC NBOBJS: The number of detected objects
- QC IQ: The Image Quality
- QC IQ DIFF AMBI: The difference between the Image Quality and the observatory seeing queried by the AS
- QC IQ DIFF TEL: The difference between the Image Quality and the delivered seeing corrected by airmass
- QC FWHM PIX: The average FWHM in pixels
- QC FWHM ARCSEC: The average FWHM in arc seconds
- QC FWHM MODE: The FWHM mode
- QC COMBINED POSX: X Position of the first raw image in the combined image
- QC COMBINED POSY: Y Position of the first raw image in the combined image
- QC COMBINED CUMOFFSETX: X Position of the combined image in the reference system used by the input offsets.
- QC COMBINED CUMOFFSETY: Y Position of the combined image in the reference system used by the input offsets.
- QC.DATANCOM: Number of frames used for the reduction
- QC RAW [MAXIMUM, MINIMUM, MEAN, MEDIAN, RMS] [MEAN, MEDIAN, MAXIMUM, MINIMUM, STDEV]: Statistics (2nd part of the keyword) of the raw frame statistics (firs part of the keyword). For instance QC RAW MEAN STDEV is the standard deviation of all the mean values of the raw frames
- QC TEL [ALT, AZ, AMBI RHUM, AMBI TAU0, AMBI WINDDIR, AMBI WINDSP, AMBI FWHM, AMBI PRESS, IA FWHM, AIRM, PARANG, PARANG START, PARANG END, PARANG DELTA] [MEAN, MED, MIN, MAX, STDEV]. Statistics (2nd part of the keyword) of several telescope data (firs part of the keyword).
- QC ADA ABSROT [START, END, DELTA] [MEAN, MEDIAN, MAX, MIN, STDEV]: Statistics for Nasmyth rotator position at the beginning, end and difference of both along the observation series
- QC OBJ [ANGLE ELLIP] [MEAN, MEDIAN, MAX, MIN, STDEV]: Statistics for the angle and ellipticity of the objects detected in the combined image.

All the parameters related with image quality are obtained using the photometric algorithm described in section 9.4.

8.6.4 Parameters

- *--offsets*: The offsets text file. If specified this offsets are used instead of the header ones. It is an ASCII file with two columns: x and y offsets and one row for each frame in the same order as in the sof (default is NULL)
- *--refine*: Flag to controlling computation of offsets refinement (default is `FALSE`).
- *--objects*: The correlation object(s) position(s). If the refine parameter is set the correlation is done around this objects. It is an ASCII file with two columns: x and y positions of the center of each object and one row per object (default is NULL)
- *--offset_max*: The maximum offset allowed (default is 1500)
- *--sky_par*: The sky filtering parameters (minimum number of frames in the whole stack to apply running mean, half size of the window for the running filter, number of low rejections, number of high rejections) (default is "10,7,3,3")
- *--xcorr*: Cross correlation search and measure sizes in x and y (default is "20,20,25,25")
- *--comb_meth*: Combination method for the stacking (union/inter/first) (default is union)
- *--rej*: High and Low rejections for the stacking (default is 0,0)
- *--borders*: Number of pixels rejected on the image borders (default is 2)
- *--max_njitter*: Maximum numbers of jitter frames to process. If set, the first max_obj frames of the sof are used and the rest are disregarded. However the statistics are computed for the whole set. This is only useful if there are memory problems, see section D.4 (default is -1)

8.7 hawki_cal_lingain

The recipe computes the coefficients of a 2nd degree polynomial to correct for the non-linearity of the detectors in a standard way as defined within the DETMON project.

8.7.1 Input

This recipe expects input frames classified as DETLIN_LAMP or DETLIN_DARK. There must be the same number of frames with both tags.

8.7.2 Products

The four products are each 4 extensions fits files. The product named hawki_cal_lingain_bpm.fits is a bad pixels map (PRO CATG = LINGAIN_BPM). hawki_cal_lingain_coeffs_cube.fits is a cube of the coefficients describing the linearity relation (PRO CATG = LINGAIN_COEFFS).

hawki_cal_lingain_gain_table.fits (PRO CATG = LINGAIN_GAIN) and hawki_cal_lingain_linearity_table.fits (PRO CATG = LINGAIN_LIN) are tables containing various statistics.

8.7.3 Quality control

The QC parameters are computed for each chip. They are stored in the 4 different extensions of the product.

Lamp flux QC LAMP FLUX in ADU/s.

Detector gain The median of the computed gain values and relative stdev QC GAIN MED, QC GAIN STD, in e^-/ADU and its inverse QC CONAD in ADU/e^-, and the autocorrelation factor QC AUTCORFA.

The median calculated via polynomial fit is also stored as QC GAIN FIT.

Readnoise QC READNOISE in e^-.

And from the linearity part of the recipe:

Lamp flux RMS The recipe monitors the RMS of the lamp flux (QC LAMP STD).

Non linear coefficients Are monitored the polynomial fit coefficients (QC LIN COEFi, i=0-2) and the corresponding errors (QC LIN COEFi ERR).

Effective non linearity An additional QC parameter is QC LIN EFF, the effective non-linearity correction at a reference user defined level **ref_level**: $Q^{'}$(**ref_level**).

Number of non polynomial pixels The recipe monitors the number of pixels deviating from the polynomial fit (QC NUM BPM).

8.7.4 Parameters

- method: Method to be used when computing GAIN. Methods applicable: <PTC | MED> (default: PIC).
- order: Polynomial order for the fit (Linearity) (default: 2).
- kappa: Kappa value for the kappa-sigma clipping (Gain) (default: 3).
- niter: Number of iterations to compute rms (Gain) (default: 25).
- llx: x coordinate of the lower-left point of the region of interest.
- lly: y coordinate of the lower-left point of the region of interest.
- urx: x coordinate of the upper-right point of the region of interest.
- ury: y coordinate of the upper-right point of the region of interest.

- ref_level: User reference level (default 10).
- intermediate: De-/Activate intermediate products (default: FALSE)
- autocorr: De-/Activate the autocorr option (default: FALSE)
- collapse: De-/Activate the collapse option (default: FALSE)
- rescale: De-/Activate the rescale option (default: TRUE)
- pix2pix: De-/Activate the pix2pix option (default: TRUE)
- bpmbin: De-/Activate the binary bpm option (default: FALSE)
- m: Maximum x-shift for the autocorr (default: 26)
- filter: Upper limit of Median flux to be filtered (default: -1)
- n: Maximum y-shift for the autocorr (default: 26)
- tolerance: Tolerance for pair discrimination (default: 0.001)
- pafname: Specific name for PAF file (default: hawki_cal_lingain)
- exts: Activate the multi-exts option (default: -1)
- fpn_method: Method for computing Fixed Pattern Noise: SMOOTH or HISTOGRAM (default: HISTOGRAM).
- fpn_smooth: Template size in pixels for smoothing during FPN computation (only for SMOOTH method) (default: 13).
- saturation_limit: All frames with mean saturation above the limit would not be used in calculation (default: 65535.0).

8.8 hawki_tec_filtchk

This recipe takes flats taken with different filters and computes statistics in order to check that the filters are properly placed.

8.8.1 Input

The recipe expects frames marked with TEC_FLAT.

8.8.2 Products

A single product named `hawki_tec_filtchk.fits` (PRO CATG = FILTERPOS_CHECK_STATS) which contains the MINIMUM, MAXIMUM, MEAN, MEDIAN and RMS for every raw frame.

8.8.3 Quality control

- QC RAW [MAXIMUM, MINIMUM, MEAN, MEDIAN, RMS] [MEAN, MEDIAN, MAXIMUM, MINIMUM, STDEV]: Statistics (2nd part of the keyword) of the raw frame statistics (first part of the keyword). For example QC RAW MEAN STDEV is the standard deviation of all the mean values of the raw frames

8.8.4 Parameters

This recipe accepts no parameters.

8.9 hawki_step_basic_calib

This recipe apply the basic calibrations: flat, dark and bpm to science images.

8.9.1 Input

The recipe expects frames marked with JITTER_OBS and optionally JITTER_SKY which correspond to objects or sky frames respectively.

It also accepts a bad pixels map (BPM), a master dark (DARK_IM) and a flat field (FLAT_IM). This inputs are optional, only the appropriated calibrations will be applied.

8.9.2 Products

For each input frame the recipes produces a frame calibrated with the dark, flat and bpm with the name `hawki_step_basic_calib_sky,obj_xxx.fits`, where the sky (PRO CATG = SKY_BASIC_CALIBRATED) and obj (PRO CATG = BASIC_CALIBRATED) postfixes are used for sky and object images respectively and xxx specifies the filter set.

8.9.3 Quality control

This recipe does not compute any quality control.

8.9.4 Parameters

This recipe accepts no parameters.

8.10 hawki_step_compute_bkg

Background removal is crucial in HAWK-I. Depending on the observational template used there may be sky images taken along the science images or not. In the first case the background is computed with the sky images,

while in the second case a running median using the nearby frames. In this case an optional object masking can be applied.

8.10.1 Input

The recipe expects frames marked with BASIC_CALIBRATED and optionally SKY_BASIC_CALIBRATED which correspond to objects and sky frames respectively.

Optionally an object mask OBJ_MASK can be applied to reject pixels that are too bright. The relative offsets of the input images can be specified with a frame of type OFFSETS_REFINED (if not set, the header nominal offsets are used). The offsets of the coordinates in the object mask are retrieved from keywords ESO QC COMBINED CUMOFFSET[X, Y] of the object mask frame. Optionally, if the object mask was computed in a distortion corrected image, distortion frames (DISTORTION_X, DISTORTION_Y) can be an input.

8.10.2 Products

A single output frame `hawki_step_compute_bkg_01.fits` is produced (PRO CATG = BKG_IM) in the following cases:

- If there are sky images in the input frameset.
- If the number of object frames is less than the parameter nmin_comb.

Otherwise a background frame is produced for each input object frame with filename `hawki_step_compute_bkg_xxx` where xxx stands for the serie number of the input object image.

8.10.3 Quality control

This recipe does not compute any quality control.

8.10.4 Parameters

- *--nmin_comb*: The minimum number of object frames to compute a running median. If the current number is below this parameter a single background is made averaging the object frames (default is 10).
- *--nhalf_window*: Number of images at both sides of the current image to compute the running median bkg (default is 7).
- *--rejlow*: The number of frames with low level to reject (default is 3).
- *--rejhigh*: The number of frames with high level to reject (default is 3).

8.11 hawki_step_subtract_bkg

This recipe subtracts the background of all the input object images.

8.11.1 Input

The recipe expects frames marked as BASIC_CALIBRATED for the object images and background images marked as BKG_IM.

8.11.2 Products

The recipe produces background subtracted images with filenames `hawki_step_subtract_bkg_xxx.fits` (PRO CATG = BKG_SUBTRACTED), one for each input image.

8.11.3 Quality control

This recipe does not compute any quality control.

8.11.4 Parameters

This recipe accepts no parameters.

8.12 hawki_step_apply_dist

This recipe applies the distortion correction to the input frames

8.12.1 Input

The recipe expects frames marked with BKG_SUBTRACTED. The distortion is retrieved from frames marked as DISTORTION_X, DISTORTION_Y.

8.12.2 Products

The recipe produces files named as `hawki_step_apply_dist_xxx.fits` (PRO CATG = DIST_CORRECTED).

8.12.3 Quality control

This recipe does not compute any quality control.

8.12.4 Parameters

This recipe accepts no parameters.

8.13 hawki_step_refine_offsets

This recipe computes refined offsets using correlation between the images at the brightest objects. It achieves subpixel accuracy.

8.13.1 Input

The recipe takes as an input a set of frames marked as DIST_CORRECTED or alternative as BKG_SUBTRACTED. Also, a frame containing detected objects information and marked as OBJ_PARAM must be provided.

8.13.2 Products

The recipe produces only one product: a table with four extensions, one for each detector (PRO CATG = OFFSETS_REFINED). In each extension there is a record for each input image that gives the refined offsets. Note that the same frame can have different offsets in each detector, depending on the stars used for correlation in that detector.

8.13.3 Parameters

The recipe has the following parameters:

- *--xcorr*: Cross-correlation parameters. It is a set of four parameters sx, sy, mx, my. sx and sy are the search sizes of the correlation points, while mx, my are the measurement sizes for the correlation.
- *--nbrightest*: Number of brightest objects in the OBJ_PARAM frame to use for the correlation. This number is applied for each detector.

8.13.4 Quality control

This recipe does not compute any quality control.

8.14 hawki_step_combine

This recipe combines jittered images into one single image taking into account the offsets.

8.14.1 Input

The recipe expects frames marked as DIST_CORRECTED or as BKG_SUBTRACTED. This allows to skip the step *hawki_step_distortion*. The offsets can be provided in frame marked as OFFSETS_REFINED. If missing, the offsets are read from the header of input frames (keywords ESO SEQ CUMOFFSET[X, Y]).

8.14.2 Products

The recipe creates the file `hawki_step_combine.fits` (PRO CATG = COMBINED) which contains the combined image in 4 extensions, one for each detector.

8.14.3 Quality control

- QC COMBINED CUMOFFSETX: X Position of the combined image in the reference system used by the input offsets
- QC COMBINED CUMOFFSETY: Y Position of the combined image in the reference system used by the input offsets
- QC COMBINED POSX: X Position of the first raw image in the combined image
- QC COMBINED POSY: Y Position of the first raw image in the combined image

8.14.4 Parameters

- *--offset_max*: Maximum offset allowed (default is 1500)
- *--comb_meth*: Combination method for the stacking (union/inter/first) (default is union)
- *--rej*: Low and high number of rejected values in the combination algorithm (default is [0,0])
- *--borders*: Number of borders pixels that are trimmed rejected (default is 2)

8.15 hawki_step_detect_obj

This recipe detects the objects above a certain threshold and characterizes their properties.

8.15.1 Input

The recipe expects just one frame marked as COMBINED.

8.15.2 Products

The recipe creates two products: `hawki_step_detect_obj_mask.fits` (PRO CATG = OBJ_MASK) which marks all the pixels that are regarded as belonging to an object and `hawki_step_detect_obj_stars.fits` (PRO CATG = OBJ_PARAM) which contains several properties for each object detected.

8.15.3 Quality control

The following parameters are obtained using the algorithm described in section 9.4:

- QC IQ: The Image Quality
- QC FWHM PIX: The average FWHM in pixels
- QC FWHM ARCSEC: The average FWHM in arc seconds
- QC FWHM MODE: The FWHM mode

8.15.4 Parameters

- *--sigma_det*: detection level above the sigma level of the background (default is 6.0)
- *--growing_radius*: radius of convolution kernel to apply to objects. This convolution is applied after the initial detection, and allows to mask the tails of objects (default is 5.0)

8.16 hawki_step_photom_2mass

This recipe computes the photometry of a given field based on the 2MASS catalog.

8.16.1 Input

The recipe expects three inputs, first the combined image (COMBINED), the detected objects by recipe *hawki_step_detect_*
and finally the master table of the 2MASS catalog in the special crafted ESO format (CAT_2MASS). Additionally, at execution time, the recipe will try to access the appropriate 2MASS file associated to sky are observed. Please contact `psd_hawki@eso.org` to have access to this files.

8.16.2 Products

The recipe produces a single file with the photometric characteristics of all the stars and the mean zpoint solution (ZPOINT_RES).

8.16.3 Quality control

- QC ZPOINT: The computed zero point (one per each chip)

8.16.4 Parameters

This recipe accepts no parameters.

8.17 hawki_step_stitch

This recipe creates a single image out of the four HAWK-I chips.

8.17.1 Input

The recipe expects one frame marked as COMBINED.

8.17.2 Products

The recipe produces a single frame with only one extension named `hawki_step_stitch.fits` (PRO CATG = STITCHED).

8.17.3 Quality control

This recipe does not compute any quality control.

8.17.4 Parameters

This recipe accepts no parameters.

8.18 hawki_step_stats

This recipe computes statistics for each of the input frames and each of the HAWK-I detectors.

8.18.1 Input

The recipe expects frames marked with one of the following tags: JITTER_OBS, JITTER_SKY, FLAT_IM, DARK_IM or ZPOINT.

8.18.2 Products

The pipeline produces a single table named `hawki_step_stats.fits`. Depending on the kind of input frames the keyword PRO CATG is JITTER_STATS, BKG_STATS, FLAT_STATS, DARK_STATS or ZPOINT_STATS.

8.18.3 Quality control

This recipe does not compute any quality control.

8.18.4 Parameters

This recipe accepts no parameters.

8.19 hawki_util_gendist

This recipe creates a FITS table from an ASCII distortion file.

8.19.1 Input

This recipe takes as an input four ASCII files marked as DISTMAP_RAW, one for each detector

8.19.2 Products

The recipe produces a FITS table with the same information as the ASCII ones (PRO CATG = DISTORTION).

8.19.3 Quality control

This recipe does not compute any quality control.

8.19.4 Parameters

This recipe accepts no parameters.

8.20 hawki_util_stdstars

This recipe creates a FITS table with a standard stars catalogue.

8.20.1 Input

The recipe expects text files marked as STDSTAR_CAT.

8.20.2 Products

The recipe produces a FITS table which contains all the stars in the input catalogues. This table has PRO CATG = STDSTARS_CATS.

8.20.3 Quality control

This recipe does not compute any quality control.

8.20.4 Parameters

This recipe accepts no parameters.

9 Algorithms

In this section the data reduction procedures applied by the 6 pipeline recipes are described in some detail.

9.1 General Algorithms

9.1.1 Cross-correlation

The cross-correlation is used to find the exact shift between two images containing the same stars field.

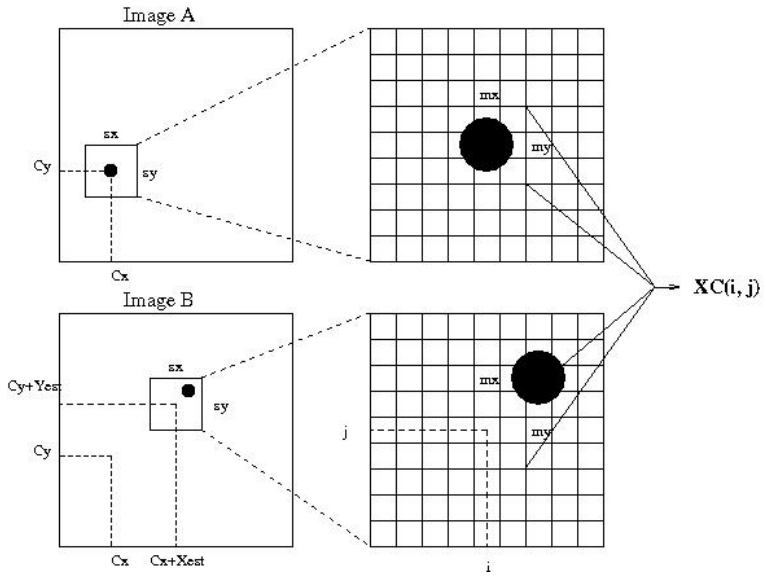


Figure 9.1.1: *Cross-Correlation algorithm.*

We suppose that we have

- two images A and B that represent about the same field, but that are shifted by (X,Y) pixels
- a rough estimate of the shift (Xest, Yest)
- a search size (sx, sy) and a measure size (mx, my)
- a correlation point in the first image (Cx, Cy)

Each pixel in a box of size (sx, sy) around the position (Cx+Xest, Cy+Yest) in image B is candidate for being the one that looks like the pixel at position (Cx, Cy) in image A. For each of those pixels (i, j), a cross-correlation factor XC(i, j) is computed. The pixel with the lowest factor 'wins'.

The cross-correlation factor is the normalised sum of the square difference of the pixels of the two images in a box of size (mx, my). The more the two windows look alike, the lower the factor will be.

The figure 9.1.1 gives an illustration of the procedure. To have a sub-pixel precision a fit of the cross-correlation values is computed around the minimum to get the 'real' minimum, and its position that corresponds to the precise sub-pixel shift we are looking for.

9.1.2 Distortion correction

Each pixel of each chip is forced to fit in a single rectangular "meta detector" array by simple integer offsets. The pixel thus defined will have positional errors arising from many sources, e.g. rotated chips, non-integer shifts, non-planar detectors, optical distortions, atmospheric refraction, etc. All of these will contribute to what we call "distortion." By monitoring the coefficients of the "distortion" correction we can keep track of the stability of the instrument.

Distortion map format The pipeline distribution package contains 4 ASCII files (see section 6) describing the distortion in the 4 HAWK-I detectors.

```
#
# file          hawki_util_gendist.fits
# extensions    4
# ---------------------------------------------
# XTENSION      1
# Number of columns 4
#
      DXGC       DYGC        I       J
         0       -200      512     512
         0       -200      512    1024
      -200          0      512    1536
       200          0     1024     512
         0          0     1024    1024
      -200          0     1024    1536
       200          0     1536     512
         0        200     1536    1024
         0        200     1536    1536
# ---------------------------------------------
# XTENSION      2
# Number of columns 4
#
      DXGC       DYGC        I       J
      -200          0      512     512
      -200          0      512    1024
      -200          0      512    1536
         0          0     1024     512
         0          0     1024    1024
         0          0     1024    1536
       200          0     1536     512
       200          0     1536    1024
       200          0     1536    1536
# ---------------------------------------------
# XTENSION      3
# Number of columns 4
#
      DXGC       DYGC        I       J
      -200       -200      512     512
      -200          0      512    1024
      -200        200      512    1536
         0       -200     1024     512
         0          0     1024    1024
         0        200     1024    1536
       200       -200     1536     512
       200          0     1536    1024
       200        200     1536    1536
# ---------------------------------------------
# XTENSION      4
# Number of columns 4
#
      DXGC       DYGC        I       J
       200        200      512     512
       200          0      512    1024
       200       -200      512    1536
         0        200     1024     512
         0          0     1024    1024
         0       -200     1024    1536
      -200        200     1536     512
      -200          0     1536    1024
      -200       -200     1536    1536
```

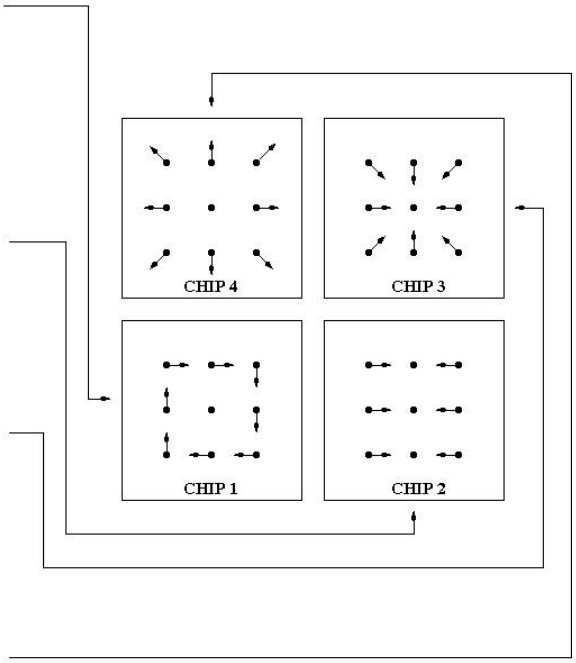


Figure 9.1.2: *Distortion files format. The arrows describe an hypothetical distortion effect, the specified shifts are the ones for the correction (opposite sign). This does NOT describe the HAWK-I distortion, it is a simple example used for the format description.*

These files are passed to *hawki_util_gendist* to generate the Multi-Extensions distortion calibration FITS file (PRO.CATG = DISTORTION).

This Multi-Extensions FITS file can then be passed to *hawki_step_distortion* or *hawki_sci_jitter* to correct for the distortion.

The pipeline user is free to specify any distortion correction he wants in these 4 ASCII files to have this specified distortion corrected later on.

As shown in figure 9.1.2, each ASCII file contains the distortion description for a given detector. It contains the X and Y shifts describing the distortion for the given pixel. The distortion correction will later apply the opposite of these specified shifts. The other pixels shifts are deduced from the grid by interpolation.

Distortion correction algorithm The distortion correction procedure uses this grid to find out for each pixel on each detector which correction to apply. The X and Y shifts for a given pixel are interpolated from the X and Y shifts specified on the 4 neighbor points on the grid as illustrated in figure 9.1.3.

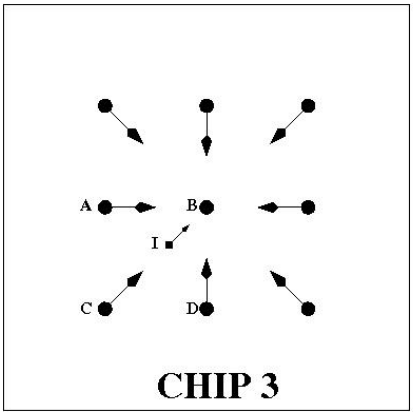


Figure 9.1.3: *Distortion correction procedure.*

Once the X and Y shifts are known for all pixels, the distortion correction is applied using a hyperbolic tangent interpolation kernel.

Distortion used for HAWK-I The actual distortion of the HAWK-I detectors has been precisely measured by L.R. Bedin. The whole process is described in appendix A and the distortion results are illustrated in Figure 9.1.4.

9.1.3 Distortion computation

Refer to Appendix A.

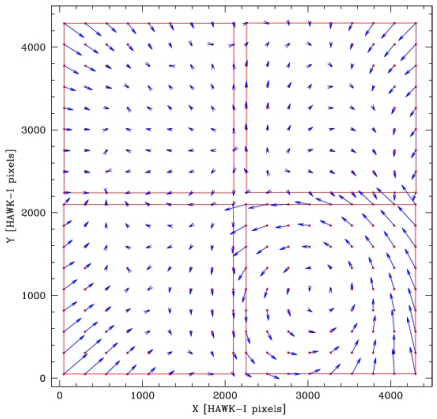


Figure 9.1.4: *Distortion correction effects.*

9.1.4 Detector noise model

Each pixel of an image that comes from the detector is a measurement of the real photon flux that falls into that pixel. The error in that measurement has two components, photon noise and readout noise. For infrared detectors using the follow-up-the-ramp technique, it is shown ([14]) that the variance in each pixel can be expressed as:

$$\sigma^2(i,j) = \frac{6S(i,j)}{5gn_{nds}n_{dit}}(\frac{n_{nds}^2+1}{n_{nds}+1}) + \frac{12RON^2(i,j)}{g^2n_{nds}n_{dit}}(\frac{n_{nds}-1}{n_{nds}+1}) \quad (1)$$

where: $\sigma^2(i,j)$ is the variance in ADU ^{2} for pixel (i,j). $S(i,j)$ is the signal in ADU for each pixel already removed from bias, g is the gain of the detector (in electrons/ADU), n_{nds} is the number of non-destructive reads (ndsamples), n_{ndit} is the number of total integrations (ndit) and $RON(i,j)$ is the noise (in ADU) introduced in a single readout.

9.2 Error propagation

Error propagation through the recipe is done actually for the dark recipe, although more recipes will implement it in the near future. The idea behind optimal error propagation is that *all* the input data of the recipe includes the error estimates and *all* the recipe algorithms propagate the error using the common error propagation theory. Note that in some cases there are quantities with unknown uncertainties, and therefore their contribution to the error budget is not taken into account.

The error propagation is performed for each recipe in the following cases:

- *hawki_cal_dark* recipe. If parameters - -*gain* and - -*ron* are explictly set.

We will briefly describe here the general error propagation theory.

If a quantity y is a function of n independent variables x_i

$$y = y(x_1, x_2, ..., x_n)$$

the variance of y, $(\Delta y)^2$, can be derived from the variances of the independent variables applying the usual formula for error propagation

$$(\Delta y)^2 = \sum_{i=1}^{n} \Big(\frac{\partial y}{\partial x_i}\Big)^2 (\Delta x_i)^2 \tag{2}$$

However, this is valid as long as the variables x_i are independent from each other: if the variable x_i correlates with the variable x_j, then their covariance

$$C_{ij} = \mathrm{cov}(x_i, x_j)$$

will be different from zero (by definition). In this case not just the variances but also the covariances of the independent variables x_i must be propagated. Variances and covariances are collected in the $n \times n$ covariance matrix $\mathbf{C}$. The variances lie on the diagonal of the matrix, since

$$C_{ii} = \mathrm{cov}(x_i, x_i) = \mathrm{var}(x_i) = (\Delta x_i)^2$$

Moreover $\mathbf{C}$ is a symmetric matrix, because

$$C_{ij} = \mathrm{cov}(x_i, x_j) = \mathrm{cov}(x_j, x_i) = C_{ji}$$

With non-diagonal elements different from zero, the full expression for the error propagation is

$$(\Delta y)^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial y}{\partial x_i} \frac{\partial y}{\partial x_j} C_{ij}$$

which turns into the usual formula in case the covariance matrix $\mathbf{C}$ were diagonal.

To further generalise, let us now suppose to have m quantities y_k ($1 \leq k \leq m$), depending on n values x_i ($1 \leq i \leq n$). In this case propagating errors means to transform the covariance matrix $\mathbf{C}$, related to the x_i, into the covariance matrix $\mathbf{G}$ related to the y_k. This is obtained by applying the most general form of the error propagation,

$$G_{kl} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial y_k}{\partial x_i} \frac{\partial y_l}{\partial x_j} C_{ij}$$

where

$$G_{kl} = \mathrm{cov}(y_k, y_l)$$

and

$$G_{kk} = \mathrm{var}(y_k) = (\Delta y_k)^2$$

for the diagonal terms. In matrix notation, if $\mathbf{J}$ is the Jacobian of the transformation from the x's to the y's

$$J_{ij} = \frac{\partial y_i}{\partial x_j}$$

the error propagation formula is a transformation of the covariance matrix:

$$\mathbf{G} = \mathbf{JCJ^T}$$

The covariance matrix $\mathbf{G}$ is the one to apply for further error propagation into new quantities computed in terms of y_k's. In summary, error propagation is nothing but the covariance matrix propagation.[4]

Note that in the special case $\mathbf{C}$ were diagonal, $\mathbf{G}$ might still contain non-diagonal elements. The formula to apply would become

$$G_{kl} = \sum_{i=1}^{n} \frac{\partial y_k}{\partial x_i} \frac{\partial y_l}{\partial x_i} (\Delta x_i)^2$$

9.3 Offsets criteria

Both the nominal header offsets (keywords SEQ CUMOFFSET[X,Y]) and the refined offsets are in the telescope reference system. For two images with offsets off_i and off_j, a given object may appear at positions pos_i and pos_j respectively. They are related by the following formula:

$$pos_i - off_j = pos_i - off_j \tag{3}$$

9.4 Object photometry

Recipes *hawki_sci_jitter* and *hawki_step_detect_obj* compute object photometry using the IQE method, inherited from MIDAS and which follows the following algorithm:

- Obtain background in a box around the object using a kappa sigma clipping algorithm.
- Get the centroid of the object using a simple moment.
- Perform a 2D gaussian fit to the source.
- The objects for which the following conditions hold are disregarded:
  - It fails to compute the gaussian fit.
  - FWHM is out of the range (0.1, 5.0).
  - Objects with $2 * \frac{fwhm_{maj} - fwhm_{min}}{fwhm_{maj} + fwhm_min} > 0.2$.

The size of the box used for photometry is 20x20 pixels.

The recipe *hawki_cal_zpoint* uses instead an aperture photometry described in section 9.7.6.

[4] This statement is valid under the assumption that the first order derivatives of the transformation are all that is needed for propagating errors. The error propagation formula is derived by Taylor expansion of the transformation, where higher order terms are neglected.

9.5 Photometric solution

The photometric solution created by recipe 9.7.6 contains both the zpoint (`ZPOINT` column) and the zpoint at airmass of 1 (`ATX0` column). It is this last value the one that has to be used to compute the magnitude of an object in the science images:

$$m = -ATX0 + E_0 A - 2.5log(counts/s) \quad (4)$$

where E_0 is the extincion and A is the airmass.

9.6 Bad pixel mask creation

The detector bad pixel masks in the HAWK-I pipeline are created in two phases:

- Hot bad pixel masks (PRO CATG = BPM_HOT). This is created by recipe *hawki_cal_dark* inspecting the pixels which are too deviant in the master dark. A sigma-clipping algorithm is used for that (with $\sigma = 10$ being the default value, controlled by parameter - -*sigma*.
- Cold pixel mask (PRO CATG = BPM_COLD). This is created from the master flat by recipe *hawki_cal_flat* combining the pixels rejected in two ways:
  - a sigma-clipping algorithm similar to the hot pixel using parameter - -*sigma_bpm* (which defaults to 10).
  - an absolute value rejection: all the pixels below and above a certain value are regarded as bad (the default values are - -*lowval_bpm* = 0.1 and - -*highval_bpm* = 10, after flat normalization).
- The final maks (PRO CATG = BPM) is created combining the hot and cold mask, and it is done by the *hawki_cal_flat* recipe.

9.7 Recipes Algorithms

Apart from very specific cases, the recipes reduce the four detectors independently and separately. The raw files contain 4 extensions with the 4 detectors images (identified with the EXTNAME keyword), the products, in the same way, contain the results (images or tables) stored in 4 extensions (identified with the EXTNAME keyword). The QC parameters are stored in the different extensions headers.

9.7.1 hawki_cal_dark

The darks are reduced with the *hawki_cal_dark* recipe.

The recipe produces one master dark by computing the average with rejection of the highest value of the input images. The final master dark is divided by the exposure time to get a master dark in ADU/s units.

If the parameters gain and ron are set, the recipe computes the uncertainty in each pixel of the master dark. This is computed using error propagation for each raw image given by formula 2 (the final variance is the sum of the

raw variances). The raw images are modeled to have an uncertainty modeled as in 9.1.4. Take into account that the ron parameter is the readout noise for a single readout and therefore may be different from the readout noise measured in the image.

The Read-Out Noise (RON) is also measured and written as QC parameter. For each frame, the next frame is subtracted from the current one. The following measurement is applied to the resulting image:

- Generate 100 13x13 windows on the input pixel surface. These windows are optimally scattered using a Poisson distribution to make sure they sample the whole area with as little overlap as possible.
- Compute the pixel standard deviation in each window.
- The readout noise is the median of all these measured standard deviations multiplied by $\sqrt{\frac{NDIT}{2}}$.

Besides, statistics are computed on the central zone of the input images, as well as on the 32 Video Channels (see Figure 9.7.1).

The hot pixels map is eventually created from the master dark with a simple 10-sigma threshold.

The results are written as QC parameters.

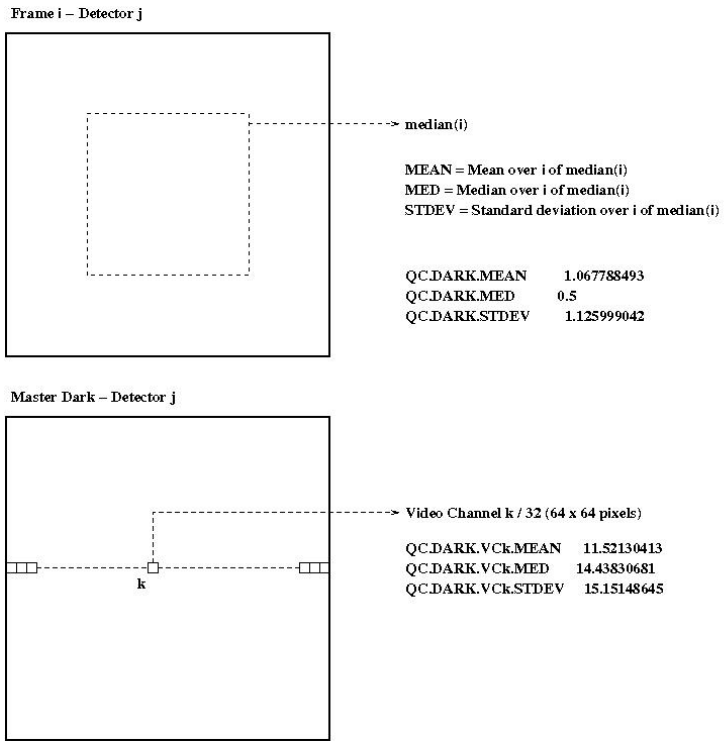


Figure 9.7.1: *Statistics on dark images.*

9.7.2 hawki_cal_flat

The flat field is created using twilight flats. The flats are derived by imaging the zenith with the tracking turned off. More than 30 exposures with constant DIT and NDIT are taken for each filter and a robust linear fit between the flux in each pixel and the median flux of all pixels is used to produce the flat field.

The implemented algorithm is the following:

- The data are classified by filter and DIT, and each different setting is separately reduced. The following steps apply for each setting.
- Various statistics are computed on the input frames (minimum, maximum, median, standard deviation)
- The minimum, maximum, median and standard deviation of the previously computed medians values are stored as QC parameters
- Apply a selection on the frames that will be used for the fit. The selection follows the following steps:
  - Flats with median level below - -*select_min_level* are rejected.
  - Flats with median level above - -*select_max_level* are rejected.
  - Flats with rms level above - -*select_max_rms* are rejected.
  - Flats with median level above - -*select_max_level* are rejected.
  - - -*If auto_flag* is set, a histogram is constructed with - -*auto_max_bins* bins or number of remaining valid flats if the later es smaller. For each bin, a frame whit median closest to the center of the bin is selected. For the first bin, it is chosen the upper bound of the bin instead of the center, while for the last bin it is chosen the lower bound of the bin. Note that the selected frame may be out of the bin bounds.
- Subtract the dark if available
- Each pixel values is fitted against the median of the whole frame.
- A second pass is applied: The pixels where the fit has a high error are recomputed. For these given pixels, the outliers are removed and the fit is recomputed. This second pass is necessary to remove the effect (star trace in the master flat) caused by stars crossing the field during the observation. This step is quite time consuming.
- If requested, the final flat is normalised by its median value.
- The cold pixel map is computed by a simple threshold applied on the computed flat image.

A 4-extensions master flat is produced for each subset of frames with the same filter. The other products are the errors of the fits, the map of the recomputed pixels, the cold pixel map, a table with the statistics results, and a bad pixel map (only if a hot pixels map is provided in input).

The flat field frames of each image is independently determined, and thus do not contain information about the relative gain variations between chips.

9.7.3 hawki_cal_illum

The illumination correction takes into account low-frequency differences between the true flat and the twilight flat. These variations have an RMS of 2%. If this is accurate enough, then an illumination correction is not needed.

The recipe expects in input a series of standard star observations moved across the 4 detectors over a regularly spaced grid (around 25 positions).

The recipe uses the header information to find out in which detector the star appears, and to guess its approximate position.

Some calibration data (flat field and bad pixels map) can be passed as well.

The implemented algorithm is the following:

- The calibrations are used to correct the input raw frames.
- The precise positions of the star on the grid are derived from the images with the use of the header offsets
- The photometry of each star is computed using a radius derived from the measurement of the star FWHM.
- 3rd degree, two dimensions polynomial is fitted to the flux as a function of position on the array. The polynomial image is generated, normalised by the center value and created as a product. An other product is the table containing the star positions and flux.

9.7.4 hawki_cal_lingain

For a detailed description of the *DETMON* algorithms, please refer to appendix B.

9.7.5 hawki_cal_distortion

This recipe uses almost the same algorithm as exposed in A. We refer the reader there for more information. In addition to the steps taken there to compute the distortion, this recipe performs one more step:

- Fit a linear component to the computed distortion and subtract it. This will remove the relative distortion introduced by atmosphere. Option - - *subtract_linear* can be used to control the execution of this step.

9.7.6 hawki_cal_zpoint

Standard stars are observed regularly in the J, H, K and Y filters.

Here are the steps taken by the pipeline:

The recipe algorithm is the following:

1. Determine the sky coordinates of the star. If parameters *--ra, --dec* are different from 999, then use them as the J2000 coordinates of the standard star. Otherwise, use the coordinates pointed by keywords `ESO TEL TARG ALPHA, ESO TEL TARG DELTA`.
2. Consult in the standard stars frame the closest star to the standard star coordinates (determined in previous step). In principle there could be several extensions in the standard stars frame, created from several ASCII files. The pipeline includes the file `HI_GSSC_081103A_stdstars_cat.fits` created from a subset of Persson and UKIRT standards. See 6.1 for details. Note that the coordinates are assumed to be J2000.
3. Correct the flatfield and the bad pixels if they are provided
4. For each detector select the frame where the star is supposed to be, using the offsets from the header. The remaining frames are averaged to compute the sky background.
5. Filter the image with a 3x3 median kernel to decrease noise.
6. Detect objects using the *--detect_sigma* parameter as a threshold.
7. Identify the star as the closest object to the theoretical star position. The theoretical star is found from the *--ra, dec* parameters and the WCS header, unless the *--xcoord, ycoord* parameters are different from -1 (either X or Y), in which case those values are used. Note that these parameters can be setup different for each chip. For instance, if *--xcoord=[-1, -1, 45., 43], --ycoord=[-1, -1, 156, -1]*, the expected position will be computed using WCS for detectors 1, 2 and 4, while for detector 3 it will be [45, 156].
8. Find the star precise position, its FWHM and compute its photometry. This uses aperture photometry with background computed between *--phot_bg_r1, --phot_bg_r2* and star flux integrated within *--phot_star_radius*. If *--phot_star_radius* is less than 0, a radius of 4 times the FWHM is used.
9. For each filter, computes the Zero Point (QC ZPOINT), extinction-corrected ZPs (QC ATX0) table for the single chips in each of the extensions of the PRO.CATG=ZPOINT_IMA product.
10. The average ZPs (in the primary header of the ZPOINT_IMA product), computed by simple mean of the values of the 4 chips.

Note that the pipeline reduction will fail completely and not generate any output at all if any of the following conditions hold:

- Not precisely 4 exposures in the template
- Filter not in the catalogue, currently the catalogue contains only H, J and Ks filters
- Standard star not in the catalogue.

The standard star database contains about 100 stars with magnitudes in the J, H, K and Y bands, although most stars only have magnitudes in a subset of these filters.

The conversion formula from ADUs to magnitudes is:

$zmag = mag + 2.5 * log_{10}(flux) - 2.5 * log_{10}(DIT)$

where:

- $zmag$ is the computed zero-point.
- mag is the known magnitude of the standard star in the observed band.
- $flux$ is the measured flux in ADUs in the image.
- DIT is the detector integration time.

The zero point is stored in keyword QC ZPOINT. The QC ATX0 contains the parameter:

$$atx_0 = zp + airmass * extinction \tag{5}$$

where $airmass$ is the mean between keyword ESO TEL AIRM START in the first image and keyword ESO TEL AIRM END in the last image of the zpoint series. $extinction$ is tabulated in 9.7.0

Table 9.7.0: Extinctions for each filter band

Band	Extinction
J	0.098
H	0.039
K	0.065
Y	0.00

9.7.7 hawki_sci_jitter

The basic steps in reducing imaging data are:

1. Compute statistics for all the frames.
2. If max_nframes is greater than 0, the recipe proceeds with the first max_nframes for the next steps.
3. Flat Fielding

   If provided, the master flat is divided to all images. This will result in photometry that is consistent to the 2% level over the HAWK-I field of view (verified for the broad-band filters). If more accurate photometry is required, then an illumination correction should be applied.

   If we look at the flat fielded images, we can see that they are far from flat. There is a jump between the two halves of the array, and structures at intermediate (5-10 pixels) and large (several hundred pixels) scales. These structures have a variety of causes. The jump in the middle is caused by the fact that we have not removed the zero level offset perfectly. The structures at intermediate scales are probably caused by pupil ghosts and by dust that has moved. The structures at large scales are probably caused by scattered light. Most of these features are additive, so they are removed at the sky subtraction stage.
4. Flagging bad pixels, removing vignetted regions

   The recipe can replace bad pixels by an average of their valid neighbors if a master bad pixel map is provided.

5. Sky subtraction

   This is the most important step and great care and a good understanding of the technique are necessary if good results are required. This is particularly important for deep imaging as an error at the 0.01% level will significantly effect the photometry of the faintest sources.

   For each input object image, a sky frame is created and subtracted to the object frame. If in the input set of frame, there are some sky frames provided, the sky estimation is simply the median of those sky frames. This median is then used for all the object frames.

   In most cases, there are no sky frames taken, and the sky estimation has to be computed from the object frames. In this case, we take advantage of the fact that we observe in jitter mode, and that a bright object does not stay at the same position. This means that a given pixel only falls on an object for a minority of the input object frames. To use this, for each pixel of the detector, the sky value is the average (with rejection of outliers) of the same pixel values before and after the current frame.

   If there are to few object frames in input to use this method, a simple median of the object frames is used.

6. Offset refinement

   To register the sky-subtracted images to a common reference, it is necessary to precisely estimate the offsets between them. That is done if the *--refine* parameter is set.

   hawki_sci_jitter applies a 2D cross-correlation routine (see section 9.1.1) to determine the offsets to an accuracy of 1/10th of a pixel. The correlation is computed at the points specified in the file pointed by parameter *--objects*. If this parameter is not set, the two first images are subtracted and the brightest peak is taken as the correlation point. If the refinement fails the recipe exits.

   As an initial estimate of the frame offsets the recipe uses those found in the FITS headers (from keywords SEQ CUMOFFSETX, SEQ CUMOFFSETY) or alternatively, by the *--offsets* parameter. If the *--refine* parameter is not set, the recipe uses the initial estimate offsets.

7. Resampling and stacking

   Registering the images is done by resampling them with sub-pixel shifts to align them all to a common reference (usually the first frame). Resampling can make use of any interpolation algorithm (currently TANH is used), but be aware that using cheap and dirty algorithms like nearest-neighbor or linear interpolation can degrade the images by introducing aliasing.

   Stacking the resulting images is done by averaging them with a rejection of the outliers using a *minmax* algorithm (controlled by parameter `--rej`.

   There are two pair of keywords that inform about the offsets of this combined image. The raw images have nominal offsets in a given telescope reference system. The pair of keywords QC COMBINED CUMOFFSET[X,Y] report the position of the lower left corner of the combined image in that system. Additionally, the lower left corner of the first raw image would be placed at position QC COMBINED POS[X,Y] in the combined image.

8. Distortion correction

   The distortion is corrected if the distortion calibration files are included in the sof. Currently, the four chips combined images (COMBINED) are corrected. Strictly speaking, this introduces some error because a correction for the physical detector position X,Y is applied in the combined image, which is already a combination of different physical pixels. The distortion can be applied on the raw files individually to fix this, but for that one has to use the modular recipes (see 9.7.11).

9. Object detection

   A threshold mask is created using different detection levels (100., 90., 80., ..., 10., 5.) till at least one object is detected. A convolution with a 3x3 kernel is applied to avoid one pixel detections. Each zone of connected pixels in the previous mask is regarded as an object and a characterization of its fwhm, position, flux, etc.. is performed.

   The photometry of the detected objects is explained in section 9.4

10. Reconstruct the stitched image

    The 4 combined images from the 4 detectors are used to reconstruct the whole field. If the jitter shifts are big enough, the gaps are filled in. Note however that the stitching is done using the nominal gaps between detectors, so no relative rotation or distortion is taking into account. That means that the stitched image is meant only for visualization purposes and it is not recommended for scientific exploitation (specially when astrometry is important).

Figure 9.7.2 shows an example of a raw image and the result obtained with the *hawki_sci_jitter* recipe.

9.7.8 hawki_step_basic_calib

The basic steps in reducing imaging data are:

1. Dark subtraction

   The HAWK-I dark current is as high as 0.5 ADU/s. In order to get accurate photometry one should apply a master dark, which gives the dark counts per second. This step is only applied if a dark frame is provided.

2. Flat Fielding

   If provided, the master flat is divided to all images. This will result in photometry that is consistent to the 2% level over the HAWK-I field of view (verified for the broad-band filters). If more accurate photometry is required, then an illumination correction should be applied.

   If we look at the flat fielded images, we can see that they are far from flat. There is a jump between the two halves of the array, and structures at intermediate (5-10 pixels) and large (several hundred pixels) scales. These structures have a variety of causes. The jump in the middle is caused by the fact that we have not removed the zero level offset perfectly. The structures at intermediate scales are probably caused by pupil ghosts and by dust that has moved. The structures at large scales are probably caused by scattered light. Most of these features are additive, so they are removed at the sky subtraction stage.

3. Flagging bad pixels

   The recipe can replace bad pixels by an average of their valid neighbors if a master bad pixel map is provided.

9.7.9 hawki_step_compute_bkg

The recipes takes the following steps:

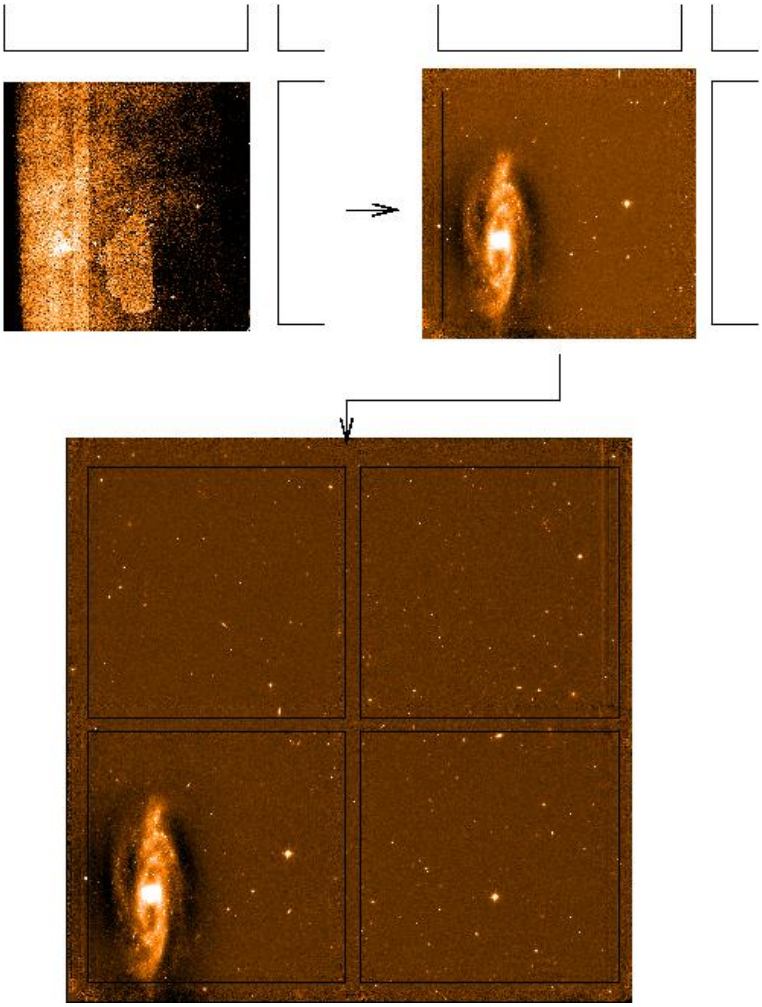


Figure 9.7.2: *Jitter result.*

1. If there are sky images the background is computed as the median of all the sky images. This background will be associated to all the input object images.

2. If there are not sky images there are two possibilities:

   (a) If the number of object images is lower than parameter nmin_comb a simple median of the object images is computed and associated to all the input object images.

   (b) The following steps are performed:

      i. If an object mask (`OBJ_MASK`) has been provided in the frameset, the mask is first inverse-corrected from distortion (if distortion frames are provided).

      ii. If an object mask is provided to the recipe, the mask is shifted for each individual image taking into account the offsets. The offsets are retrieved from a frameset if provided (`OFFSETS_REFINED`) or from the object header otherwise.

iii. For a given target image and a given pixel, a running buffer is computed with the neighboring images, the number of which is given by parameter *--half_window*. For the first image, for example, only the *half_window* images to the right are considered. For image `n`, the interval `[n - half_window,n + half_window]` is considered.

iv. Once we have a given set of background values after mask rejection, the parameters *--rejlow, --rejhigh* are used to reject extreme values (basically a *minmax* algorithm). After that the median is taken from the remaining values. In the case that no values are left, the pixel is added to the bad pixel mask of the background.

9.7.10 hawki_step_subtract_bkg

The steps taken by this recipe are:

1. If there is only one background image provided, this image is subtracted to all the input object images.
2. If there are several background images provided, its number must match those of input object images. A one to one match is done to correct each object image by its corresponding background image.

9.7.11 hawki_step_apply_dist

The steps taken by this recipe are:

1. The distortion table contains for several points in the image which are the distortions to apply. With this information a distortion map for each image pixel is created, using a linear interpolation with four nearest points in the table.
2. For each input image the distortion correction is performed warping the original image (see 9.1.2). Note that the final image will show correlations between data values of different pixels.
3. The WCS solution of the original images is retained, although now is only an approximate solution, since it does not take into account the correction of the distortion.

9.7.12 hawki_step_refine_offsets

The steps taken by this recipe are:

1. Read the nominal offsets from the header of each file (keywords SEQ CUMOFFSET[X,Y]).
2. Only the brightest objects are regarded as reference points. The number of selected frames is controlled by parameter nbrightest.
3. For each reference point a cross-correlation is performed in a square of size *--s_hx, --s_hy*. This correlation is done between each image and the first one, given as a result the offsets which provide the lowest squared difference of the correlation (see 9.1.1).

9.7.13 hawki_step_combine

The algorithm to combine to combine the images comprises the following actions:

1. The offsets are read from the refined offsets frame if provided or the header telescope nominal offsets otherwise. Note that in the first case there could be different offsets for each detector and the final combined image may have different dimensions in each detector.
2. Registering the images is done by resampling them with sub-pixel shifts to align them all to a common reference (usually the first frame). Resampling can make use of any interpolation algorithm, but be aware that using cheap and dirty algorithms like nearest-neighbor or linear interpolation can degrade the images by introducing aliasing.
3. Stacking the resulting images is done by averaging them with a rejection of the outliers.
4. There are two pair of keywords that inform about the offsets of this combined image. The raw images have nominal offsets in a given telescope reference system. The pair of keywords QC COMBINED CUMOFFSET[X,Y] report the position of the lower left corner of the combined image in that system. Additionally, the lower left corner of the first raw image would be placed at position QC COMBINED POS[X,Y] in the combined image.
5. The WCS of the combined image is derived from the first image WCS applying the relative offset.

9.7.14 hawki_step_detect_obj

The steps taken by this recipe are:

1. A threshold mask is created using the parameter - *-sigma_det*. A convolution with a 3x3 kernel is applied to avoid one pixel detections. Should this mask being used to mask object in the raw images one must provide the offsets and distortion to the *hawki_step_compute_bkg*.
2. If the parameter - *-growing_radius* is greater than 0, the mask is convoluted with an expansion kernel. The shape of the kernel is given by:

$$kernel(x,y) = 1 - \frac{\sqrt{(x-x_c)^2 + (y-y_c)^2}}{growing\_radius}$$

   where x_c, y_c are the coordinates of the pixel to convolve.
3. Each zone of connected pixels in the previous mask is regarded as an object and a characterization of its fwhm, position, flux, etc.. is performed.

9.7.15 hawki_step_photom_2mass

This recipe performs the following steps:

- Look in the WCS information which is the current pointing.
- Figure out which 2MASS files should be used and retrieve the stars enclosed in the HAWK-I field of view.
- Match the 2MASS stars with the detected stars (given in input with PRO CATG = OBJ_PARAM, created by recipe *hawki_step_detect_obj*)
- Compute photometric characterization for each matched star.

9.7.16 hawki_step_stitch

The 4 combined images are used to reconstruct the whole field. If the jitter shift are big enough, the gaps are reconstructed. The images are not rotated or resampled, only the nominal gaps between the detectors are used. That means that the stitched image is meant only for visualization purposes and it is not recommended for scientific exploitation (specially when astrometry is important).

A Geometrical distortion computation

This work has been achieved and documented by L.R. Bedin.

A.1 Overall strategy

We want to define astrometric fields for HAWK-I that would be useful for future monitoring of the variations in the geometrical distortion. In the process of defining these astrometric fields we will determine the present-day geometrical distortion of HAWK-I, and monitor its stability in the short therm.

A.2 Astrometric field calibration

To determine the Geometric distortion correction we will follow an auto-calibration approach analogous to the one described in Anderson et al. (2006, A&A, 454, 1029), where we obtained the geometrical correction for the Wide Field Imager at the focus of the 2.2m MPI ESO telescope.

Briefly, we observed the same part of the sky in one filter (J) with large dithers. The idea is to map the same patch of the sky into as many different place on the 4-chips detector as possible, avoiding repetition of the gaps.

Since the focus and/or the flexure and/or the general conditions will change during the observations (even within the same night) what we will find would be an *average* distortion correction.

A.3 Average distortion solution

1) The first step is to bring the 4 chip of each observation (image) into a common *meta-chip* coordinate system. We do this using only integer shifts. Comparing residuals from different images tell us empirically how much it is the total amount of the geometrical distortion, which amount to few pixels (see below the section on preliminary distortion solution).

2) The procedure to determine the geometric distortion correction is necessarily iterative. Here below we will give a brief summary of the procedure.

We parametrize the distortion solution by a look-up table of corrections for each chip that covered each 2048×2048-pixel chip, sampling every 256 pixels. This resulted in a 9×9 element array of corrections for each chip.

The distortion corrections for each stars (pixels) will then be the meta-chip positions plus the bi-linear interpolated value from the distortion correction table:

$$\begin{aligned} X_{\rm corr} &= x_{\rm meta} + \Delta x_{\rm GC}(x,y) \\ Y_{\rm corr} &= y_{\rm meta} + \Delta y_{\rm GC}(x,y). \end{aligned} \tag{6}$$

where $\Delta x_{\rm GC}(x,y)$ and $\Delta y_{\rm GC}(x,y)$ come from interpolating the table for that chip.

The challenge in finding an optimal distortion solution is then to find a set of table values which will allow to transform star list positions from one image into each other image using only linear transformations. So we are looking for a single set of table values that can be applied to all the images to minimize the non-linear transformation residuals.

Transformation residuals are constructed in this way.

- For each pair of images i, j we find all the stars common to the two images. Then N common stars give us N pairs of raw positions $(x_i, y_i; x_j, y_j)$.
- We first correct these positions using the current best distortion solution, so we get $(X_i, Y_i; X_j, Y_j)$ for each star.
- Using least square we find the best linear transformation between the two frames which use the current corrections.
- This allow us to compute residuals. For each of the N stars in image i we have $(x_i, y_i; \delta_{x_j}, \delta_{y_j})$. Where δ are the differences between where the star was found in image i and where its position in image j say should be in image i.
- We generate such residuals for each star in common to each image pair. This will result in many tens of millions of residuals (dense field, several images).
- Each residual have several contributions:
  a) the distortion error in one image;
  b) the distortion error in the other image;
  c) the measurement errors.
- If we examine all the residuals from all image-pairs for a particular region of the detector, the the average residual will be indicative of the distortion error at that chip location. The other contribution will cancel out.
- We examine the average residuals about each of the the distortion-array grid points.
- We begin our iteration with a null correction table. Next we adjust each grid point by half the recommended adjustment.
- We will smooth the distortion table with a 5×5 quadratic smoothing kernel to ensure smooth variations of the geometric distortion (we will verify that this will affect our solution).
- Once we have a solution, we solve for residuals, this time including the last correction in the computation of the residual.
- Residuals get smaller and smaller. Convergence is reached when the iteration-to-iteration adjustment for the distortion arrays is less then 0.005 pixels (which does not mean that individual frames can be corrected down to that precision).

A.4 Choice of the calibration fields

The choice of the calibration fields is driven by several parameters: right stellar density, high S/N in short amount of time, low airmass, availability of pre-existing astrometric reference fields.

A.4.1 Field 1

In the HST archive there is a mosaic of five WFC/ACS fields in the Bulge Baade window, for a total of $\sim 7' \times 7'$ [visible close to the Zenith from Paranal in August]. This provide an almost ideal astrometrically-calibrated reference field for HAWK-I. See Figure A.4.1 for a comparison of the mosaic of the 5 WFC/ACS field of view with the one of HAWK-I.

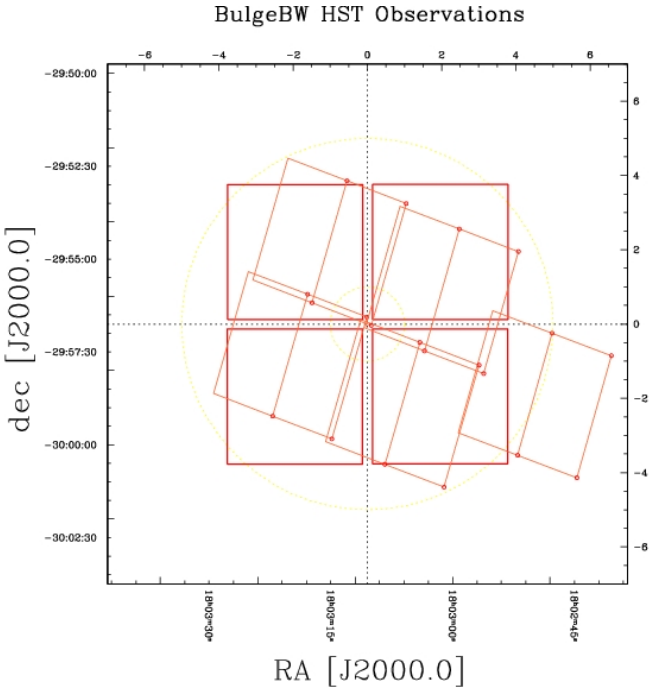


Figure A.4.1: *Footprint of the mosaic of WFC/ACS fields that we used to measure the preliminary geometrical distortion of HAWK-I. In thick line it is also over-imposed a footprint of the HAWK-I field of view (specifically dither 13, see the text).*.

Since WFC/ACS geometric distortion is corrected down to few mas, we are able to have an important cross-check of the goodness of our distortion solution, as done in Anderson et al. (2006, see Sect. 6). Furthermore, the same area has been imaged by WFI at MPI 2.2m ESO telescope, covering more than 30×30 arcmin.

In Sect. A.6, here below, we use the HST observations to derive a preliminary geometrical distortion solution for HAWK-I.

Bulge is ideal because of the flat spatial distribution of its stars across the HAWK-I field of view. However, there is the risk to be limited in the accuracy of our geometrical distortion correction by the stellar crowding and by the relatively low S/N of the Bulge stars.

A.4.2 Field 2

For these reasons we also observed a field in the geometrically closest Galactic cluster, M4 (NGC 6121). The advantage is to have several stars (in 10s we map stars around the TOs) with very high signal to noise. [The very

saturated bright sources in this field have been also used to study the persistence of images on the detector]. For this cluster too, several HST-archive-fields exist, but covering a smaller region than in the case of the Bulge. An obvious draw-back of using a star cluster is that its spatial stellar distribution is far from flat.

A.4.3 Field 3

Bulge's stars have a internal proper motion dispersion, with respect to their average motion, of 3 mas/yr, M4's stars have an internal proper motion dispersion of 0.5 mas/yr, and with respect to the field in foreground+background, M4's stars move at a rate of $\sim$20 mas/yr.

It is very valuable to build an astrometric reference field for future monitoring of the HAWK-I geometrical distortion solution not only in the short, and medium time, but also in the long therm (which, maybe, will be useful not only for HAWK-I).

This field should contain several stars that do not move a lot, and that have high enough S/N. Also the stars should cover an area on the sky of the size of HAWK-I field of view, and finally this field should be visible in August at a low air-masses.

So, a distant stellar cluster, or better a Local galaxy, far enough that its stars do not move more than 0.05 mas/yr, should do the job.

For these reasons we also observed the rather loose dwarf galaxy NGC 6822.

A.5 Observing strategy

We took as maximum duration of the single exposures DIT=10s. Each single pointing was repeated 6 consecutive times and staked together (NDIT=6). This minimize image-motions effects, give and enough S/N (we want to have at least 20,000 DEN (Digital Numbers) above the local sky for the brightest unsaturated stars, in their central pixel).

We will first calibrate the geometrical distortion for filter J. In order to do that, as described in Section A.3, we took as many as possible large dithers compatibly with the allocated time for these calibrations, and with the visibility of the objects, with-in the same night. The adopted dither pattern within and Observing Block (OB) is given in Table A.5.0.

Figure A.5.1 shows the dither pattern used for M4 and the Galactic Bulge in the Baade Window, and the labels that we will use for single images within the same OBs. [Note that for time restrictions, in the case of NGC 6822 in the OB we used only 9 dithers: 01, 03, 05, 11, 13, 15, 21, 23, and 25, although with a longer integration times: DIT=10s and NDIT=12.]

In the case of M4 we repeated in filter J this OBs, four times in four consecutive different nights, and anytime using a different reference starting point, according to the pattern (x,y) = (00;10), (05;10), (10;05), and (15,15).

The size of the dithers decide the scale of the geometrical distortions that can be corrected, in the sense that distortions smaller than those size could not be corrected.

For the Galactic Bulge, we repeated the same OBs, in three consecutive different nights, and anytime using a different filter among J, H, and K.

Table A.5.0: Chosen offsets for the telescope to map any part of the observed object in as many as possible different part of the detector.

Image_ID	Δ_X[arcsec]	Δ_Y[arcsec]
01	−225.00	−225.00
02	−206.25	−131.25
03	−187.50	−037.50
04	−168.75	+056.25
05	−150.00	+150.00
06	−131.25	−206.25
07	−112.50	−112.50
08	−093.75	−018.75
09	−075.00	+075.00
10	−056.25	+168.75
11	−037.50	−187.50
12	−018.75	−093.75
13	+000.00	+000.00
14	+018.75	+093.75
15	+037.50	+187.50
16	+056.25	−168.75
17	+075.00	−075.00
18	+093.75	+018.75
19	+112.50	+112.50
20	+131.25	+206.25
21	+150.00	−150.00
22	+168.75	−056.25
23	+187.50	+037.50
24	+206.25	+131.25
25	+225.00	+225.00

In the case of NGC 6822, instead, the 9-only-exposure OB has been repeated in filter J and K consecutively within the same night.

[For photometric purposes: within each OB, the images taken consecutively, will be combined in a way that these large dithers would be opportunely combined to determine the best estimate of the variable sky.]

To maximize the image quality and to minimize the DCR effects, all observations have been take as close as possible to the zenith and under most of the average seeing conditions under which the instrument is expected to work.

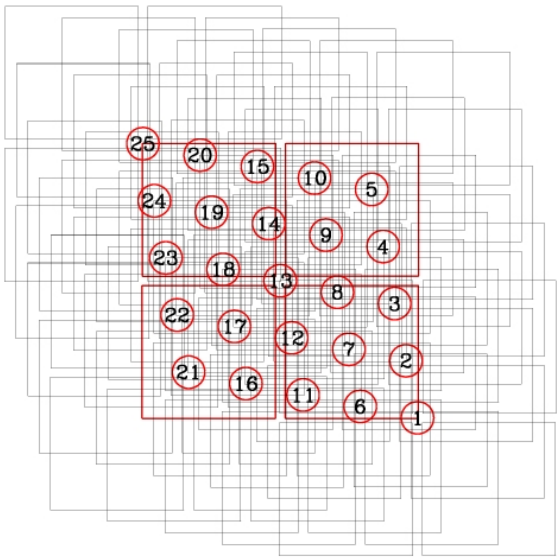


Figure A.5.1: *Dither pattern used for the auto-calibration of the HAWK-I geometrical distortion. The red circle marks the relative position of each offset, and the numbers within them report the dither ID of the image in the OB, according to the labels given in Table A.5.0. Image 13 is the center of our dither pattern (offset 0″.00,0″.00). We highlighted with a thick red line the HAWK-I footprint for this image..*

A.6 Preliminary geometrical distortion solution

Auto-calibration requires fast computers, and disk space. In this document we will not follow this approach (described above in Section A.3), but rather will do something quicker to deliver a preliminary distortion solution good down to fraction of a pixels, and limited to the temporal-window of the sub-set of images used.

Nevertheless, this will not hamper us to show the overall behavior of the geometrical distortion across the HAWK-I field of view. A following more careful analysis will merely refine the geometrical distortion that we derive here, and give a characterization of its temporal variations.

Here we use only a fraction of the data (five images of the Bulge Baade window during the night of August 2^{nd} 2007), and the astrometric calibrated catalog from HST observations (shown in Figure A.4.1). We will assume the HST catalog (which is good down to few milliarcsec) to be completely distortion free.

We match the point sources measured in five HAWK-I images (namely: dithers 01, 05, 13, 21, 25) with the ones in the HST astrometric catalog.

The results of this match, for image 13, is shown in Figure A.6.1 where for the common stars we plot, in the HAWK-I meta-chip reference system (defined above), the residuals of the raw positions vs. the geometrically corrected ones (from the HST catalog).

The median of these residuals within a box of ± 128, for each coordinate, gives the amount of the geometrical

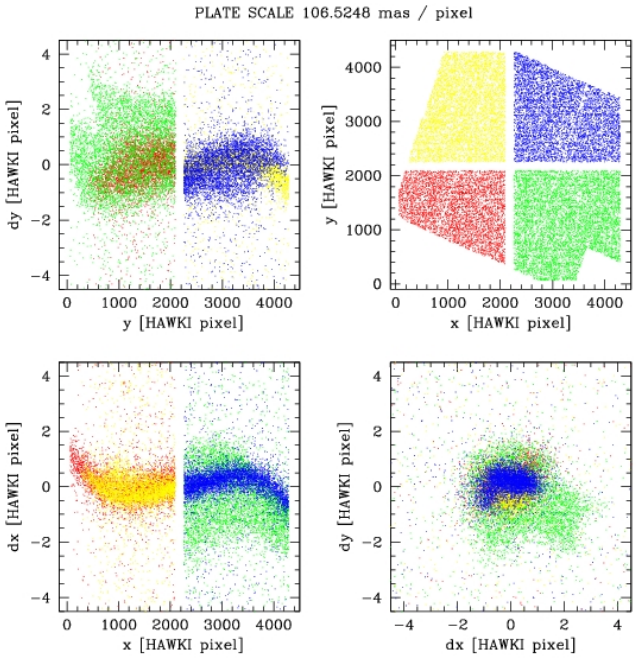


Figure A.6.1: (Top-Right): *Stars in common between HAWK-I in image 13, and the distortion free HST reference frame. Stars in different chip are showed in different color. All plots in units of HAWK-I pixels.* (Top-Left): *Residual in Y versus Y.* (Bottom-Left): *Residual in X versus X.* (Bottom-Left): *Residual in X versus residuals in Y.* .

distortion for the considered grid point of the look-up table of corrections that we defined above.

The result is shown in Figure A.6.2, where the size of the geometrical distortion has been exaggerated by a factor 100.

The final distortion correction will be delivered in the format illustrated in Table A.6.0 (however, note that those values are preliminary, and their purpose is merely indicative).

This table has 4×9×9=324 lines. To correct any pixels of the meta-chip system into a distortion free frame one need to consider the closest four grid points, and apply a bi-linear interpolation.

In Figure A.6.3 we repeat what shown in Figure A.6.1, after applying a preliminary geometrical distortion. Obtained from the one single image 13. Note that since HST observations were taken in 2003, the dispersion of the stars is consistent with the expected dispersion for the Bulge's star ($\sim$ 100km/s at $\sim$8kpc, $\sim$3mas/yr, in $\sim$4 years, $\sim$12 mas), even if the centroid algorithm we are using it is very crude.

A.7 Cross-checks

Of course Figure A.6.3 is not a proof of the goodness of our preliminary geometrical distortion, since we just verified that the average correction we applied, with our bi-linear interpolations, was actually done. A robust check of the goodness of our preliminary distortion is given by the r.m.s. of the positions of the stars from the

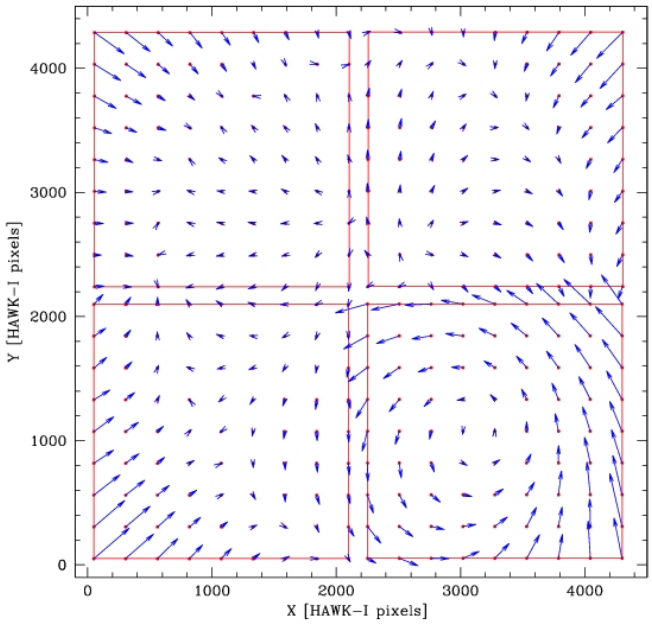


Figure A.6.2: *The red boxes show the positions of the single HAWK-I chips in the meta-chip system. The red dot are our grid points in the look-up table used to represent the geometrical distortion. The arrows give the sizes (exaggerated by a factor 100) and the directions of the correction to be applied to the grid-points to bring them into a distortion free frame. The correction for any pixels in the detector can be obtained with a bi-linear interpolation of the 4 closest grid-points..*

Table A.6.0: Look-up table that will be used to represent the geometrical distortion correction, where: ch = chip number; ni(nj) = grid point positions in the look-up table; dxcg(dygc) = amount of correction needed to be added to the raw x(y)-position, in the meta-chip coordinate system, to get a x(y)-position in a distortion free frame; i(j) = grid point positions in the chip (ranging from 0 to 2048); mi(mj) = grid point positions in the meta-chip coordinate system (ranging from 0 to 4400).

ch	ni	nj	dxgc	dygc	i	j	mi	mj
1	1	1	+2.84390	+2.45460	0000	0000	0050	0050
1	2	1	+2.23870	+1.98040	0256	0000	0306	0050
1	3	1	+1.63340	+1.60680	0512	0000	0562	0050
...	...	...	...	...	...	...	...	...
4	7	9	−0.97580	−1.51180	1536	2048	3791	4290
4	8	9	−1.58400	−1.84180	1792	2048	4047	4290
4	9	9	−2.06240	−2.08660	2048	2048	4303	4290

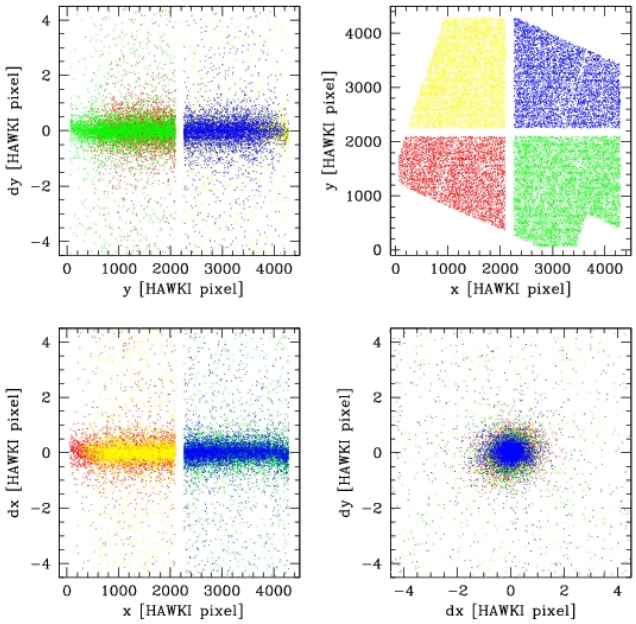


Figure A.6.3: *As in Figure A.6.1 but after applying a preliminary correction. The residuals, as expected, are now well below 1 pixels. [Note that here there are residual on the borders, those disappear once combined the five images used for the present preliminary measurement of the distortion.] .*

five HAWK-I images used here.

To do this we measured the raw positions of each source in each image, we corrected these positions with our look-up table, and brought all the corrected positions from all the five images into a common reference frame.

We selected on purpose the five images with the largest relative offsets, in a way that if the positions of the sources where overall well corrected, the positions from the single-exposures (measured in each image in a different part of the detector) should have small r.m.s. in the common reference frame.

These r.m.s. are function of the S/N of the single source, so the result of this test is show in Figure A.7.1, as a function of the source magnitude. In the bottom and it the top panel are shown the r.m.s. in X and Y positions, respectively, for the stars measured in three out the five considered frames (i.e. r.m.s. of 3 up to 5 values).

The adopted instrumental magnitude has been computed from the flux (in digital numbers DN) in a 5-pixel aperture, above the local sky (estimated as median of the pixels within a annulus of radii 20 and 30 pixels).

For bright stars, just before the saturation limit, the r.m.s. of the positions are as good as $\sim$0.1 HAWK-I pixels ($\sim$15 mas). The size of the r.m.s. being comprehensive of the goodness of this preliminary distortion correction (and its stability among the five considered frames), and the rather crude centering algorithm used here to measure the stars positions (which should be the dominant source of error for the best corrected part of the detector).

However, we noticed a departure of r.m.s. (as large as $\sim$0.5 pixels) in the corners, probably due to a not yet

perfect distortion correction, or to a frame-to-frame variation of the distortion, induced by focus variation, or to biases of the crude centering algorithms in the presence of a variable image-quality across the field of view. Work in progress.

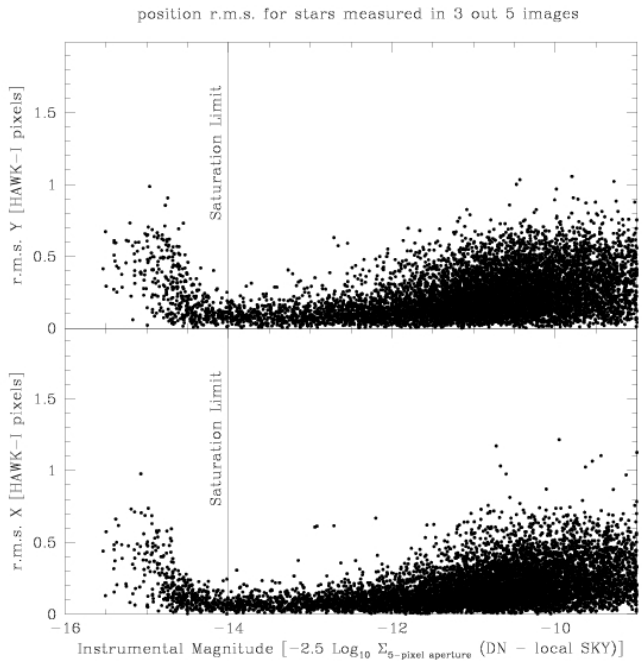


Figure A.7.1: *Size of the r.m.s. in X and Y positions, for stars measured in three out of five images. Saturation begin roughly at an instrumental magnitude of* ~ -14 *is indicated with a vertical line. .*

A.7.1 Orientation of CHIP2

The features that strike the most in Figure A.6.2 are the rotated position of chip 2, and the stretch of the scales toward the corners of the HAWK-I field of view.

If we assume the grid point (ch,ni,nj)=(2,4,5) as the center of the rotation of chip 2, we can measure along x and y axes, an average rotation, anti-clock wise, of $0^{\circ}.1127/$ and $0^{\circ}.085$.

A.7.2 Scale

Assuming a plate-scale of 49.7248 mas/pixels (from ACS-ISR-07-2007) for WFC/ACS, we derived an average plate scale for HAWK-I of 106.5248 mas/pixels.

A.7.3 Temporal stability

This issue will be carefully addressed in the future i.e. measuring precisely the short-time stability of VLT+HAWK-I

A.7.4 Immediate future

We have to keep in mind that this is quick preliminary measurement of the HAWK-I geometrical distortion correction.

When we will have fine tuned our PSF fitting algorithms (that will take a while) we will improve our results, and under take an extensive study of the geometrical distortion, its variation in the short, middle and long time scale, with the position of the co-rotator, maintenance, etc...

B DETMON project Algorithms

In this section the main algorithms implemented within the Detector Monitoring Project are described.

Relevant data reduction parameters are typed in **bold** face. For convenience we omit the common prefix **detmon.** for the recipe description as well as the step prefix name for the algorithm description.

There are 2 different methods to compute the Gain, Photon Transfer Curve and Median. We explain them here:

B.1 PTC Algorithm for Gain computation

For the gain reduction, a single algorithm is implemented for both optical and infrared, as the algorithm itself is the same for both. The inputs are different (in particular what here we indicate as *off* frame indicated in the near infrared an off-lamp frame, in the optical domain is replaced by a bias frame) but the algorithm is implemented to work with both cases.

Input frames are a set of pairs of on/off flats each set with a different DIT.

The frames are ordered by increasing DIT.

A table to hold results is created with the following columns:
DIT, EXPTIME, MEAN_ON1, MEAN_ON2, MEAN_OF1, MEAN_OF2, SIG_ON_DIF, SIG_OF_DIF, GAIN.

If we indicate with on_{dif} the on frames difference:

$$on_{dif} = on_2 - on_1 \tag{7}$$

If we indicate with of_{dif} the off frames difference:

$$of_{dif} = of_2 - of_1 \tag{8}$$

If we indicate with mon_i the on frame clean mean (obtained either with a histogram or with a kappa sigma clipping algorithm):

$$mon_i = mean(on_i) \tag{9}$$

And with mof_i the off frames clean mean:

$$mof_i = mean(of_i) \tag{10}$$

One may found the gain by fitting the following linear (one may even fit a parabola and verify that the coefficient of the quadratic term is small enough, this being a QC parameter) relation:

$$(mon_1 + mon_2) - (mof_1 + mof_2)_i = gain(e^-/ADU)(\sigma^2_{ondif} - \sigma^2_{ofdif})_i\alpha + a \tag{11}$$

where α is the autocorrelation function computed as:

$$\alpha = \sum_{m,n} R_{m,n} \tag{12}$$

where:

$$R_{m,n} = \frac{\sum_{i,j} V_{i,j} * V_{i+m,j+n}}{\sum_{i,j} V_{i,j}^2} \tag{13}$$

where $V_{i,j}$ is the intensity of the pixel (i, j).

The values of mon_1, mof_2, mof_1, mof_2, σ^1_{ondif}, σ^2_{ondif} from equation 11 together with the values of DIT and EXPTIME for each frame set should be logged in the corresponding columns of the product table (PRO.CATG=GAIN_INFO).

$$conad = gain^{-1} \tag{14}$$

The coefficient a and its error, the gain and its error should be computed and logged as QC parameters (QC NOISE, QC NOISE STD, in ADU and QC GAIN, QC GAIN STD, in e^-/ADU and QC CONAD in ADU/e^-, QC AUTCORFA).

The Read Out Noise is computed as:

$$READNOISE(e^{-1}) = \frac{\sigma_{ofdif}}{\sqrt{2}} \cdot gain \tag{15}$$

And monitored as QC parameter (QC READNOISE in e^-).

In eq. 11 it is critical to compute a clean mean and a clean standard deviation. Two methods are possible:

1. Build a histogram and compute the histogram centre and sigma.
2. Determine the mean and standard deviation repeating **niter** a clean mean computation with kappa-sigma cleaning of bad pixels.

B.2 Detector linearity computation (IR): irplib_detmon_nir_linear

The input interface is the same to irplib_detmon_gain, that is, for each exposure time value two pairs of frames (a pair ON and a pair OFF) are provided. The input frames DPR keyword should be:

DPR CATG	DPR TECH	DPR TYPE
CALIB	IMAGE	LINEARITY,FLAT

For each exposure time value, two difference frames are computed: $dif_1 = on_1 - of_1$ and $dif_2 = on_2 - of_2$. These two difference frames are then averaged.

Each average frame is stored on an imagelist. Then for each x,y pixel intensity I(x,y,z) is computed a polynomial fit of order **order** in direction z.

$$Q(x,y,t) = \sum_{1,\mathbf{order}} c_i(x,y)t^i \tag{16}$$

$$Q^{'}(x,y,t) = \frac{Q(x,y,t)}{c_1(x,y)t} \tag{17}$$

The corresponding coefficients c_i are stored in an imagelist of size **order**. This is a recipe product (LIN_COEF). From each imagelist plane are computed the corresponding medians and standard deviations. Those values are used to identify bad pixels (threshold pixels, trap columns, dead pixels, non linear pixels, noisy pixels). Bad pixels are flagged and stored in a corresponding bad pixel map. The median and standard deviation computation is repeated to remove the contribute of bad pixels. Finally the median and standard deviation of each coefficient plane are monitored as quality control parameters (QC LIN COEFi, QC LIN COEFi ERR).

$$Q(t) = \sum_{1,\mathbf{order}} c_i t^i \tag{18}$$

$$Q^{'}(t) = \frac{Q(t)}{c_1 t} \tag{19}$$

Alternatively the detector linearity may be monitored on the collapsed frame.

For each input frame on-off pair is computed the difference frame: $dif = on - of$. Each frame difference is stored on an imagelist. Then the clean mean/median and standard deviation are computed on a given frame region (**llx, lly, urx, ury**). Those values are computed for each frame difference and stored in a vector. Then computed a polynomial fit of order **order** of vector values.

$$P(t) = \sum_{1,\mathbf{order}} a_i t^i \tag{20}$$

$$P^{'}(t) = \frac{P(t)}{a_1 t} \tag{21}$$

The corresponding coefficients a_i are monitored as quality control parameters (QC LIN COEFi, QC LIN COEFi ERR).

Note that $Q^{'}(t)$ (or $P^{'}(t)$) is independent from the lamp flux.

The correction that can be applied in a later step of the cascade would be:

$$I_{corrected} = I_{original}/Q^{'}(t) \tag{22}$$

Then one build a frame showing the intensity at (t=0) [units=ADU], and divides this by the coefficient c_1 (or a_1) [units ADU/s] to obtain the shutter error frame [units s]. This frame indicate the delay of the shutter in blocking

the light at its entrance. This should be an additional recipe product. Its median value and standard deviation are monitored as QC parameters: QC SHUTMED and QC SHUTSTD

A map of deviating pixels should be also generated. Those pixels are the one which deviate by more than **kappa** sigma by the fit, where sigma is the standard deviation of the points from the fit.

Indeed several bad pixel masks can be obtained: A map of threshold pixels (pixels with intensities beyond a given **threshold**). A map of dead pixels (pixels for which the c_0 - or the corresponding pixel intensity - is below a minimum threshold **threshold_min**). A map of trap pixels (pixels for which c_0 deviates more than **kappa***sigma by the median of c_0). A map on non linear pixels (pixels which deviates more than **kappa*** sigma from the linear behaviour-or pixels whose c_1 deviates more than **kappa** *sigma from the median of $c_1(x, y)$). A map of non quadratic pixels (pixels which deviates more than **kappa*** sigma from the quadratic behaviour-or pixels whose c_2 deviates more than **kappa***sigma from the median of $c_2(x, y)$).

An additional QC parameter is QC LIN EFF, the effective non-linearity correction at a reference user defined level **ref_level**: $Q^{'}$(**ref_level**).

B.3 Detector linearity correction

Assume a set of measurements of $ADU(F_m)$ at different $DIT(t_{exp})$. From this set, we want to determine the linearized flux as a function of the measured flux: $F_r = F_r(F_m)$. Since the detector is non linear a relation of the form $a + b \cdot t_{exp}$ bis not sufficient to describe the behaviour, but may be replaced by a quadratic relation (fit) of the form $a + b \cdot t_{exp} + c \cdot t_{exp}^2$. The real flux F_r is linearly related to t_{exp}. Assuming a constant light source: $F_r = const \cdot t_{exp}$, or $t_{exp} = \frac{F_r}{const}$. At time close to 0, the non linear part can be neglected, thus the slope of F_m and F_r at zero point are the same:

$$\frac{dF_m}{dt_{exp}}(t_{exp} = 0) = \frac{dF_r}{dt_{exp}}$$

But

$$\frac{dF_m}{dt_{exp}}(t_{exp} = 0) = b + 2c \cdot t_{exp} = b$$

$$\frac{dF_r}{dt_{exp}}(t_{exp} = 0) = const$$

Which means that

$$const = b$$

$$\frac{F_r}{b} = t_{exp}$$

$$F_m = a + b \cdot t_{exp} + c \cdot t_{exp}^2 = a + b \cdot (\frac{Fr}{b}) + c \cdot (\frac{F_r}{b})^2 = a + F_r + \frac{c}{b^2} \cdot F_r^2$$

The solution to this quadratic equation in F_r is:

$$Fr^{1,2} = \frac{-b^2 \pm b^2 \cdot \sqrt{1 - 4c(a - Fm)/b^2}}{2c}$$

Obviously, as Fr is positive, the good solution to be used is the "+" one.

This algorithm is general, can be applied to both optical and infrared.

B.4 Detector linearity computation (optical): irplib_detmon_opt_linear

B.5 Pixel to pixel linearity computation

The algorithm is almost identical to irplib_detmon_nir_linear, with the difference that usually one does not have as many biases as flat fields. In that case in computing the differences $dif_1 = on_1 - off_1$ and $dif_2 = on_2 - off_2$. The off frames should be considered either always the same two raw biases in the input list (which set as requirement to have at least two raw biases), or the same master bias (which allows to have in input any number of raw biases or even a reference master bias frame, in which case $off_2 = off_1 = MBIAS$).

B.6 Bad pixel determination

Several are the possible detector pixels which may indicate misbehaviour. Pixels may have a non linear response if exposed by signals beyond a certain level. Those pixels can be detected by the analysis of a set of flat fields of increasing intensities after subtraction of the bias and dark level. There are hot pixels, pixels which have a significantly higher intensity than most of the detector pixels. Those can be found by the analysis of bias frames (normal hot pixels) or flat frames (threshold-hot pixels, pixels whose intensity is above a given threshold). There are also cold pixels, usually detectable on bias frames, as the ones with an intensity significantly lower than the others. Dead pixels are again cold pixels well detectable on flat images as the ones where the signal lies well below the one of adjacent pixels. Noisy pixels are the ones whose intensity lies beyond $\pm$ 5 sigma from the median intensity of a master dark frame. Other bad pixels are the ones due to detector construction defects. Those are usually well detectable on flat field images, as pixels whose intensity deviates from the mean (or median) by more than 5 sigmas. Detector defects are usually pixel aggregates, while threshold and dead pixels may be isolated. The relevant bad pixel kind will be determined on the appropriate frame by comparing the pixel intensity with the median (Master pixel intensity) and the local RMS.

C Installation

This chapter gives generic instructions on how to obtain, build and install the HAWK-I pipeline.

The supported platforms are listed in Section C.1. It is recommended reading through Section C.2.3 before starting the installation.

A bundled version of the HAWK-I pipeline with all the required tools and an installer script is available from [12], for users who are not familiar with the installation of software packages.

C.1 Supported platforms

The utilization of the GNU build tools should allow to build and install the HAWK-I pipeline on a variety of UNIX platforms, but it has only been verified on the following target platforms:

- Scientific Linux 5.3
- Fedora 11

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

C.2 Building the HAWK-I pipeline

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

C.2.1 Software requirements

To compile and install the HAWK-I 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.

For Gasgano support one needs in addition

- the Java Development Kit (version 1.5 or newer)

These packages are part of the supported platform Fedora 11, and a simple way to install all the needed dependencies is executing as root the following command:

```
# yum install java-1.6.0-openjdk python gcc-c++ make gzip tar
```

If you are installing the pipeline using the HAWK-I kit (as recommended), skip the rest of this section and head to C.2.2.

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

Please note that CPL itself depends on an existing cfitsio installation.

In order to run the HAWK-I 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 (see Section **??**, page **??**). 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.

C.2.2 Hardware requirements and performance

Several recipes keep an important amount of frames into memory. If the sof sufficient large it is advised to have more than 3Gb of physical memory. If required to process a large number of frames a 64-bit Linux platform is recommended (see D). Fast disk access is also recommended. If the modular recipes are being used at least 20Gb for intermediate products should be available.

Table C.2.0 gives an idea of the timing and memory consumption when processing an observation with 5 dark frames, 15 flat frames and 15 jitter frames. Both the monolithic and modular recipes are shown. The hardware used was a 2.3 Ghz CPU with 8Gb of RAM and an array of SAS disks (15000 RPM) configured in RAID-5. Your results may vary depending on hardware, number of input files and recipe options used.

Table C.2.0: Hardware resources for several recipes using 5 dark frames, 20 flat frames and 15 science frames. Tested on a machine at 2.3 Ghz, 8Gb RAM and SAS RAID-5 15000 RPM array disks

Recipe	Running time	Max. used memory	Products size
hawki_cal_dark	23 s.	$\approx$ 200 MB	$\approx$ 200 MB
hawki_cal_flat	208 s.	$\approx$ 310 MB	$\approx$ 328 MB
hawki_sci_jitter	465 s.	$\approx$ 555 MB	$\approx$ 160 MB
hawki_step_stats	21 s.	$\approx$ 87 MB	$\approx$ 0 MB
hawki_step_basic_calib	38 s.	$\approx$ 280 MB	$\approx$ 985 MB
hawki_step_compute_bkg	344 s.	$\approx$ 350 MB	$\approx$ 1930 MB
hawki_step_subtract_bkg	15 s.	$\approx$ 130 MB	$\approx$ 985 MB
hawki_step_apply_dist	110 s.	$\approx$ 408 MB	$\approx$ 1100 MB
hawki_step_combine	74 s.	$\approx$ 580 MB	$\approx$ 80 MB
hawki_step_refine_offsets	63 s.	$\approx$ 276 MB	$\approx$ 80 MB
hawki_step_detect_obj	37 s.	$\approx$ 271 MB	$\approx$ 80 MB
hawki_step_stitch	15 s.	$\approx$ 310 MB	$\approx$ 80 MB

C.2.3 Compiling and installing the HAWK-I pipeline

The HAWK-I pipeline distribution kit 1.8.1 contains:

`hawki-pipeline-manual-1.6.pdf`	The HAWK-I pipeline manual
`install_pipeline`	Install script
`cfitsio3090.tar.gz`	CFITSIO 3090
`gsl-1.9.tar.gz`	GSL 1.9
`cpl-5.3.1.tar.gz`	CPL 5.3.1
`esorex-3.9.0.tar.gz`	*EsoRex* 3.9.0
`gasgano-2.4.0.tar.gz`	*Gasgano* 2.4.0
`hawki-1.8.1.tar.gz`	HAWK-I 1.8.1
`hawki-calib-1.8.1.tar.gz`	HAWK-I calibration files 1.8.1

Here is a description of the installation procedure:

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

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

2. Download from the ESO ftp server, `http://www.eso.org/pipelines/`, the latest release of the HAWK-I pipeline distribution.

3. Verify the checksum value of the tar file with the cksum command.

4. Unpack using the following command:

   ```
   gunzip hawki-kit-1.8.1.tar.gz
   tar -xvf hawki-kit-1.8.1.tar
   ```

5. Install: after moving to the top installation directory,

   ```
   cd hawki-kit-1.8.1
   ```

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

   ```
   ./install_pipeline
   ```

   (beware: the execution may take a few minutes on Linux and several minutes on SunOS).

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

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

Finally, the tt $PATH variable should include the path to the *EsoRex* and *Gasgano* utilities.

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

C.2.4 Installing the HAWK-I data demo package

The last version of the data set can be downloaded from:

`ftp://ftp.eso.org/pub/dfs/pipelines/hawki/hawki-demo.tar.bz2`.

To extract the data it is required to have the following tools in the path:

- the `bunzip2` data compression program,
- a version of the `tar` file-archiving program,
- `wget` or other download manager

To unpack the demo package perform the following steps:

1. Change directory to where you want to store the data demo files. You will need at least 15Gb of free space to unpack and later run the pipeline just one time with one data set. However, at least 50Gb is recommended if several executions are done, though.
2. Retrieve the package with `wget` or other download tool:
3. `wget ftp://ftp.eso.org/pub/dfs/pipelines/hawki/hawki-demo.tar.bz2`
4. Unpack it: `bunzip2 hawki-demo.tar.bz2; tar xvf hawki-demo.tar`

C.2.5 Advanced installation

Alternatively, it is possible to perform a manual installation which is recommended only for experienced users. We briefly sketch the steps to carry out a manual installation. We assume that the user is familiar with the autotools software.

1. Untar the pipeline kit and `cd` to `hawki-kit-1.8.1`.
2. For each package found in the installation kit proceed as follows:
   (a) Untar and `cd` to the source directory.
   (b) Found the relevant `--with-xxx` options using `./configure -help`. In particular, for *cpl* the `--with-cfitsio=/path/...` and `--with-wcs=/path/...` are relevant. *EsoRex* and *hawki* will need `--with-cpl/path/...` option.

(c) Run `./configure --prefix=/path/to/install`.

(d) Run `make install`

(e) If the *Doxygen* API documentation is needed (only for developers), then the additional option `--enable-maintainer-mode` has to be set to the `./configure` call. Then `make html` would create the documentation in the `html` directory.

D Known Problems

D.1 Distortion calibration

The distortion files provided by the pipeline were computed during comissioning of the instrument. There is a new project devoted to compute and monitor regularly the distortion of the instrument. The pipeline already provides a recipe to use the autocalibration method described in A, but to date there are regular calibration datasets and extensive testing is still pending.

D.2 Background computation

The pipeline is known to oversubtract the sky near bright and large objects. There has been an improvement lately with the addition of the *--growing_radius* parameter in the *hawki_step_detect_obj recipe*, but still there might be cases where the problem shows up.

D.3 Star trails in flat

D.4 Running out of memory

There are some steps in the pipeline that require to load a large number of frames in memory. Given the large size of the HAWK-I images that means that enough memory should be available for the process running the recipe (either *Gasgano* or *EsoRex*). However if the pipeline is being deployed in a 32-bit environment the amount of available memory for a single process is limited to 3Gb in Linux and 4Gb in Solaris. That means that one can process sof as large as 130 (Linux) or 170 (Solaris) frames in the *hawki_sci_jitter*. Other utilities affected by these limit are *hawki_cal_dark, hawki_step_compute_bkg and hawki_step_combine*.

There message that appears in case of memory exhaustion is similar to this:

```
cpl_xmemory fatal error: malloc(16777216) returned NULL

esorex: cpl_xmemory.c:202: cpl_xmemory_malloc_count:

Assertion 'ptr != ((void *)0)' failed.
```

E Abbreviations and acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CalibDB	Calibration Database
CPL	Common Pipeline Library
HAWK-I	High Acuity Wide field K-band Imager
DFO	Data Flow Operations department
DFS	Data Flow System department
DMD	Data Management and Operations Division
DRS	Data Reduction System
ESO	European Southern Observatory
ESOREX	ESO-Recipe Execution tool
FITS	Flexible Image Transport System
FOV	Field Of View
FPN	Fixed Patter Noise
GUI	Graphical User Interface
OB	Observation Block
OCA	Organization, classification and association rules
QC	Quality Control
RON	Read Out Noise
SOF	Set Of Frames
UT	Unit Telescope
VLT	Very Large Telescope