

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

EsoReflex HAWK-I Tutorial

VLT-MAN-ESO-XXXXX-XXXX

Issue 2.5.20

Date November 14, 2026

Prepared:	UK In-kind Team	November 14, 2026	.....
	Name	Date	Signature
Approved:			.....
	Name	Date	Signature
Released:			.....
	Name	Date	Signature

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
2.2.0	01/04/2017	All	First version
2.2.0	10/04/2017	All	A.Gabasch & M.Neeser: textual updates and edits
2.4.2	06/01/2018	None	Version number aligned
2.4.3	01/06/2018	None	Bugfix in two plotting routines
2.4.6	01/05/2019	None	Version number aligned
2.4.7	01/04/2020	None	Version number aligned
2.4.7	20/04/2020	All	M. Neeser: revision of common Reflex files.
2.4.8	03/06/2020	None	Update of HAWK-I specific sections.
2.4.9	02/05/2021	None	Version number aligned
2.4.10	15/03/2022	None	Version number aligned
2.4.11	04/05/2022	None	Version number aligned
2.4.12	14/03/2023	None	Version number aligned
2.4.13	10/05/2023	None	Version number aligned
2.4.14	14/12/2023	None	Version number aligned
2.4.15	07/07/2024	None	Version number aligned
2.5.7	13/07/2024	None	Version number aligned
2.5.8	09/08/2024	None	Version number aligned
2.5.9	23/09/2024	None	Version number aligned
2.5.10	14/02/2025	None	Version number aligned
2.5.11	27/03/2025	None	Version number aligned
2.5.12	07/04/2025	None	Version number aligned
2.5.13	17/06/2025	None	Version number aligned
2.5.15	09/10/2025	None	Version number aligned
2.5.17	11/12/2025	None	Version number aligned
2.5.19	27/09/2026	None	Version number aligned
2.5.20	14/11/2026	None	Version number aligned

This page was intentionally left blank

Contents

1 Introduction to `Esoreflex` 7

2 Introduction And Scope 8

3 System Requirements 9

4 Software Installation 10

4.1 Installing `Esoreflex` workflows via `macports` . . . 10

4.2 Installing `Esoreflex` workflows via `rpm/yum/dnf` . . . 10

4.3 Installing `Esoreflex` workflows via `install_esoreflex` . . . 10

5 HAWK-I Demo Data Set 12

6 Quick Start: Reducing The Demo Data 13

7 About the main `esoreflex` canvas 17

7.1 Saving And Loading Workflows . . . 17

7.2 Buttons . . . 17

7.3 Workflow States . . . 17

8 The HAWK-I Workflow 18

8.1 Workflow Canvas Parameters . . . 18

8.2 HAWK-I Specific Workflow Canvas Parameters . . . 19

8.2.1 Simple Actors . . . 20

8.2.2 Composite Actors . . . 20

8.2.3 Recipe Execution within Composite Actors . . . 22

8.2.4 Lazy Mode . . . 24

8.3 Workflow Steps . . . 24

8.3.1 Data Organisation And Selection . . . 24

8.3.2 DataSetChooser . . . 25

8.3.3 The ProductExplorer . . . 26

8.4 Workflow Details . . . 28

8.4.1 Data Organisation and Selection Actors . . . 28
8.4.2 Editing Recipe Parameters . . . 29
8.4.3 Master Calibration Actors . . . 30
8.4.4 Standard Star and Science Fields Actors . . . 30
8.4.5 Output Organisation . . . 30
8.5 Interactive Windows . . . 31
8.5.1 Master Dark . . . 31
8.5.2 Master Flat . . . 33
8.5.3 Master Readgain . . . 35
8.5.4 Standard Stars . . . 37
8.5.5 Science Fields . . . 39
8.5.6 Science Postprocess . . . 44
8.6 Improving Results through Workflow Interaction . . . 45

9 Frequently Asked Questions **47**

1 Introduction to `Esoreflex`

This document is a tutorial designed to enable the user to to reduce his/her data with the ESO pipeline run under an user-friendly environment, called `EsoReflex`, concentrating on high-level issues such as data reduction quality and signal-to-noise (S/N) optimisation.

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

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

A workflow accepts science and calibration data, as downloaded from the archive using the CalSelector tool[1] (with associated raw calibrations) and organises them into DataSets, where each DataSet contains one science object observation (possibly consisting of several science files) and all associated raw and static calibrations required for a successful data reduction. The data organisation process is fully automatic, which is a major time-saving feature provided by the software. The DataSets selected by the user for reduction are fed to the workflow which executes the relevant pipeline recipes (or stages) in the correct order. Full control of the various recipe parameters is available within the workflow, and the workflow deals automatically with optional recipe inputs via built-in conditional branches. Additionally, the workflow stores the reduced final data products in a logically organised directory structure employing user-configurable file names.

[1] *https://www.eso.org/sci/archive/calselectorInfo.html*

2 Introduction And Scope

This document is a tutorial designed to enable the user to employ the HAWK-I workflow to reduce their data in a user-friendly way, concentrating on high-level issues such as data reduction quality and signal-to-noise (S/N) optimisation.

A workflow accepts science and calibration data, as delivered to PIs in the form of PI-Packs (until October 2011) or downloaded from the archive using the CalSelector tool[2] and organises them into DataSets. A DataSet typically contains one set of science object observations (possibly consisting of several science files) and all associated raw and static calibrations required for a successful data reduction. The data organisation process follows a particular set of rules and is fully automatic. This can save a considerable amount of users's time. However, it comes with a cost: the data may be organised into DataSets that are contrary to a users' particular needs or expectations. Users are **very strongly advised** to check the data organisation before proceeding onto the data reduction. The DataSets selected by the user for reduction are fed to the workflow which executes the relevant pipeline recipes in the correct order.

The HAWK-I workflow provides optional user interactivity for most steps of the data reduction process. This interactivity enables a user to assess the pipeline products and/or re-run a recipe with different parameters before moving to the next step. Full control of all recipe parameters is available within the workflow, and the workflow deals automatically with optional recipe inputs via built-in conditional branches. Additionally, the workflow stores the reduced final data products in a logically organised directory structure employing user-configurable filenames.

The HAWK-I `Reflex` workflow described in this tutorial supports the reduction of HAWK-I images taken in all operational modes **except** for the *BURST* or *Fast Jitter* mode. The user is referred to the HAWK-I User Manual[3] for more more information about the instrument and the pipeline.

By default, the workflow will group data together by the start time of the **Observation Block** in which it was taken (`OBS.START`). This means that the workflow cannot be used to combine or stack images that were taken as part of OBs with different start times. As discussed in the HAWK-I Pipeline manual [3], the pipeline is not designed to process science data or standard star fields that span more than one OB.

The HAWK-I pipeline can provide high quality data products using the default parameters for most recipes. These include calibrated sky and science images, mosaics, and catalogues of objects in the images. The user is encouraged to adjust and experiment with the recipe parameters in order to achieve the best results for his/her data.

The quick start section (§6) describes the minimum effort to get started and reduce the demonstration data.

[2] http://www.eso.org/sci/archive/calselectorInfo.html
[3] http://www.eso.org/sci/facilities/paranal/instruments/hawki/doc.html

3 System Requirements

Due to the large number of files with short exposures, data reduction for infrared imaging may use a significant amount of memory. This can be exacerbated when the images are combined into a tile that covers a large part of the sky. The amount of memory required to run a recipe depends on a number of factors, e.g. the sky subtraction scheme, number of files, and sky coverage. Below is a table with the minimum resident memory needed to process data as a function of recipe and number of input science frames. This table assumes that all recipe parameters are set to their default and a small maximum jitter offset between frames. In general, the execution time of a recipe can be shortened by choosing 'fast' stacking, but this will increase the memory requirements. The reason that `hawki_science_process` with 20 files uses more memory than with 50 files is because (using default recipe parameters) the stacking method changes from 'fast' to 'slow' (see description of `stk_fast` and `stk_nfst` parameters in [3]).

Recipe	**# science frames**	**Min. RAM**
hawki_standard_process	4	3.2 GB
hawki_science_process	5	6.6 GB
hawki_science_process	10	10.5 GB
hawki_science_process	20	17.3 GB
hawki_science_process	50	7.9 GB

Table 3.1: Minimum memory requirements for selected HAWK-I recipes

4 Software Installation

`Esoreflex` and the workflows can be installed in different ways: via package repositories, via the `install_esoreflex` script or manually installing the software tar files.

The recommended way is to use the package repositories if your operating system is supported. The pipelines and Reflex can be installed from the ESO `macports` repositories that support macOS platforms, the and the `rpm/yum` repositories that support Fedora and CentOS platforms. For any other operating system it is recommended to use the `install_esoreflex` script.

The installation from package repository requires administrative privileges (typically granted via sudo), as it installs files in system-wide directories under the control of the package manager. If you want a local installation, or you do not have sudo privileges, or if you want to manage different installations on different directories, then use the `install_esoreflex` script. Note that the script installation requires that your system fulfill several software prerequisites, which might also need sudo privileges.

Please note that in case of major or minor (affecting the first two digit numbers) Reflex upgrades, the user should erase the `$HOME/KeplerData`, `$HOME/.kepler` directories if present, to prevent possible aborts (i.e. a hard crash) of the `esoreflex` process.

4.1 Installing `Esoreflex` workflows via `macports`

This method is supported for the macOS operating system. It is assumed that macports (*https://www.macports.org*) is installed. Please read the full documentation at *https://www.eso.org/sci/software/pipelines/installation/macports.html*, which also describes the versions of macOS that are currently supported.

4.2 Installing `Esoreflex` workflows via `rpm/yum/dnf`

This method is supported for Fedora and CentOS platforms and requires sudo rights. Please read the full documentation at *https://www.eso.org/sci/software/pipelines/installation/rpm.html*, which also describes the versions of Fedora and CentOS that are currently supported.

4.3 Installing `Esoreflex` workflows via `install_esoreflex`

This method is recommended for operating systems other than those indicated above, or if the user has no sudo rights. Software dependencies are not fulfilled by the installation script, therefore the user has to install all the prerequisites before running the installation script.

The software pre-requisites for `Reflex 2.9` may be found at:
https://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.9` software and demo data, please follow these instructions:

1. From any directory, download the installation script:

```
wget https://eso.org/sci/software/pipelines/install_esoreflex
```

2. Make the installation script executable:

```
chmod u+x install_esoreflex
```

3. Execute the installation script:

```
./install_esoreflex
```

and the script will ask you to specify three directories: the download directory <download_dir>, the software installation directory <install_dir>, and the directory to be used to store the demo data <data_dir>. If you do not specify these directories, then the installation script will create them in the current directory with default names.

4. Follow all the script instructions; you will be asked whether to use your Internet connection (recommended: yes), the pipelines and demo-datasets to install (note that the installation will remove all previously installed pipelines that are found in the same installation directory).

5. To start Reflex, issue the command:

```
<install_dir>/bin/esoreflex
```

It may also be desirable to set up an alias command for starting the Reflex software, using the shell command alias. Alternatively, the PATH variable can be updated to contain the <install_dir>/bin directory.

5 HAWK-I Demo Data Set

The HAWK-I pipeline kit comes with a set of demonstration data. These data are intended to be used as a means to become familiar with the pipeline and workflow. They were selected to show the typical steps to reduce most HAWK-I data sets. Data taken with an uncommon or complex observing strategy may not be suitable for reduction with the `Reflex` workflow. In this case, a deeper understanding of the pipeline may be required and a user may have to use `EsoRex` to process their data.

The full collection of demo data comprise raw data, static calibration data, and photometric and astrometric catalogues. Three complete data sets can be formed from the demo data: one standard star field ('FS232'), and two sets of science observations toward the object 'BDF4'. The data in each data set can be used to create a master dark image, master twilight flat, and (optionally) a table of the detector readnoise and gain. A full description of the demo data is beyond the scope of this document and can be found in the HAWK-I Pipeline Manual [3].

The HAWKI pipeline can use a number of optional files to calibrate the astrometry and photometry of processed images. **These files are only necessary if a user wishes to process data without an internet connection.** If a user has an internet connection, the HAWKI recipes can retrieve the required data automatically through the Strasbourg astronomical Data Center[4] (CDS). It is recommended that HAWK-I data use the 2MASS point source catalogue [2] for photometric calibration, and the WISE [1] catalogue for astrometric calibration.

[4] `http://cdsweb.u-strasbg.fr/`

6 Quick Start: Reducing The Demo Data

For the user who is keen on starting reductions without being distracted by detailed documentation, we describe the steps to be performed to reduce the science data provided in the HAWK-I demo data set supplied with the `esoreflex 2.9` release. By following these steps, the user should have enough information to perform a reduction of his/her own data without any further reading:

1. First, type:

   ```
   esoreflex -l
   ```

   If the `esoreflex` executable is not in your path, then you have to provide the command with the executable full path `<install_dir>/bin/esoreflex -l` . For convenience, we will drop the reference to `<install_dir>`. A list with the available `esoreflex` workflows will appear, showing the workflow names and their full path.

2. Open the HAWK-I by typing:

   ```
   esoreflex hawki&
   ```

   Alternatively, you can type only the command `esoreflex` the empty canvas will appear (Figure 6.1) and you can select the workflow to open by clicking on `File -> Open File`. Note that the loaded workflow will appear in a new window. The HAWK-I workflow is shown in Figure 6.2.

3. To aid in the visual tracking of the reduction cascade, it is advisable to use component (or actor) highlighting. Click on `Tools -> Animate at Runtime`, enter the number of milliseconds representing the animation interval (100 ms is recommended), and click `OK`.

4. Change directories set-up. Under "Setup Directories" in the workflow canvas there are seven parameters that specify important directories (green dots).

   By default, the `ROOT_DATA_DIR`, which specifies the working directory within which the other directories are organised. is set to your `$HOME/reflex_data` directory. All the temporary and final products of the reduction will be organized under sub-directories of `ROOT_DATA_DIR`, therefore make sure this parameter points to a location where there is enough disk space. To change `ROOT_DATA_DIR`, double click on it and a pop-up window will appear allowing you to modify the directory string, which you may either edit directly, or use the `Browse` button to select the directory from a file browser. When you have finished, click `OK` to save your changes.

   Changing the value of `RAW_DATA_DIR` is the only necessary modification if you want to process data other than the demo data

5. Click the  button to start the workflow

6. The workflow will highlight the `Data Organiser` actor which recursively scans the raw data directory (specified by the parameter `RAW_DATA_DIR` under "Setup Directories" in the workflow canvas) and constructs the datasets. Note that the raw and static calibration data must be present either

in `RAW_DATA_DIR` or in `CALIB_DATA_DIR`, otherwise datasets may be incomplete and cannot be processed. However, if the same reference file was downloaded twice to different places this creates a problem as `esoreflex` cannot decide which one to use.

7. The `Data Set Chooser` actor will be highlighted next and will display a "Select Datasets" window (see Figure 6.3) that lists the datasets along with the values of a selection of useful header keywords[5]. The first column consists of a set of tick boxes which allow the user to select the datasets to be processed. By default all complete datasets which have not yet been reduced will be selected. A full description of the options offered by the `Data Set Chooser` will be presented in Section 8.3.2.

8. Click the `Continue` button and watch the progress of the workflow by following the red highlighting of the actors. A window will show which dataset is currently being processed.

9. Once the reduction of all datasets has finished, a pop-up window called *Product Explorer* will appear, showing the datasets which have been reduced together with the list of final products. This actor allows the user to inspect the final data products, as well as to search and inspect the input data used to create any of the products of the workflow. Figure 6.4 shows the Product Explorer window. A full description of the *Product Explorer* will be presented in Section 8.3.3.

10. After the workflow has finished, all the products from all the datasets can be found in a directory under `END_PRODUCTS_DIR` named after the workflow start timestamp. Further subdirectories will be found with the name of each dataset.

Well done! You have successfully completed the quick start section and you should be able to use this knowledge to reduce your own data. However, there are many interesting features of `Reflex` and the HAWK-I workflow that merit a look at the rest of this tutorial.

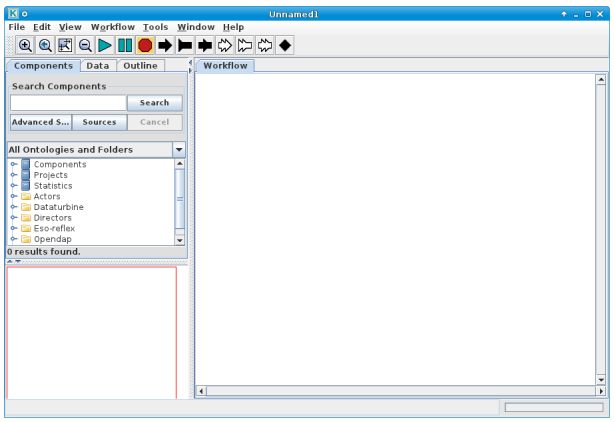


Figure 6.1: *The empty* `Reflex` *canvas.*

[5]The keywords listed can be changed by double clicking on the `DataOrganiser` Actor and editing the list of keywords in the second line of the pop-up window. Alternatively, instead of double-clicking, you can press the right mouse button on the `DataOrganiser` Actor and select `Configure Actor` to visualize the pop-up window.

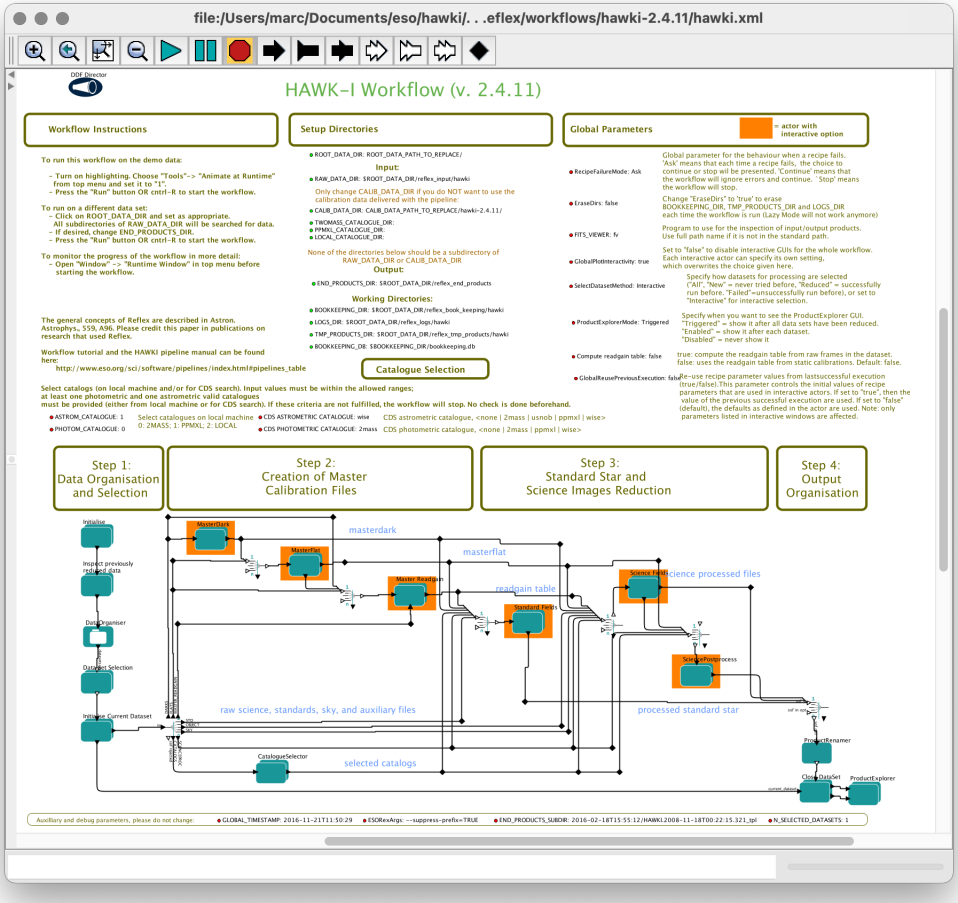


Figure 6.2: The HAWK-I workflow general layout.

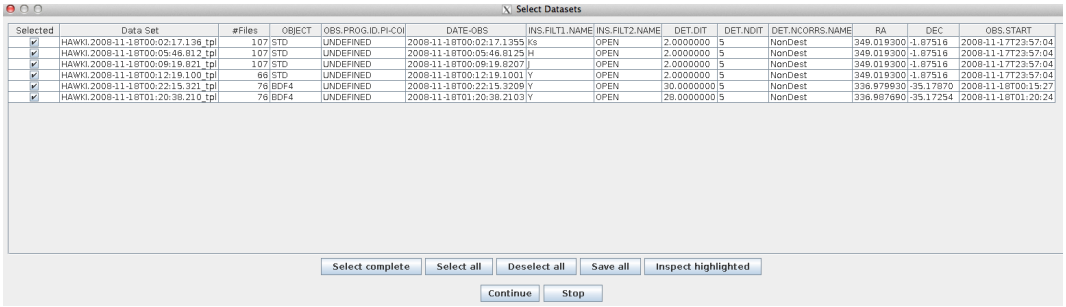


Figure 6.3: *The "Select Datasets" pop-up window.*

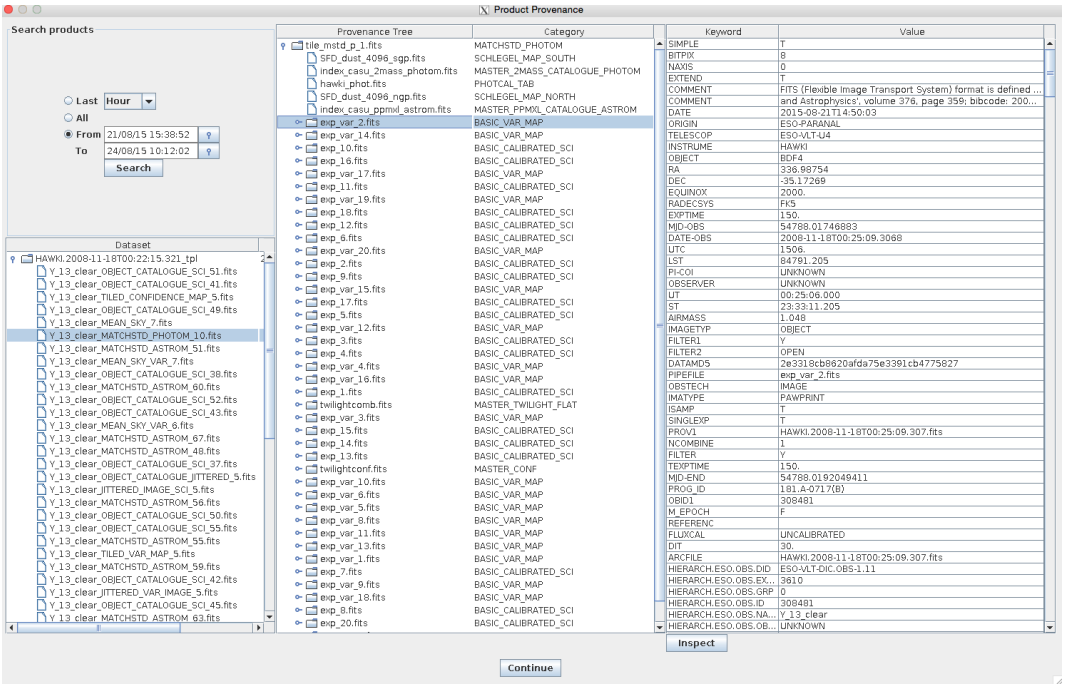


Figure 6.4: *The Product Explorer shows all datasets reduced in previous executions together with the full reduction chain for all the pipeline products..*

7 About the main `esoreflex` canvas

7.1 Saving And Loading Workflows

In the course of your data reductions, it is likely that you will customise the workflow for various data sets, even if this simply consists of editing the `ROOT_DATA_DIR` to a different value for each data set. Whenever you modify a workflow in any way, you have the option of saving the modified version to an `XML` file using `File -> Export As` (which will also open a new workflow canvas corresponding to the saved file). The saved workflow may be opened in subsequent `esoreflex` sessions using `File -> Open`. Saving the workflow in the default Kepler format (.kar) is only advised if you do not plan to use the workflow with another computer.

7.2 Buttons

At the top of the `esoreflex` canvas are a set of buttons which have the following functions:

-  - Zoom in.
-  - Reset the zoom to 100%.
-  - Zoom the workflow to fit the current window size (Recommended).
-  - Zoom out.
-  - Run (or resume) the workflow.
-  - Pause the workflow execution.
-  - Stop the workflow execution.

The remainder of the buttons (not shown here) are not relevant to the workflow execution.

7.3 Workflow States

A workflow may only be in one of three states: executing, paused, or stopped. These states are indicated by the yellow highlighting of the , , and  buttons, respectively. A workflow is executed by clicking the  button. Subsequently the workflow and any running pipeline recipe may be stopped immediately by clicking the  button, or the workflow may be paused by clicking the  button which will allow the current actor/recipe to finish execution before the workflow is actually paused. After pausing, the workflow may be resumed by clicking the  button again.

8 The HAWK-I Workflow

The HAWK-I workflow canvas is organised into a number of areas. From top-left to top-right you will find general workflow instructions, directory parameters, and global parameters. In the middle row you will find five boxes describing the workflow general processing steps in order from left to right, and below this the workflow actors themselves are organised following the workflow general steps.

8.1 Workflow Canvas Parameters

The workflow canvas displays a number of parameters that may be set by the user. Under "Setup Directories" the user is only required to set the `RAW_DATA_DIR` to the working directory for the dataset(s) to be reduced, which, by default, is set to the directory containing the demo data. The `RAW_DATA_DIR` is recursively scanned by the `Data Organiser` actor for input raw data. The directory `CALIB_DATA_DIR`, which is by default within the pipeline installation directory, is also scanned by the `Data Organiser` actor to find any static calibrations that may be missing in your dataset(s). If required, the user may edit the directories `BOOKKEEPING_DIR`, `LOGS_DIR`, `TMP_PRODUCTS_DIR`, and `END_PRODUCTS_DIR`, which correspond to the directories where book-keeping files, logs, temporary products and end products are stored, respectively (see the Reflex User Manual for further details; [4]).

There is a mode of the `Data Organiser` that skips the built-in data organisation and uses instead the data organisation provided by the CalSelector tool. To use this mode, click on `Use CalSelector associations` in the `Data Organiser` properties and make sure that the input data directory contains the XML file downloaded with the CalSelector archive request (note that this does not work for all instrument workflows).

Under the "Global Parameters" area of the workflow canvas, the user may set the `FITS_VIEWER` parameter to the command used for running his/her favourite application for inspecting FITS files. Currently this is set by default to `fv`, but other applications, such as `ds9`, `skycat` and `gaia` for example, may be useful for inspecting image data. Note that it is recommended to specify the full path to the visualization application (an alias will not work).

By default the `EraseDirs` parameter is set to `false`, which means that no directories are cleaned before executing the workflow, and the recipe actors will work in Lazy Mode (see Section 8.2.4), reusing the previous pipeline recipe outputs if input files and parameters are the same as for the previous execution, which saves considerable processing time. Sometimes it is desirable to set the `EraseDirs` parameter to `true`, which forces the workflow to recursively delete the contents of the directories specified by `BOOKKEEPING_DIR`, `LOGS_DIR`, and `TMP_PRODUCTS_DIR`. This is useful for keeping disk space usage to a minimum and will force the workflow to fully re-reduce the data each time the workflow is run.

The parameter `RecipeFailureMode` controls the behaviour in case that a recipe fails. If set to `Continue`, the workflow will trigger the next recipes as usual, but without the output of the failing recipe, which in most of the cases will lead to further failures of other recipes without the user actually being aware of it. This mode might be useful for unattended processing of large number of datasets. If set to `Ask`, a pop-up window will ask whether the workflow should stop or continue. This is the default. Alternatively, the `Stop` mode will stop the workflow execution immediately.

The parameter `ProductExplorerMode` controls whether the `ProductExplorer` actor will show its window or not. The possible values are `Enabled`, `Triggered`, and `Disabled`. `Enabled` opens the Product-

Explorer GUI at the end of the reduction of each individual dataset. `Triggered` (default and recommended) opens the ProductExplorer GUI when all the selected datasets have been reduced. `Disabled` does not display the ProductExplorer GUI.

8.2 HAWK-I Specific Workflow Canvas Parameters

The workflow may use index files and catalogue files for astrometric and photometric calibration. See the data description section in [3] for details. If a user chooses not to use CDS to retrieve catalogue data over the Internet, these files should be placed in the `TWOMASS_CATALOGUE_DIR` and `PPMXL_CATALOGUE_DIR`, respectively. If a user would like to use their own *custom* catalogue for calibration, the catalogue should be placed in the `LOCAL_CATALOGUE_DIR` directory; see the HAWK-I Pipeline Manual[3] for details on how to construct and use such a catalogue. If a user would rather use CDS for astrometric and photometric calibration, these directories should be set to an empty string, e.g. "".

There a number of parameters that can be set under the "Global Parameters" area of the workflow canvas:

`Compute readgain table`: If set to `false`, the default values for the HAWK-I detector readnoise and gain (provided by a static calibration file) will be used. It set to `true`, the workflow will attempt to identify the appropriate raw data files and will process them in the `Master Readgain` actor. If the recipe is successful, the values derived from the input raw data will be used in subsequent actors.

and under the "Catalogue Selection" area of the workflow canvas:

`ASTROM_CATALOGUE`: The default behaviour is to use the CDS to retrieve the astrometric catalogue online. If a user does not use CDS to retrieve an astrometric catalogue online, this parameter is used to identify the catalogue that should be used. Accepted values are 0 for 2MASS, 1 for PPMXL (default), and 2 for a LOCAL catalogue. Note that index files and the catalogues themselves must be present in one of the Setup Directories.

`PHOTOM_CATALOGUE`: The default behaviour is to use the CDS to retrieve the photometric catalogue online. If a user does not use CDS to retrieve a photometric catalogue online, this parameter is used to identify the catalogue that should be used. Accepted values are 0 for 2MASS (default), 1 for PPMXL, and 2 for a LOCAL catalogue. Note that index files and the catalogues themselves must be present in one of the Setup Directories.

`CDS ASTROMETRIC CATALOGUE`: If a user wishes to use CDS to retrieve an astrometric catalogue online, this parameter is used to identify the catalogue that should be used. Accepted values are 2mass, usnob, ppmxl, wise (default), and none. If 'none' is selected, the data will be calibrated using the catalogue specified in `ASTROM_CATALOGUE`.

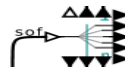
`CDS PHOTOMETRIC CATALOGUE`: If a user wishes to use CDS to retrieve an photometric catalogue online, this parameter is used to identify the catalogue that should be used. Accepted values are 2mass (default), ppmxl, wise, and none. If 'none' is selected, the data will be calibrated using the catalogue specified in `PHOTOM_CATALOGUE`.

8.2.1 Simple Actors

Simple actors have workflow symbols that consist of a single (rather than multiple) green-blue rectangle. They may also have an icon within the rectangle to aid in their identification. The following actors are simple actors:

- 

- The `DataOrganiser` actor.
- 

- The `DataSetChooser` actor (inside a composite actor).
- 

- The `FitsRouter` actor Redirects files according to their categories.
- 

- The `ProductRenamer` actor.
- 

- The `ProductExplorer` actor (inside a composite actor).

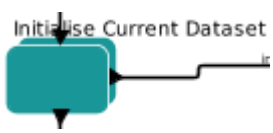
Access to the parameters for a simple actor is achieved by right-clicking on the actor and selecting `Configure Actor`. This will open an "Edit parameters" window. Note that the `Product Renamer` actor is a jython script (Java implementation of the Python interpreter) meant to be customised by the user (by double-clicking on it).

8.2.2 Composite Actors

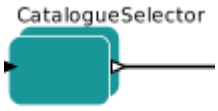
The following is a list of composite actors and a very brief description:

- 

- The `Initialise` actor will execute some preliminary steps: erase directories if requested, and assign the timestamp and Setup Directories.
- 

- The `DataSet Selection` actor constructs data sets and selects which one(s) to process.
- 

- The `Initialise Current Dataset` executes some preliminary steps required to process a particular dataset, e.g. create directories, show a window with the processing status, keep track of which data set this is, etc.

- 

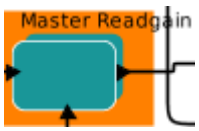
- The `CatalogueSelector` actor selects which photometric and astrometric catalogue to use when processing a data set; this is determined by the values of `ASTROM_CATALOGUE` and `PHOTOM_CATALOGUE` parameters on the main canvas.

- 

- The `MasterDark` actor executes the `hawki_dark_combine` recipe and may launch an interactive window displaying the results.

- 

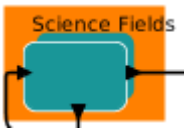
- The `MasterFlat` actor executes the `hawki_twilight_flat_combine` recipe and may launch an interactive window displaying the results.

- 

- The `MasterReadgain` actor executes the `hawki_detector_noise` recipe (if requested) and may launch an interactive window displaying the results.

- 

- The `Standard Fields` actor executes the `hawki_standard_process` recipe and may launch an interactive window displaying the results.

- 

- The `Science Fields` actor executes the `hawki_science_process` recipe and may launch an interactive window displaying the results.

- 

- The `SciencePostprocess` actor executes the `hawki_science_postprocess` recipe and may launch an interactive window displaying the results.

- 

- The `Close Dataset` actor creates a README file with the list of final product files, displays a window with the processed data set, and organises data for the `ProductExplorer` actor.

8.2.3 Recipe Execution within Composite Actors

The composite actors that run the HAWK-I recipes contain a number of components. With the exception of MasterReadgain, the actors generally have the same basic structure. Only a brief description of the structure is described here. If a user would like more information or would like to alter the functionality of these actors, please visit *https://support.eso.org*.

The actors begin by splitting the input files into those that share the same purpose; this is done by the `SofSplitter` actor. The appropriate files and recipe parameters are converted into a SoF file (eventually converted into a set of parameters, a.k.a. SOP) and recipe configuration file, respectively. These are then fed into a `RecipeLooper`. The looper starts by feeding the two files into the recipe executer actor (the actor named after the recipe). The output of the recipe and an interactivity flag are sent to a `PythonActor`. The `PythonActor` runs a Python script that inspects the output, and if requested, launches an interactive window. This window enables a user to examine the recipe products and to alter any recipe parameters. If a user selects "Re-run Recipe" in the interactive window, then the whole process starts again with the new recipe parameters. If a user selects "Continue" in the interactive window, or if interactivity is disabled, then some diagnostic data files might be thrown away, and the resultant data products are accumulated by the `SofAccumulator`. A new SoF file containing the recipe products is then created and sent to an output port. The interactivity is set or restored according to the user's preference. Note that for some actors, there are additional `Fits Router` actors, boolean switches, etc. that are used to handle missing or incomplete data. Figure 8.1 shows the inside of the MasterFlat actor.

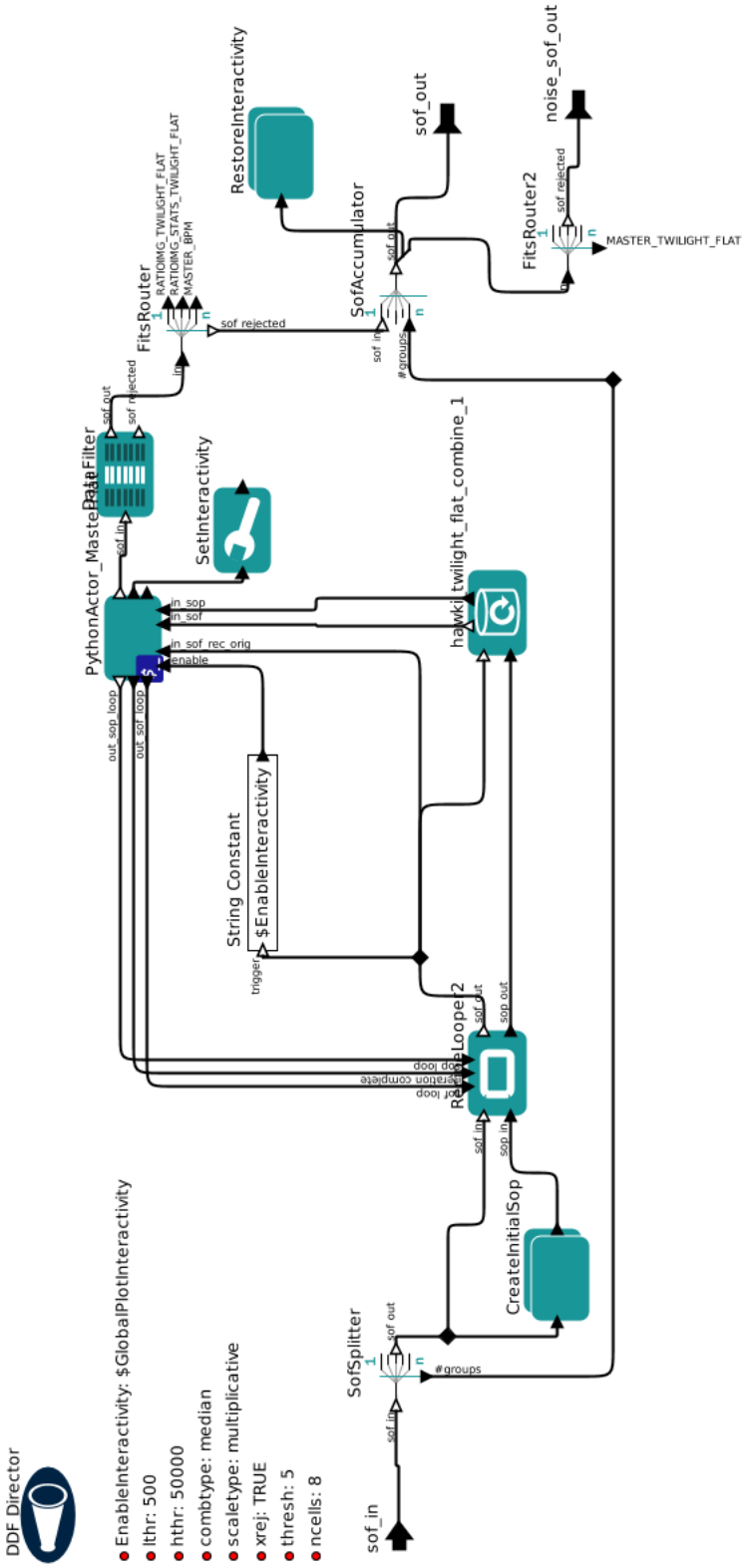


Figure 8.1: The internal actors within the MasterFlat actor.

8.2.4 Lazy Mode

By default, all `RecipeExecuter` actors in a pipeline workflow are "Lazy Mode" enabled. This means that when the workflow attempts to execute such an actor, the actor will check whether the relevant pipeline recipe has already been executed with the same input files and with the same recipe parameters. If this is the case, then the actor will not execute the pipeline recipe, and instead it will simply broadcast the previously generated products to the output port. The purpose of the Lazy Mode is therefore to minimise any reprocessing of data by avoiding data re-reduction where it is not necessary.

One should note that the actor's Lazy Mode depends on the contents of the directory specified by the parameter `BOOKKEEPING_DIR` and the relevant FITS file checksums. Any modification to the directory contents and/or the file checksums will cause the corresponding actor to run the pipeline recipe again when executed, thereby re-reducing the input data.

The re-reduction of data at each execution may sometimes be desirable. To force a re-reduction of data for any single `RecipeExecuter` actor in the workflow, right-click the actor, select `Configure Actor`, and uncheck the Lazy mode parameter tick-box in the "Edit parameters" window that is displayed. For many workflows the `RecipeExecuter` actors are actually found inside the composite actors in the top level workflow. To access such embedded `RecipeExecuter` actors you will first need to open the sub-workflow by right-clicking on the composite actor and then selecting `Open Actor`.

To force the re-reduction of all data in a workflow (i.e. to disable Lazy mode for the whole workflow), you must uncheck the Lazy mode for every single `RecipeExecuter` actor in the entire workflow. It is also possible to change the name of the bookkeeping directory, instead of modifying any of the Lazy mode parameters. This will also force a re-reduction of the given dataset(s). A new reduction will start (with the lazy mode still enabled), but the results of previous reduction will not be reused. Alternatively, if there is no need to keep any of the previously reduced data, one can simply set the `EraseDirs` parameter under the "Global Parameters" area of the workflow canvas to `true`. This will then remove all previous results that are stored in the bookkeeping, temporary, and log directories before processing the input data, in effect, starting a new clean data reduction and re-processing every input dataset. *Note: The option* `EraseDirs = true` *does not work in* `esoreflex` *version 2.9.x and causes the workflow to crash.*

8.3 Workflow Steps

8.3.1 Data Organisation And Selection

The `DataOrganiser` (DO) is the first crucial component of a Reflex workflow. The DO takes as input `RAW_DATA_DIR` and `CALIB_DATA_DIR` and it detects, classifies, and organises the files in these directories and any subdirectories. The output of the DO is a list of "DataSets". A DataSet is a special Set of Files (SoF). A DataSet contains one or several science (or calibration) files that should be processed together, and all files needed to process these data. This includes any calibration files, and in turn files that are needed to process these calibrations. Note that different DataSets might overlap, i.e. some files might be included in more than one DataSet (e.g., common calibration files).

A DataSet lists three different pieces of information for each of its files, namely 1) the file name (including the path), 2) the file category, and 3) a string that is called the "purpose" of the file. The DO uses the

OCA[6] rules to find the files to include in a DataSet, as well as their categories and purposes. The file category identifies different types of files, and it is derived by information in the header of the file itself. A category could for example be RAW_CALIBRATION_1, RAW_CALIBRATION_2 or RAW_SCIENCE, depending on the instrument. The purpose of a file identifies the reason why a file is included in a DataSet. The syntax is action_1/action_2/action_3/ ... /action_n, where each action_i describes an intended processing step for this file (for example, creation of a MASTER_CALIBRATION_1 or a MASTER_CALIBRATION_2). The actions are defined in the OCA rules and contain the recipe together with all file categories required to execute it (and predicted products in case of calibration data). For example, a workflow might include two actions action_1 and action_2. The former creates MASTER_CALIBRATION_1 from RAW_CALIBRATION_1, and the later creates a MASTER_CALIBRATION_2 from RAW_CALIBRATION_2. The action_2 action needs RAW_CALIBRATION_2 frames and the MASTER_CALIBRATION_1 as input. In this case, these RAW_CALIBRATION_1 files will have the purpose action_1/action_2. The same DataSet might also include RAW_CALIBRATION_1 with a different purpose; irrespective of their purpose the file category for all these biases will be RAW_CALIBRATION_1.

The Datasets created via the DataOrganiser will be displayed in the DataSet Chooser. Here the users have the possibility to inspect the various datasets and decide which one to reduce. By default, DataSets that have not been reduced before are highlighted for reduction. Click either Continue in order to continue with the workflow reduction, or Stop in order to stop the workflow. A full description of the DataSet Chooser is presented in Section 8.3.2.

Once the Continue is pressed, the workflow starts to reduce the first selected DataSet. Files are broadcasted according to their purpose to the relevant actors for processing.

The categories and purposes of raw files are set by the DO, whereas the categories and purpose of products generated by recipes are set by the RecipeExecuter. The file categories are used by the FitsRouter to send files to particular processing steps or branches of the workflow (see below). The purpose is used by the SofSplitter and SofAccumulator to generate input SoFs for the RecipeExecuter. The SofSplitter and SofAccumulator accept several SoFs as simultaneous input. The SofAccumulator creates a single output SoF from the inputs, whereas the SofSplitter creates a separate output SoF for each purpose.

8.3.2 DataSetChooser

The DataSetChooser displays the DataSets available in the "Select Data Sets" window, activating vertical and horizontal scroll bars if necessary (Fig. 6.3).

Some properties of the DataSets are displayed: the name, the number of files, a flag indicating if it has been successfully reduced (a green OK), if the reduction attempts have failed or were aborted (a red FAILED), or if it is a new dataset (a black "-"). The column "Descriptions" lists user-provided descriptions (see below), other

[6]OCA stands for OrganisationClassificationAssociation and refers to rules, which allow to classify the raw data according to the contents of the header keywords, organise them in appropriate groups for processing, and associate the required calibration data for processing. They can be found in the directory <install_dir>/share/esopipes/<pipeline-version>/reflex/, carrying the extension .oca. The variable <install_dir> depends on the operative system and installation procedure. For installation through rpm: <install_dir>=/usr; for installation through macport <install_dir>=/opt/local; for installation through the installation script install_esoreflex it depends on the path specified during installation, e.g. <install_dir>=<specified_path>/install

columns indicate the instrument set-up and a link to the night log.

Sometimes you will want to reduce a subset of these DataSets rather than all DataSets, and for this you may individually select (or de-select) DataSets for processing using the tick boxes in the first column, and the buttons `Deselect All` and `Select Complete` at the bottom, or configure the "Filter" field at the bottom left. Available filter options are: "New" (datasets not previously reduced will be selected), "Reduced" (datasets previously reduced will be selected), "All" (all datasets will be selected), and "Failed" (dataset with a failed or aborted reduction will be selected).

You may also highlight a single DataSet in blue by clicking on the relevant line. If you subsequently click on `Inspect Highlighted`, then a "Select Frames" window will appear that lists the set of files that make up the highlighted DataSet including the full filename[7], the file category (derived from the FITS header), and a selection tick box in the right column. The tick boxes allow you to edit the set of files in the DataSet which is useful if it is known that a certain calibration frame is of poor quality (e.g: a poor raw flat-field frame). The list of files in the DataSet may also be saved to disk as an `ASCII` file by clicking on `Save As` and using the file browser that appears.

By clicking on the line corresponding to a particular file in the "Select Frames" window, the file will be highlighted in blue, and the file FITS header will be displayed in the text box on the right, allowing a quick inspection of useful header keywords. If you then click on `Inspect`, the workflow will open the file in the selected FITS viewer application defined by the workflow parameter `FITS_VIEWER`.

To exit from the "Select Frames" window, click `Continue`.

To add a description of the reduction, press the button `...` associated with the field "Add description to the current execution of the workflow" at the bottom right of the Select Dataset Window; a pop-up window will appear. Enter the desired description (e.g. "My first reduction attempt") and then press `OK`. In this way, all the datasets reduced in this execution, will be flagged with the input description. Description flags can be visualized in the `SelectFrames` window and in the `ProductExplorer`, and they can be used to identify different reduction strategies.

To exit from the "Select DataSets" window, click either `Continue` in order to continue with the workflow reduction, or `Stop` in order to stop the workflow.

8.3.3 The ProductExplorer

The ProductExplorer is an interactive component in the `esoreflex` workflow whose main purpose is to list the final products with the associated reduction tree for each dataset and for each reduction attempt (see Fig. 6.4).

Configuring the ProductExplorer

You can configure the ProductExplorer GUI to appear after or before the data reduction. In the latter case you can inspect products as reduction goes on.

1. To display the ProductExplorer GUI at the end of the datareduction:

- Click on the global parameter "ProductExplorerMode" before starting the data reduction. A configuration window will appear allowing you to set the execution mode of the Product Explorer. Valid options are:

[7] keep the mouse pointer on the file name to visualize the full path name.

  - "Triggered" (default). This option opens the ProductExplorer GUI when all the selected datasets have been reduced.
  - "Enabled". This option opens the ProductExplorer GUI at the end of the reduction of each individual dataset.
  - “Disable”. This option does not display the ProductExplorer GUI.

- Press the  button to start the workflow.

2. To display the ProductExplorer GUI “before” starting the data reduction:

- double click on the composite Actor "Inspect previously reduced data". A configuration window will appear. Set to "Yes" the field "Inspect previously reduced data (Yes/No)". Modify the field "Continue reduction after having inspected the previously reduced data? (Continue/Stop/Ask)". "Continue" will continue the workflow and trigger the DataOrganizer. "Stop" will stop the workflow; "Ask" will prompt another window deferring the decision whether continuing or not the reduction after having closed the Product Explorer.

- Press the  button to start the workflow. Now the ProductExplorer GUI will appear before starting the data organization and reduction.

Exploring the data reduction products

The left window of the ProductExplorer GUI shows the executions for all the datasets (see Fig. 6.4). Once you click on a dataset, you get the list of reduction attemps. Green and red flags identify successfull or unsucessfull reductions. Each reduction is linked to the “Description” tag assigned in the “Select Dataset” window.

1. To identify the desired reduction run via the “Description” tag, proceed as follows:

- Click on the symbol at the left of the dataset name. The full list of reduction attempts for that dataset will be listed. The column Exec indicates if the reduction was succesful (green flag: "OK") or not (red flag: "Failed").

- Click on the entries in the field "Description" to visualize the description you have entered associated to that dataset on the Select Dataset window when reducing the data.

- Identify the desired reduction run. All the products are listed in the central window, and they are organized following the data reduction cascade.

You can narrow down the range of datasets to search by configuring the field "Show" at the top-left side of the ProductExplorer (options are: "All", "Successful", "Unsuccessful"), and specifying the time range (Last, all, From-to).

2. To inspect the desired file, proceed as follows:

- Navigate through the data reduction cascade in the ProductExplorer by clicking on the files.
- Select the file to be inspected and click with the mouse right-hand button. The available options are:
  - Options available always:
    - Copy full path. It copies the full name of the file onto the clipboard. Shift+Ctr+v to past it into a terminal.
    - Inspect Generic. It opens the file with the FITS viewer selected in the main workflow canvas.
    - Inspect with. It opens the file with an executable that can be specified (you have to provide the full path to the executable).
  - Options available for files in the `TMP_PRODUCTS_DIR` directory only:
    - command line. Copy of the environment configuration and recipe call used to generate that file.
    - Xterm. It opens an Xterm at the directory containing the file.
  - Options available for products associated to interactive windows only:
    - Display pipeline results. It opens the interactive windows associated to the recipe call that generated the file. Note that this is for visualization purposes only; the recipe parameters cannot be changed and the recipe cannot be re-run from this window.

8.4 Workflow Details

This section describes the inner workings of some actors in more detail. It is intended for those users who wish to know what is going on behind the scenes, to diagnose errors or unexpected behvaior, or for those that wish to make modifications to the workflow.

8.4.1 Data Organisation and Selection Actors

The `Data Organiser` actor uses a special set of 'rules' to organise, classify, associate, and define purposes for data files. These rules are defined in a file called `hawki.oca`. The default location for this file is `<install_dir>/share/esopipes/<pipeline-version>/reflex/`. Users may edit this file to suit their data reduction needs. Note, however, that the syntax and implementation of rules in an .oca file are somewhat arcane. Please refer to the OCA User Manual [5] for details.

By default, the workflow will group data together by the start time of the **Observation Block** in which it was taken (`OBS.START`). This means that the workflow cannot be used to combine, stack, or tile images that were taken as part of OBs with different start times. As discussed in the HAWK-I Pipeline manual [3], the pipeline is not designed to process science data or standard star fields that span more than one OB. A key role of the `Data Organiser` is to assign one or more 'purposes' to each file. The purpose of a file identifies the reason why a file is included in a DataSet. It is used, in part, as a filtering and accounting mechanism to ensure the correct files are sent to the correct recipes in the correct order. A `DataFilter` actor may be inserted into any relationship (i.e. solid black line) to see which files are being broadcast and to see the purposes of each file. The syntax for a single purpose is `ACTION_1/ACTION_2/ACTION_3/ ... ACTION_n`, where each `ACTION_i` describes each processing step for this file. These actions are defined in the OCA rules. In this context, an ACTION comprises a recipe name, a list of required and optional inputs for that recipe, and the classification of

output files from the recipe. Note that a single file may have more than one purpose. For example, the same raw `DARK` file may be used to create a master dark image and to create a master readgain table.

The OCA rules are also used to assess whether or not a data set is complete. In order for a data set to be considered complete, a number of criteria must be met: a) there are a sufficient number of static calibration files of the right classification, b) there are enough files to create the appropriate master dark and master flat frames, and c) at least one raw standard star or science image is present. An incomplete data set will appear in the interactive DataSet Selection window in grey text. The 'missing' components of the data set are listed if the mouse is hovered over the row in which the missing data set appears, e.g. "MISSING MASTER_DARK". Inspecting incomplete data sets can be used to reveal why the organiser considers the data set to be incomplete. A user may still select incomplete data sets for processing, but the workflow will not be completed successfully.

8.4.2 Editing Recipe Parameters

Most parameters are explicitly set to their default values. The exceptions are in the `Standard Fields`, `Science Fields`, and `Science Postprocess` actors: the `savemstd` and `savecat` parameters are set to 'true' rather than the default value of 'false', and `cacheloc` is set to `$TMP_PRODUCTS_DIR` instead of '.'.

There are several ways to view or edit the value of recipe parameters. The choice of method depends on how a user prefers to interact with the workflow.

1. If `GlobalPlotInteractivity` is set to `true`, an interactive window will be launched when a recipe finishes. This window shows the values of the recipe parameters that were used to create the products for interactive inspection. Users may change the value of these parameters as needed and then click Re-run Recipe to run the recipe with the new parameters. Note that some recipes have a large number of parameters; some parameters may appear under another tab on the upper right-hand side of the interactive window.

2. From the main workflow canvas, a user may double-click on a composite actor that contains an actor that executes recipes (i.e. those with orange boxes around them). A list of recipe parameters, and perhaps other actor parameters, will appear. To edit a parameter, change the value in the box and press Commit.

3. If a user opens a composite actor that executes a recipe, the parameters can be seen on the canvas of that actor. The parameters are represented as "StringParameters" on the canvas and have a small red dot next to them. They can be changed by clicking on them in the canvas.

4. A user may double-click on the actor that runs a recipe; these actors share the name of the recipe they execute and appear to have a cylinder with a thick circular arrow on them. A window will appear that enables a user to change that actor's behaviour, including parameter values (see Figure 8.2). This method of changing recipe parameters is *not* recommended. Instead, most recipe parameters have a value set to `PORT`; this tells the actor to use specially crafted values from an input port. If a non-`PORT` value is specified, the actor will ignore any changes to parameters made using the three methods listed above. However, there are two recipe parameters that can only be changed using this method: `prettynames` and `preview_only`. They are both set to `false`; `true` values have no use in a workflow environment.

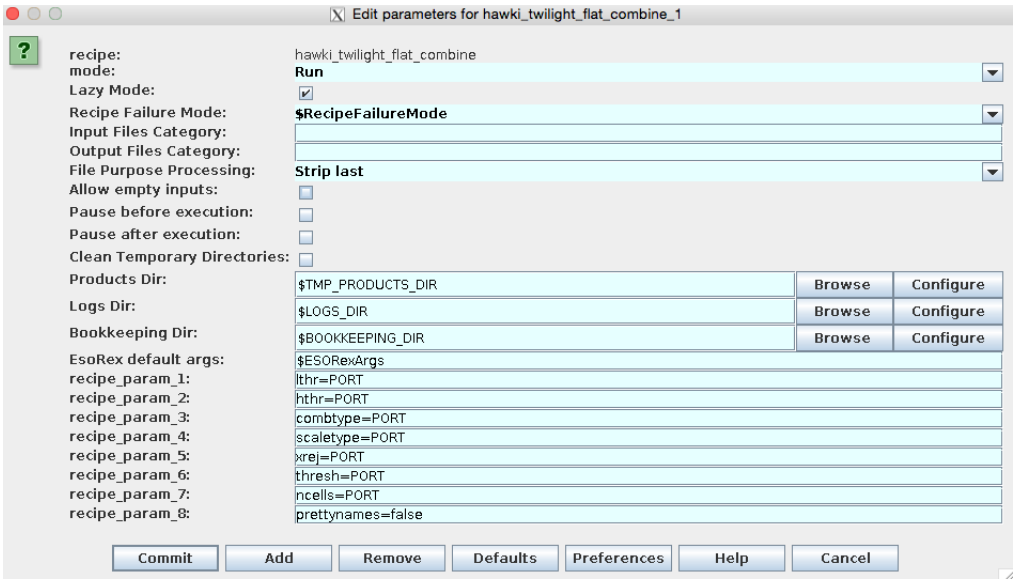


Figure 8.2: Example window for modifying master flat frame parameters.

8.4.3 Master Calibration Actors

The layout of the `MasterDark` and `MasterFlat` composite actors is fairly self-explanatory. However, the `Master Readgain` actor is complicated. The reason for the complexity is that a) a user can choose whether to run the recipe, and b) the master readgain table in the static calibration directory may be used instead of the recipe product. If the recipe is run, the interactive window compares the values in the static calibration file with the pipeline product.

8.4.4 Standard Star and Science Fields Actors

In addition to the main recipe loop, these actors check if any raw standard star or science fields are present before attempting to process them. A DataSet may only have one of these categories of files. If the required category of files is missing, the boolean logic in the actor ensures that the workflow will skip the recipe and create an empty product before moving to the next actor on the main canvas.

8.4.5 Output Organisation

After processing the input data for a particular DataSet, the workflow executes the `Product Renamer` actor. This actor copies the final products of the `Standard Fields`, `Science Fields`, and `SciencePostprocess` actors into the `END_PRODUCTS_DIR` directory and renames them with a name derived from values of certain FITS header keyword values. By default, the final products are renamed to a file of the form `<HIERARCH.ESO.OBS.NAME>_<HIERARCH.ESO.PRO.CATG>.fits`, with `<HIERARCH.ESO.OBS.NAME>` and `<HIERARCH.ESO.PRO.CATG>` representing the values of the corresponding FITS header keywords. If a file of that name already exists, an underscore followed by an incre-

mental integer is appended to the filename. A user may customise this format (or other actor behaviour) by right-clicking on the `Product Renamer` actor, selecting "Configure Actor", and then editing `RenameKeywords` as appropriate. Users are referred to the HAWK-I Pipeline Manual [3] for a description of the pipeline products and associated `HIERARCH.ESO.PRO.CATG` values.

8.5 Interactive Windows

The HAWK-I workflow contains six interactive windows that allow the user to iterate on the processing of their data. The windows are launched by a Python actor that is part of the main recipe execution loop. A user may inspect or change the Python script that runs these windows. The name and location of the source `.py` files can be seen by double-clicking on the Python actor.

Every interactive window shares the same eight buttons on the top left side of the window, e.g. a Home icon. These buttons control zooming in on images and plots, the colour scale of images, etc. Users are referred to the Reflex Users Manual[4] for details on the functionality of these buttons.

Each window also shares a similar layout on the right hand side. One or more tabs appear in the top right; these show the values of the recipe parameters used to generate the data shown on the left. A short description of each parameter, the default value, and accepted values are displayed if the mouse is hovered over the white box. Three buttons appear below the list of parameters: 1) Continue Wkf will close the window and the workflow will continue with the pipeline products, 2) Re-run Recipe will re-run the recipe with some new parameters, and 3) Help opens a small window describing how the interactivity works. The "Disable this window in subsequent runs" is a tick-box. If clicked, the window will change the value of `EnableInteractivity` to `false` after closing the window.

8.5.1 Master Dark

This window (Figure 8.3) opens by showing a master dark image, one chip per image panel. The title of each image shows the DIT, NDIT, and extension name of the image. The radio buttons on the upper left allow a user to change what is shown in each image panel. Clicking anywhere on the button or explanatory text will change the selection; note that there may be a short delay of 1-2 seconds between clicking and seeing the change.

If a reference dark is not provided to the recipe, the button options are a) the master dark image (default), and b) a histogram of the values in each image. The histograms show the pixel value distribution over ten bins within a range of the median value $\pm$ 7.4 times the median absolute deviation (equivalent to 5σ for a Gaussian distribution). The panels also show the useful values for how the pixel values are distributed (median, mean, median absolute deviation, and root-mean-square deviation from mean).

If a reference *is* provided to the recipe, two additional radio button options are available: c) an image of the reference dark subtracted from the master dark, and d) a scatter plot showing how the difference image varies over the chip in different "cells". The error bars on the scatter plot points are 1.48 times the median absolute deviation of the pixel values ($\approx 1\sigma$) within a cell on the difference image.

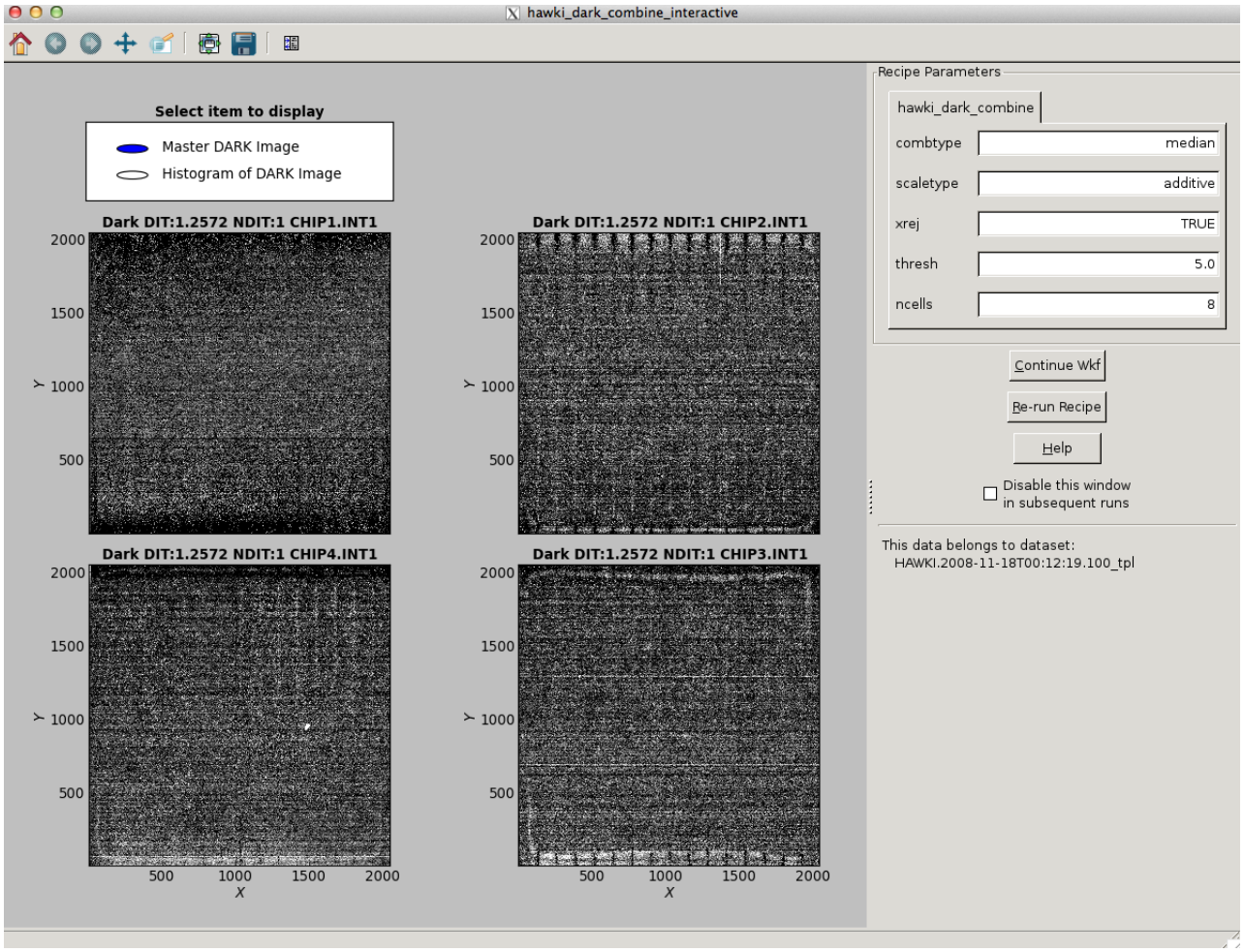


Figure 8.3: Example interactive window for evaluating master dark frames.

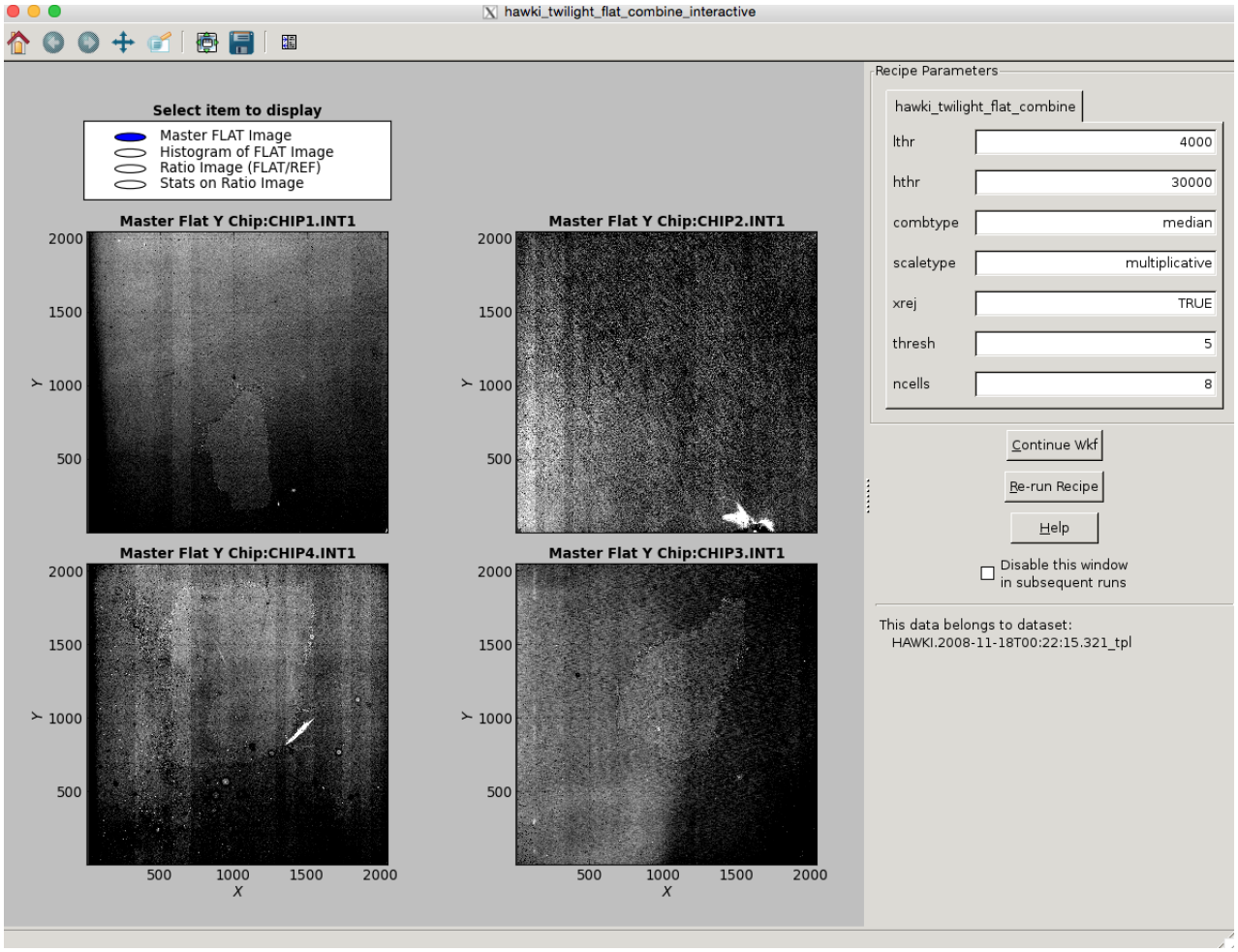


Figure 8.4: Example interactive window for evaluating master flat frames.

8.5.2 Master Flat

This window (Figure 8.4) opens by showing a master flat image, one chip per image panel. The title of each image shows filter and extension name of the image. The radio buttons on the upper left allow a user to change what is shown in each image panel. Clicking anywhere on the button or explanatory text will change the selection; note that there may be a short delay of 1-2 seconds between clicking and seeing the change.

If a reference flat is not provided to the recipe, the button options are a) the master flat image (default), and b) a histogram of the values in each image. The histograms (Figure 8.5) show the pixel value distribution over ten bins within a range of the median value $\pm$ 7.4 times the median absolute deviation (equivalent to 5σ for a Gaussian distribution). The panels also show the useful values for how the pixel values are distributed (median, mean, median absolute deviation, and root-mean-square deviation from mean).

If a reference flat *is* provided to the recipe, two additional radio button options are available: c) an image of the master flat divded by the reference flat (Figure 8.6), and d) a scatter plot showing how the ratio image varies

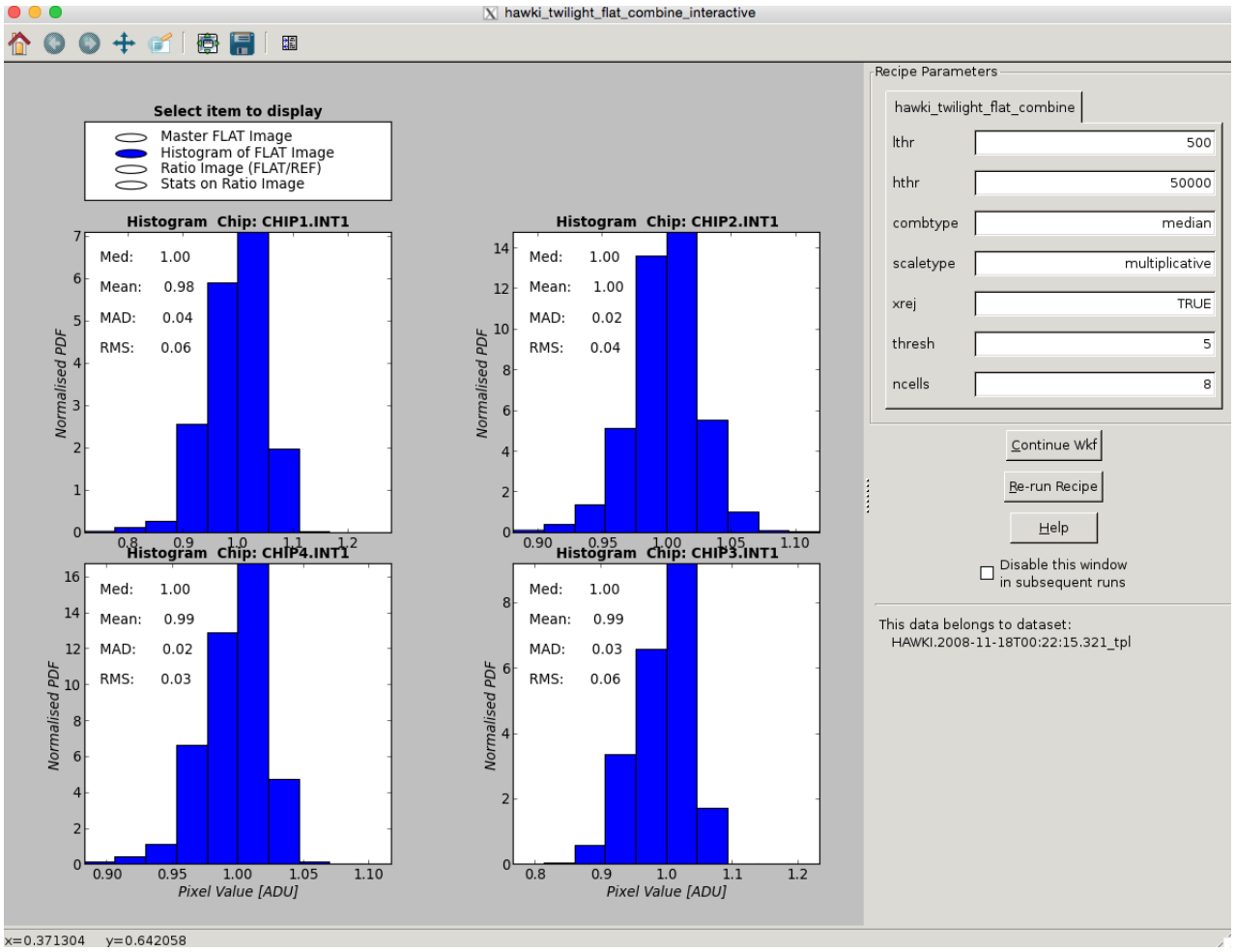


Figure 8.5: Example interactive window for evaluating master flat frames; histograms of pixel values are shown.

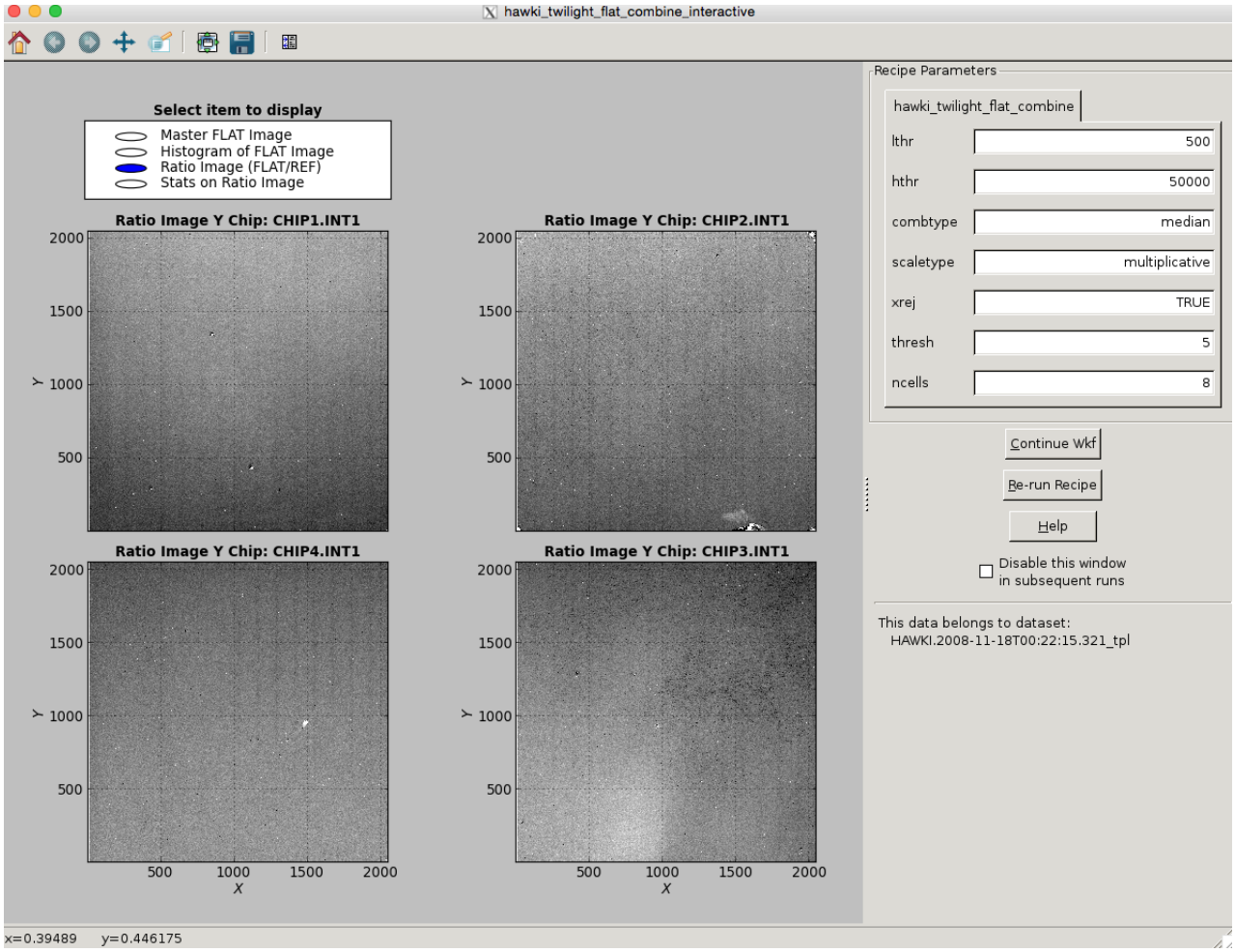


Figure 8.6: Example interactive window for evaluating master flat frames; the ratio of the master flat to a reference flat is shown.

over the chip in different "cells" (Figure 8.7). The error bars on the scatter plot points are 1.48 times the median absolute deviation of the pixel values ($\approx 1\sigma$) within a cell on the difference image.

8.5.3 Master Readgain

This window (Figure 8.8) shows two scatterplots; there are no radio buttons. The lefthand plot compares the readnoise values in the static calibration file ("reference readnoise") to the master readnoise values derived by the recipe; the units are ADUs. The righthand plot compares the gain values in the static calibration file ("reference gain") with the gain derived by the recipe; the axes are in units of electrons per ADU.

Each data point represents a different chip as indicated in the legend. In both panels, the solid blue line shows where the data points would be if the reference and master values are identical.

If any of the values in the recipe product are NULL or less than zero, the interactive window will not show any

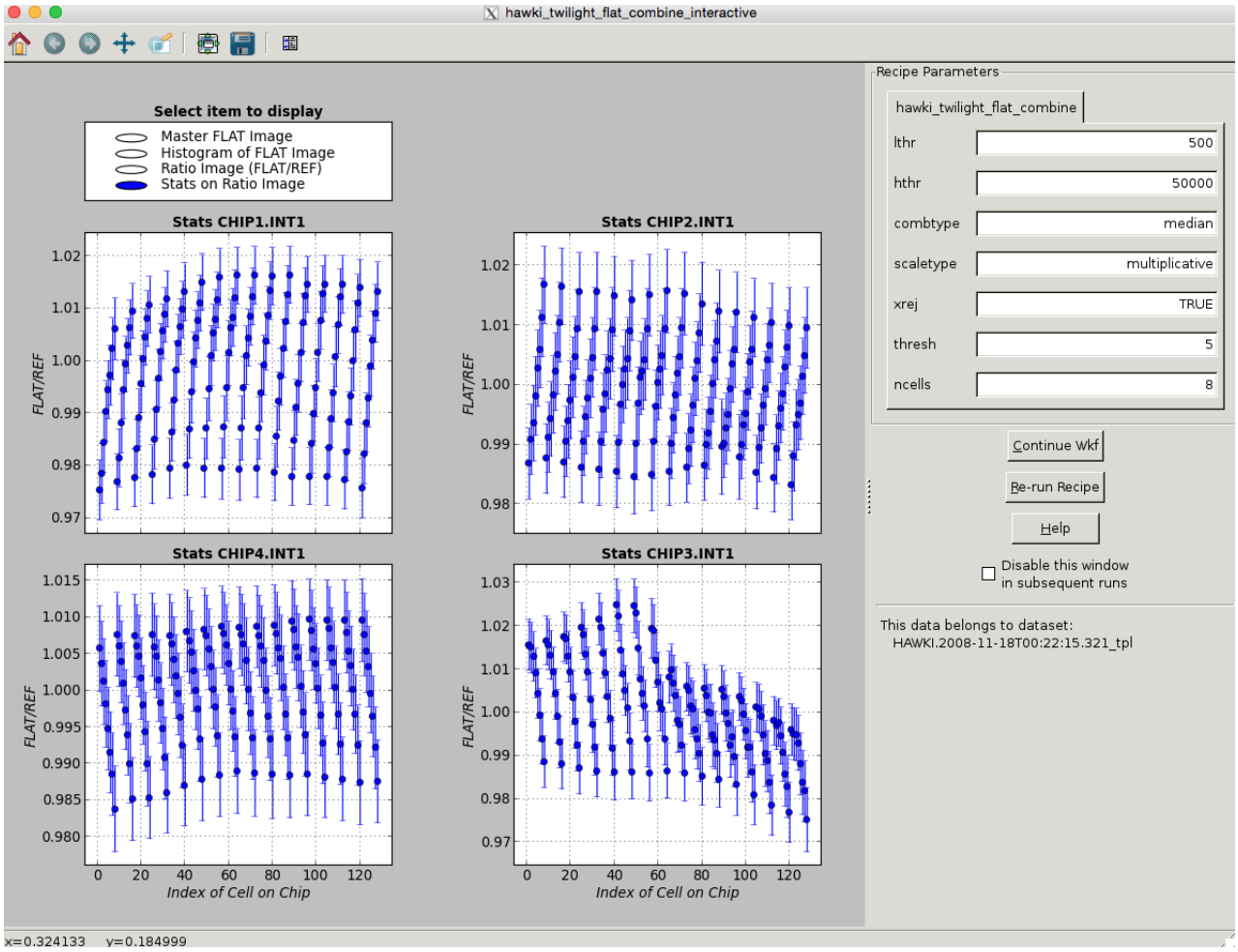


Figure 8.7: Example interactive window for evaluating master flat frames; a statistical description of the ratio image in Figure 8.6 is shown.

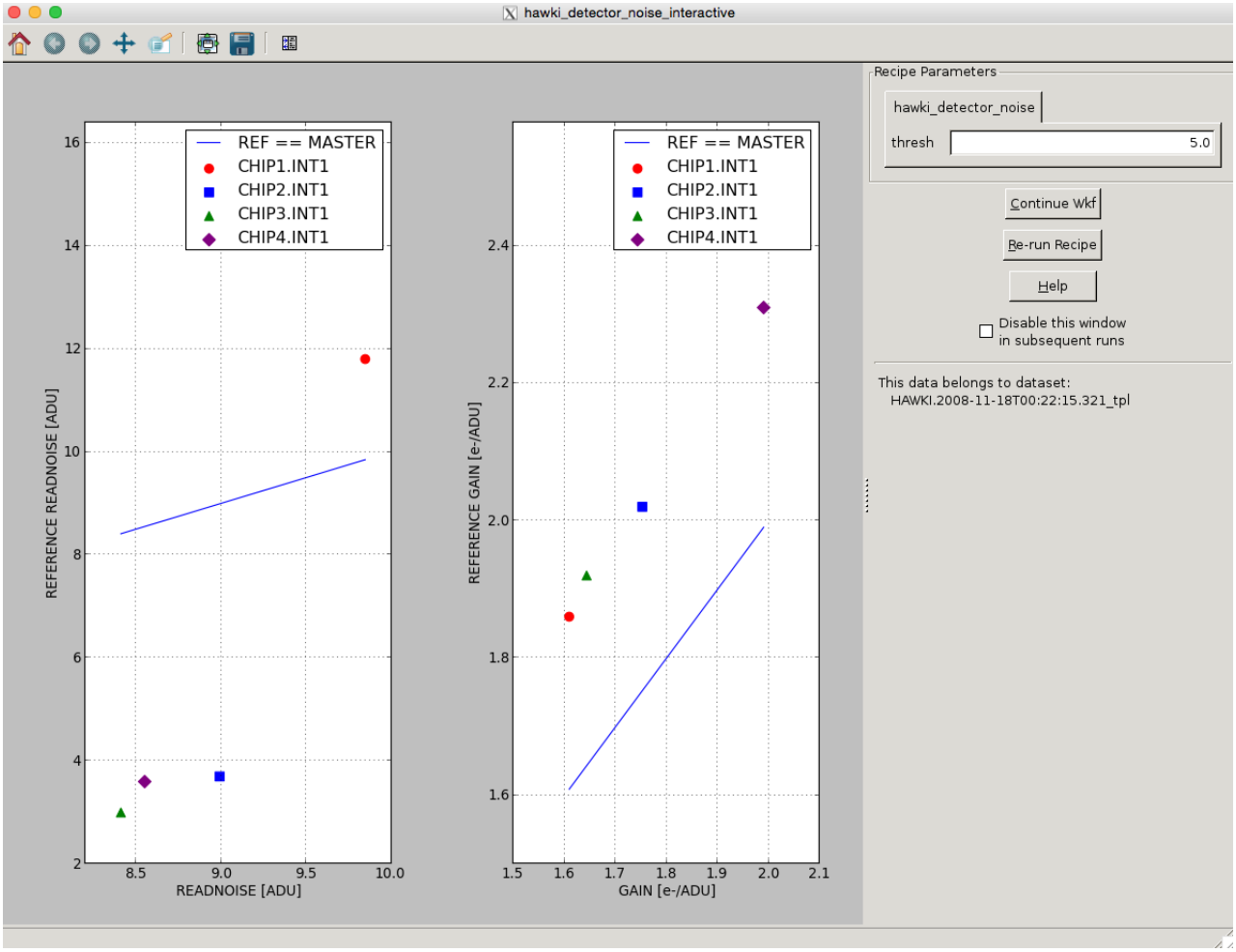


Figure 8.8: Example interactive window for evaluating master readnoise and gain values.

data. Instead, a message indicating that at least one value is invalid is displayed.

8.5.4 Standard Stars

This window opens by showing a processed image of a standard star field, one chip per panel. The title of each panel indicates which image this is in the sequence of standard star fields. For example, if the recipe is provided with four `STD` files, the window will open with an image with `1/4` in the title to indicate that the first image (out of four) is shown. The full pathname of the file being displayed is shown if the mouse hovers over the image. The title also shows the extension name of the image.

There are three boxes with radio buttons across the top of the window. The button in the leftmost "Select group:" box does not respond to any clicks; it only indicates that the window is showing processed `STD` images.

The middle box ("Select item in group:") show five options:

1. Image (default): show processed image (tag `BASIC_CALIBRATED_STD`).

2. Assess matched astrom stds: if the recipe parameter `savemstd` is set to `true`, clicking this button will show a comparison between the location of the astrometric standards in the reference catalogue (e.g. 2MASS) and the derived location of the same objects on the calibrated image. Each image panel is now split into two subpanels. The top subpanel shows the difference between the right ascension (weighted by cos(DEC)) for each catalogued standard in units of arcseconds. The bottom subpanel shows the difference between the declination for each catalogued standard in units of arcseconds. In all subpanels, the x-axis refers to the row number of that object in the matched standard catalogue table. The full pathname to the table is shown if the mouse hovers over any panel. These plots may be useful to identify any outliers or systematic error in the astrometric calibration.

3. Histogram of matched astrom stds: if the recipe parameter `savemstd` is set to `true`, clicking this button will show a histogram of the angular distance between the celestial coordinates of astrometric standards in the reference catalogue and the same objects on the calibrated image (in units of arcseconds). All data points are split into 10 bins in the histogram. The x-axis of each panel is angular distance; the y-axis is the frequency of occurrence. The red text in the upper right of each panel shows some statistical values describing the distribution of all the data points (median, mean, median absolute deviation, and rms deviation from the mean).

4. Assess matched photom stds: if the recipe parameter `savemstd` is set to `true`, clicking this button will show a comparison between the flux of the photometric standards in the reference catalogue (e.g. 2MASS) and the flux of the same objects on the calibrated image. This is represented as a magnitude zero point (using the measured fluxes from the "aper5" column of the object catalogue). The x-axis refers to the row number of each object in the matched standard catalogue table. The full pathname to that table is shown if the mouse hovers over any panel. Note that not all objects in the matched standard catalogue are shown; only those used for calibration because they have an error less than `magerrcut` are displayed on this plot.

5. Histogram of matched photom stds: if the recipe parameter `savemstd` is set to `true`, clicking this button will show a histogram of the magnitude zero points from the item above. All data points are split into 10 bins in the histogram. The x-axis of each panel is magnitude zero point in units of (Vega) magnitudes; the y-axis is the frequency of occurrence. The red text in the upper left of each panel shows some statistical values describing the distribution of all the data points (median, mean, median absolute deviation, and rms deviation from the mean).

The rightmost box contains one radio button "Click to advance to next item (if available)". Clicking this button will advance each panel to show the same information for the next processed image. For example, if a user is looking at a histogram of matched photometric standards for the fourth image (out of four), clicking the button will show the same information, but for the first image in the sequence.

Two types of standard star observations exist in HAWK-I data and both are support by the HAWK-I pipeline. From the beginning of HAWK-I operations, the standard star template observed the same (single) standard star, placed in the centre of each detector, in a series of four exposures. From about mid-2015, the standard star template observes one of several 2MASS touchstone fields. These fields have been chosen to fill the HAWK-I field of view with a large number of standard stars.

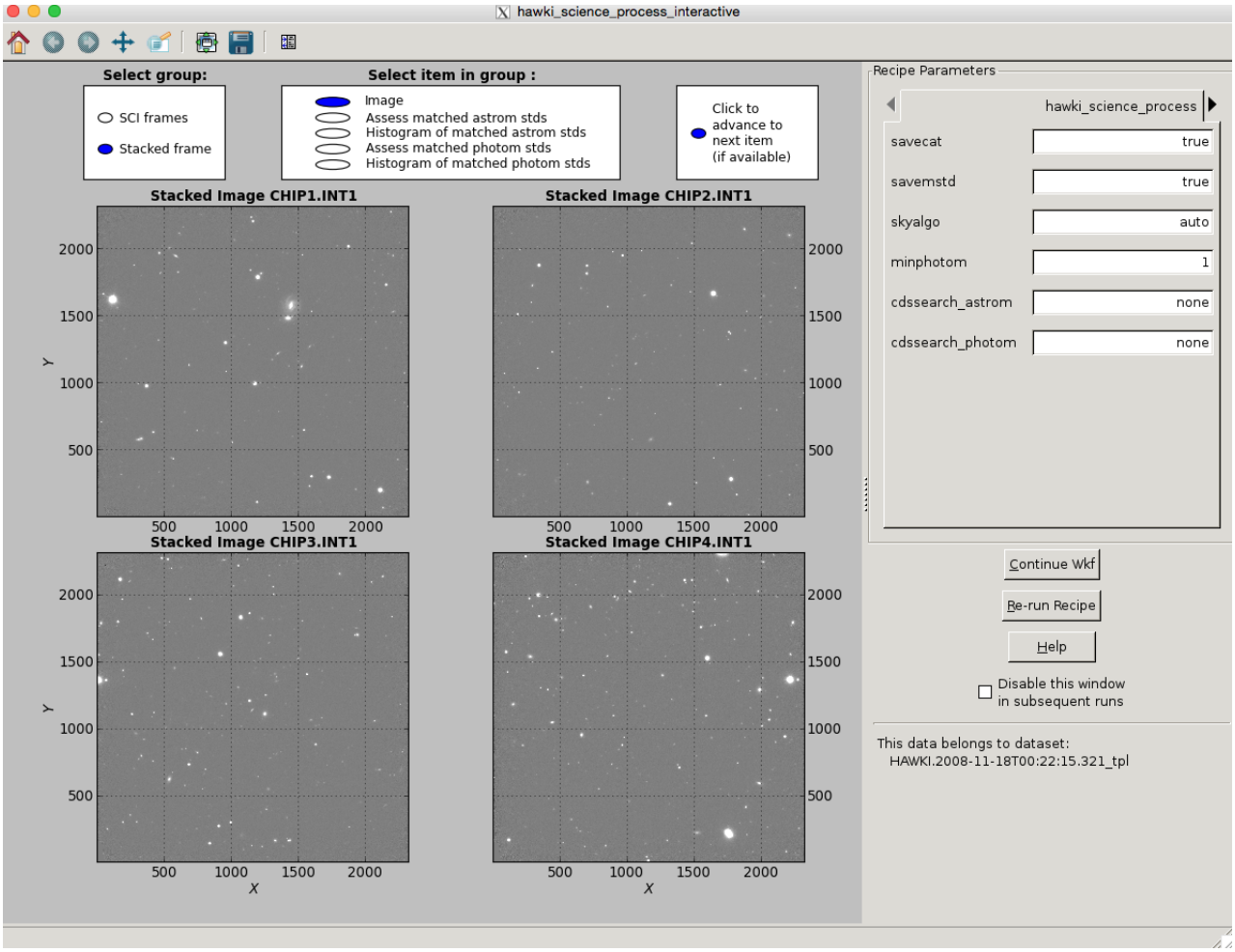


Figure 8.9: Example interactive window for evaluating the results of `hawki_science_process`.

8.5.5 Science Fields

This window (Figure 8.9) opens by showing a processed, stacked image of a science field (one chip per panel). The full pathname of the file being displayed is shown if the mouse hovers over the image. The title shows the extension name of the image.

There are three boxes with radio buttons across the top of the window. The buttons in the leftmost "Select group:" box are used to select either the individual processed science images (`SCI` tag) or the single stacked image. If `SCI` is selected, the title of each panel will indicate which image in the sequence is being shown, e.g. "1/20" for the first image in a sequence of twenty. As above, the full pathname of the file being displayed is shown if the mouse hovers over an image.

The middle box ("Select item in group:") show five options:

1. Image: show processed image. If `SCI` is selected in the lefthand box, the image is from a file with a

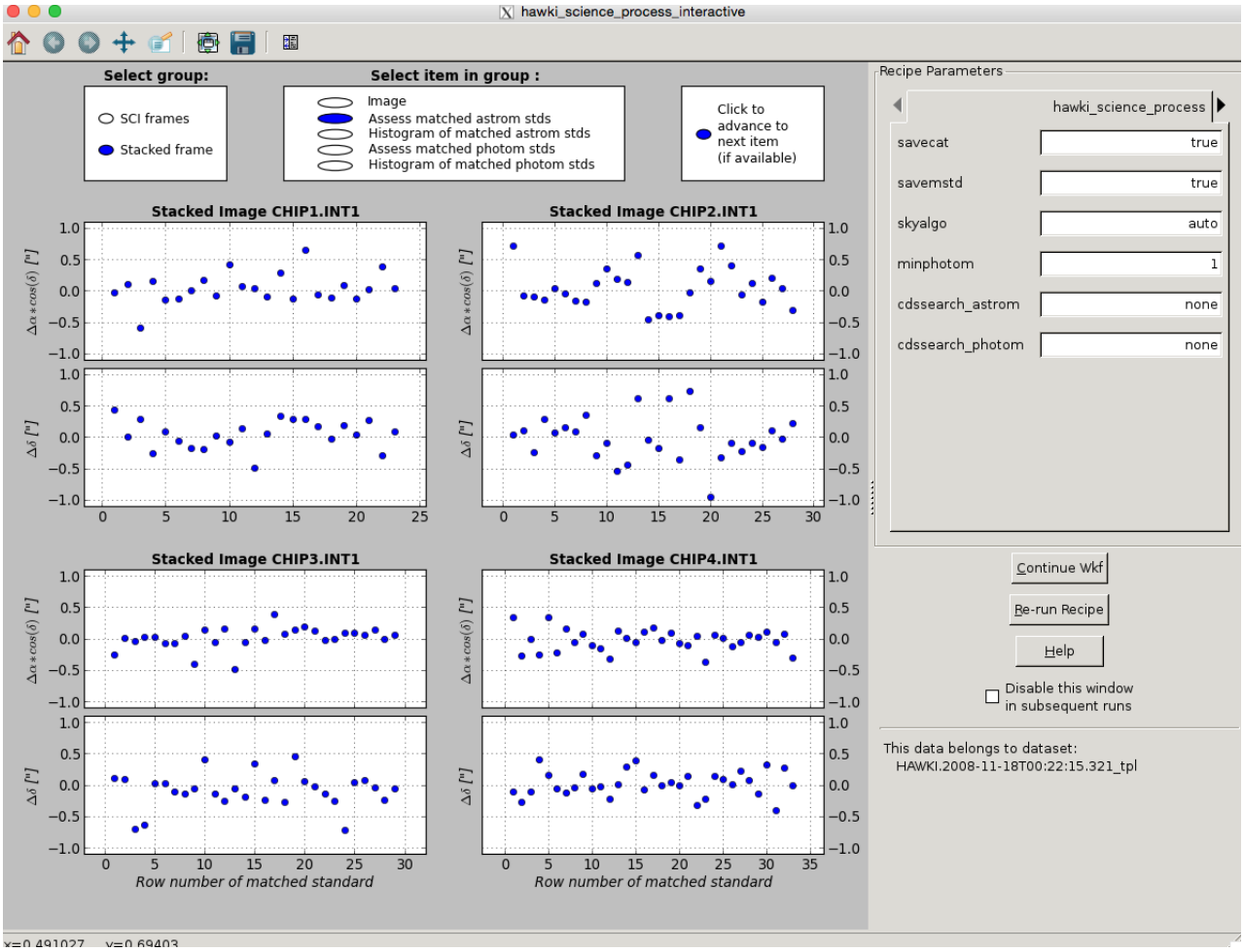


Figure 8.10: Example interactive window for evaluating the results of `hawki_science_process`; the offsets between the calibrated coordinates of objects found in the science image and those found in the reference catalogue are shown.

`BASIC_CALIBRATED_SCI` tag. Otherwise, it is the `JITTERED_IMAGE_SCI` file (default).

2. Assess matched astrom stds (Fig8.10): if the recipe parameter `savemstd` is set to `true`, this button shows a comparison between the location of the astrometric standards in the reference catalogue (e.g. 2MASS) and the derived location of the same objects on the selected calibrated image. Each image panel is now split into two subpanels. The top subpanel shows the difference between the right ascension (weighted by cos(dec)) for each catalogued standard in units of arcseconds. The bottom subpanel shows the difference between the declination for each catalogued standard in units of arcseconds. In all subpanels, the x-axis refers to the row number of that object in the matched standard catalogue table. The full pathname to the table is shown if the mouse hovers over any panel. These plots may be useful to identify any outliers or systematic error in the astrometric calibration.

3. Histogram of matched astrom stds (Fig8.11): if the recipe parameter `savemstd` is set to `true`, this

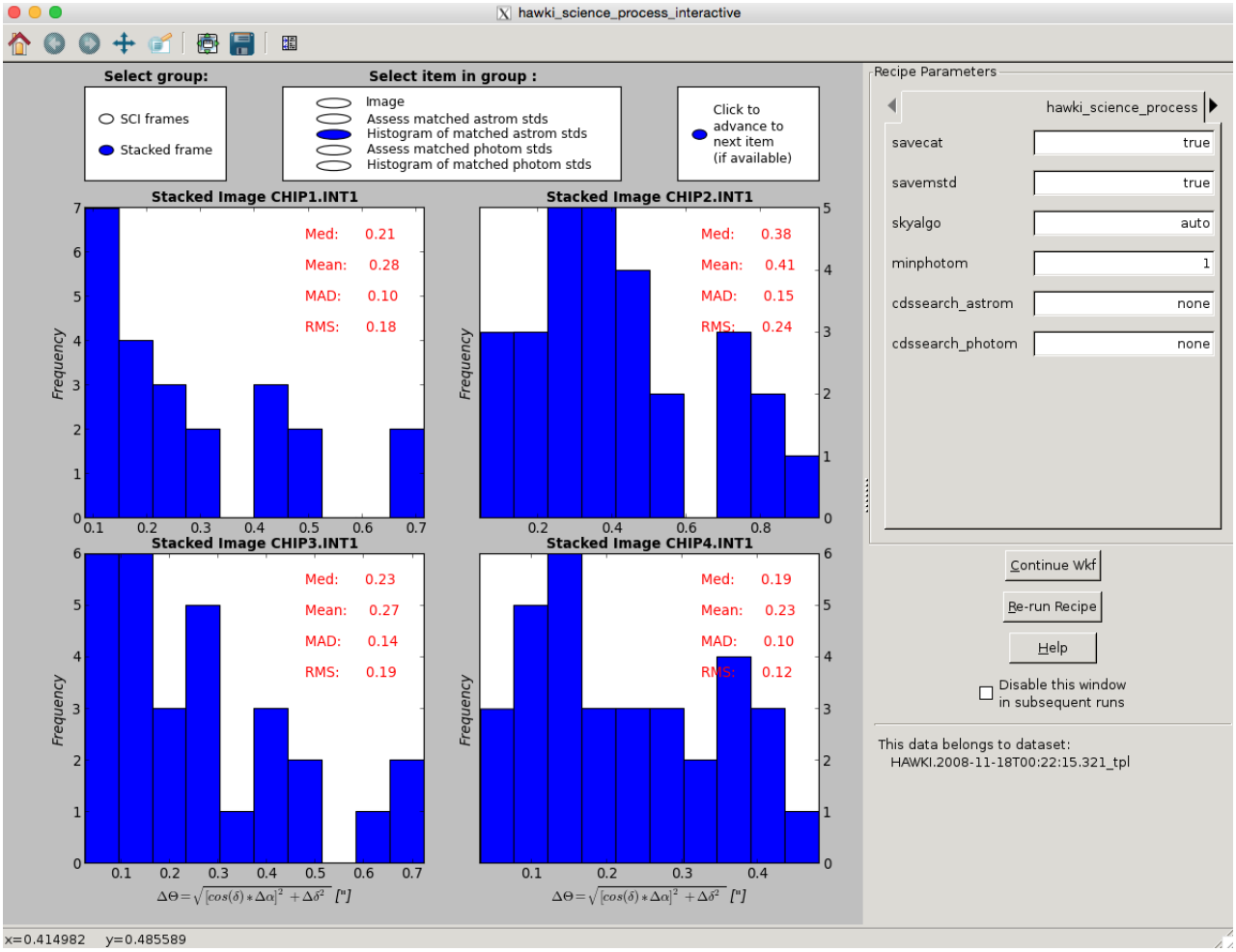


Figure 8.11: Example interactive window for evaluating the results of `hawki_science_process`; a histogram of offsets shown in Figure 8.10 are shown.

button shows a histogram of the angular distance (in arcseconds) between the celestial coordinates of astrometric standards in the reference catalogue and the same objects on the selected calibrated image. All data points are split into 10 bins in the histogram. The x-axis of each panel is angular distance; the y-axis is the frequency of occurrence. The red text in the upper right of each panel shows some statistical values describing the distribution of all the data points (median, mean, median absolute deviation, and rms deviation from the mean).

4. Assess matched photom stds (Fig8.12): if the recipe parameter `savemstd` is set to `true` and "Stacked frame" is selected in the lefthand radio button box, this button shows a comparison between the flux of the photometric standards in the reference catalogue (e.g. 2MASS) and the flux of the same objects on the calibrated image. This is represented as a magnitude zero point (using the measured fluxes from the "aper5" column of the object catalogue). The x-axis refers to the row number of each object in the matched standard catalogue table. The full pathname to that table is shown if the mouse hovers over any

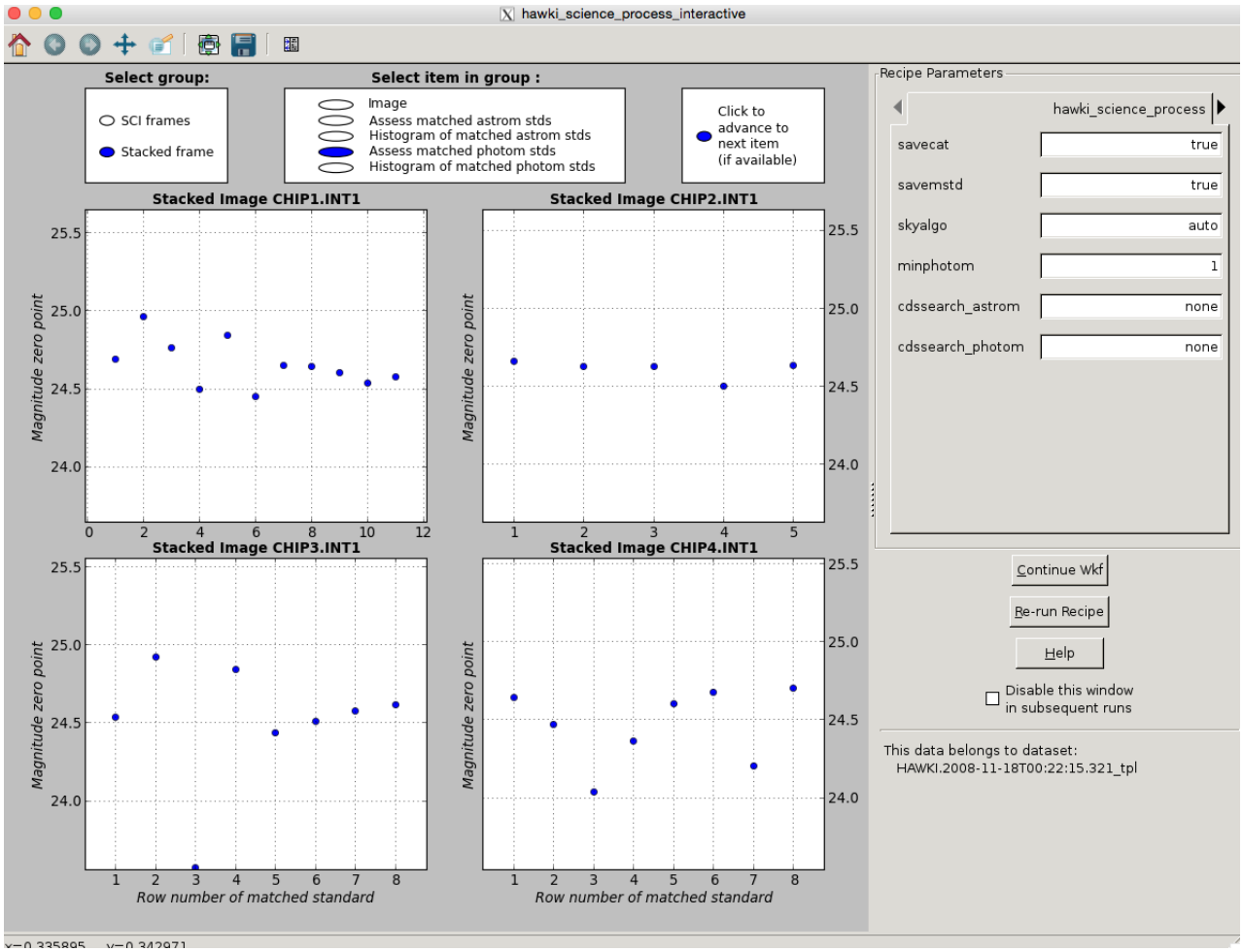


Figure 8.12: Example interactive window for evaluating the results of `hawki_science_process`; the inferred magnitude zero points for objects on the calibrated science image are shown.

panel. No matched photometric standard catalogues are created for individual science frames. Note that not all objects in the matched standard catalogue are shown; only those used for calibration because they have an error less than `magerrcut` are displayed on this plot.

5. Histogram of matched photom stds (Fig8.13): if the recipe parameter `savemstd` is set to `true` and "Stacked frame" is selected in the lefthand radio button box, this button shows a histogram of the magnitude zero points from the item above. All data points are split into 10 bins in the histogram. The x-axis of each panel is magnitude zero point in units of magnitudes; the y-axis is the frequency of occurrence. The red text in the upper left of each panel shows some statistical values describing the distribution of all the data points (median, mean, median absolute deviation, and rms deviation from the mean).

The rightmost box contains one radio button "Click to advance to next item (if available)". Clicking this button will advance each panel to show the similar information for the next image in the sequence. For example, if a user is looking at a histogram of matched astrometric standards for the fourth image (out of four), clicking the

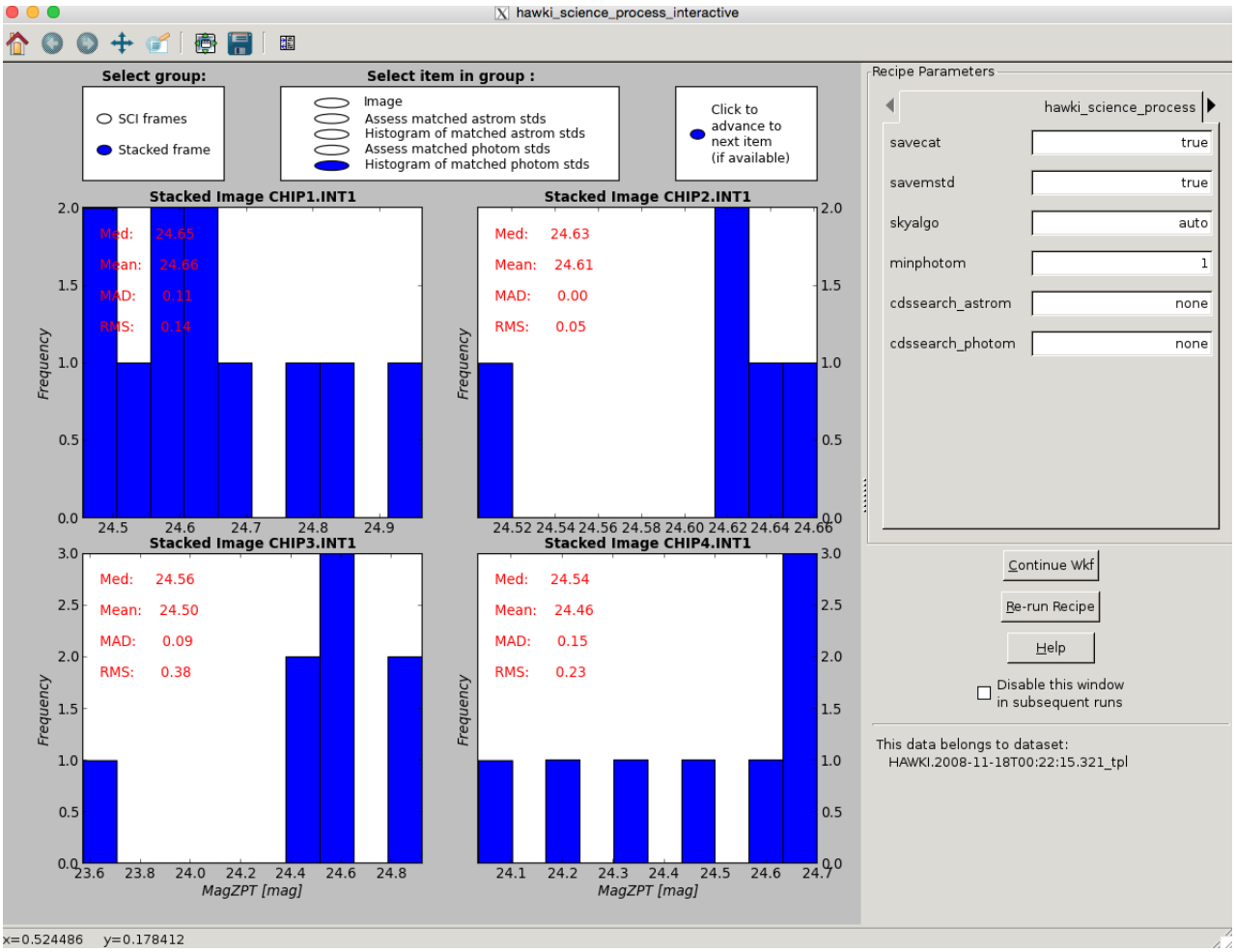


Figure 8.13: Example interactive window for evaluating the results of `hawki_science_process`; a histogram of magnitude zero points from Figure 8.12 are shown.

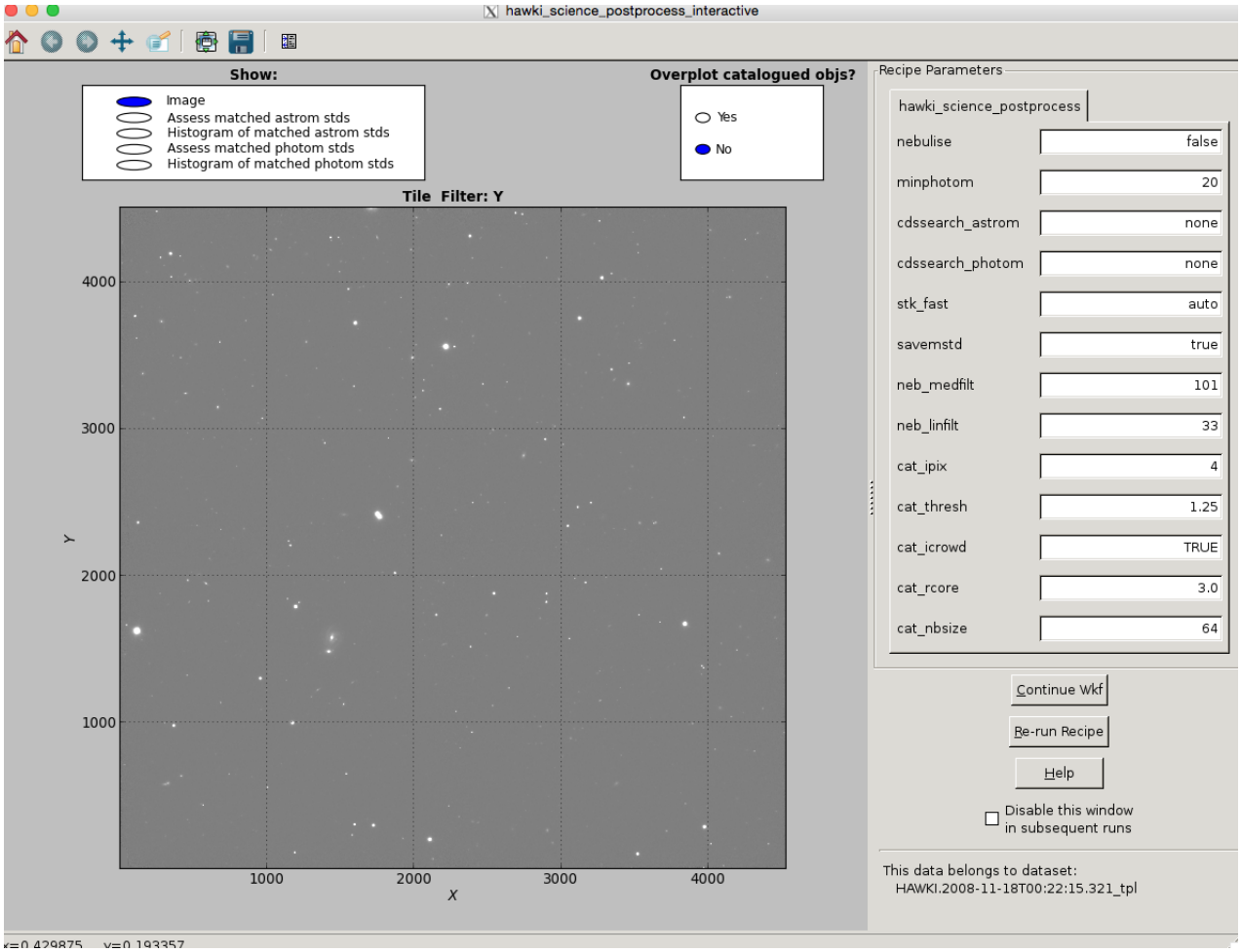


Figure 8.14: Example interactive window for evaluating the results of `hawki_science_postprocess`; the `TILED_IMAGE` is shown.

button will show similar plots, but for the first image in the sequence.

8.5.6 Science Postprocess

This window (Figure 8.14) opens to show one tiled image of a science field. The full pathname of the file being displayed is shown if the mouse hovers over the image.

In the lefthand box at the top of the window ("Show:"), there are five radio button options:

1. Image: show the `TILED_IMAGE` file.
2. Assess matched astrom stds: if the recipe parameter `savemstd` is set to `true`, this button shows a comparison between the location of the astrometric standards in the reference catalogue (e.g. 2MASS) and the derived location of the same objects on the tile. The image panel is now split into two subpanels. The

top subpanel shows the difference between the right ascension (weighted by cos(dec)) for each matched standard in units of arcseconds. The bottom subpanel shows the difference between the declination for each matched standard in units of arcseconds. In all subpanels, the x-axis refers to the row number of that object in the matched standard catalogue table. The full pathname to the table is shown if the mouse hovers over any panel. These plots may be useful to identify any outliers or systematic error in the astrometric calibration.

3. Histogram of matched astrom stds: if the recipe parameter `savemstd` is set to `true`, this button shows a histogram of the angular distance (in arcseconds) between the celestial coordinates of astrometric standards in the reference catalogue and the same objects on the tile. All data points are split into 10 bins in the histogram. The x-axis of each panel is angular distance; the y-axis is the frequency of occurrence. The red text in the upper right of each panel shows some statistical values describing the distribution of all the data points (median, mean, median absolute deviation, and rms deviation from the mean).

4. Assess matched photom stds: if the recipe parameter `savemstd` is set to `true`, this button shows a comparison between the flux of the photometric standards in the reference catalogue (e.g. 2MASS) and the flux of the same objects on the calibrated image. This is represented as a magnitude zero point (using the measured fluxes from the "aper5" column of the object catalogue). The x-axis refers to the row number of each object in the matched standard catalogue table. The full pathname to that table is shown if the mouse hovers over any panel. Note that not all objects in the matched standard catalogue are shown; only those used for calibration because they have an error less than `magerrcut` are displayed on this plot.

5. Histogram of matched photom stds: if the recipe parameter `savemstd` is set to `true`, this button shows a histogram of the magnitude zero points from the item above. All data points are split into 10 bins in the histogram. The x-axis of each panel is magnitude zero point in units of magnitudes; the y-axis is the frequency of occurrence. The red text in the upper left of each panel shows some statistical values describing the distribution of all the data points (median, mean, median absolute deviation, and rms deviation from the mean).

In the righthand box near the top of the window, there are two radio buttons that control whether the location of catalogued objects are overlayed onto the image. These options only affect the image, ie. if "Show: Image" is selected on the left.

8.6 Improving Results through Workflow Interaction

The workflow is a convenient way to process data without much effort needed by the user. The default values for the recipes are selected to produce high quality results in most circumstances. However, the automatic nature of the workflow means that a user may not know if the pipeline products are valid. This section describes a few tips to assess if the processed data matches a user's expectations for accuracy and usefulness.

Check the log files. Each recipe will create a log file with information, warnings, and errors that occurred during processing. In several instances, such warning or errors may not cause the workflow to stop. Depending on a users's workflow settings, there may be no messages that appear to tell a user that a warning or error occurred. Users are therefore strongly encouraged to check the contents of every logfile for each recipe. The log files can be found in `$LOGS_DIR/<recipe_name>/esorex.log`.

Check the astrometric calibration. The recipes will attempt to calibrate the astrometry by fitting the location of point sources to a model of the World Coordinate System (WCS). One way to check this is to examine the matched astrometric standards catalogue in the interactive windows. An alternative is to look at the value of the Quality Control FITS header keyword `QC.WCS_RMS`. Values greater than $\approx 0.25''$may indicate problems with the fitting. Common sources of problems with WCS fitting are a) non-photometric conditions, b) very few standards in the field, c) extremely inaccurate values of the approximate location of the field (RA/DEC keywords from the telescope control system). Large errors in the WCS fit may also lead to strange results for the stacked or tiled images. Another recommendation is to check the value of the mean ellipticity of point sources in the object catalogues `QC.ELLIPTICITY`. The ellipticity of objects in individual images should be very similar to the ellipticity of objects in the stack and tile.

Check the photometric calibration. The recipes calculate the fluxes from objects that appear in the photometric calibration catalogue to evaluate a magnitude zero point for each object. These values can be viewed in the interactive windows as described above. An alternative is to check the `QC.MAGZPT` and `QC.MAGZERR` keywords. If a recipe used one or more inappropriate calibration sources, these can be edited out of the matched standard catalogue and fed back into the recipe for re-calibration. However, this cannot be done using the workflow; it can only be done by running the pipeline with `esorex`. Another option using the workflow is to re-run the recipe with a different value of `magerrcut`.

9 Frequently Asked Questions

- **The error window fills the whole screen - how can I get to the `Continue`/`Stop` buttons?**

  Press the Alt key together with your left mouse button to move the window upwards and to the left. At the bottom the `Continue`/`Stop` buttons will be visible. This bug is known but could not yet be fixed.

- **I tried to `Open` (or `Configure`) an `Actor` while the workflow is running and now it does not react any more. What should I do?**

  This is a limitation of the underlying Kepler engine. The only way out is to kill the workflow externally. If you want to change anything while a workflow is running you first need to pause it.

- **After a successful reduction of a data set, I changed this data set in some way (e.g. modified or removed some files, or changed the rules of the Data Organizer). When I restart Reflex, the Data Set Chooser correctly displays my new data set, but marks it as "reduced ok", even though it was never reduced before. What does this mean?**

  The labels in the column "Reduced" of the Data Set Chooser mark each dataset with "OK", "Failed" or "-". These labels indicate whether a data set has previously successfully been reduced at least once, all previous reductions failed, or a reduction has never been tried respectively. Data sets are identified by their name, which is derived from the first science file within the data set. As long as the data set name is preserved (i.e. the first science file in a data set has not changed), the Data Organizer will consider it to be the same data set. The Data Organizer recognizes any previous reductions of data sets it considers to be the same as the current one, and labels the current data set with "OK" if any of them was successful, even if the previously reduced data set differs from the current one.

  Note that the Product Explorer will list all the previous reductions of a particular data set only at the end of the reduction. This list might include successful and/or unsuccessful reduction runs with different parameters, or in your case with different input files. The important fact is that these are all reductions of data sets with the same first raw science file. By browsing through all reductions of a particular raw science file, the users can choose the one they want to use.

- **Where are my intermediate pipeline products?** Intermediate pipeline products are stored in the directory `<TMP_PRODUCTS_DIR>` (defined on the workflow canvas, under Setup Directories) and organised further in directories by pipeline recipe.

- **Can I use different sets of bias frames to calibrate my flat frames and science data?** Yes. In fact this is what is currently implemented in the workflow(s). Each file in a DataSet has a purpose attached to it ([4]). It is this purpose that is used by the workflow to send the correct set of bias frames to the recipes for flat frame combination and science frame reduction, which may or may not be the same set of bias frames in each case.

- **Can I run `Reflex` from the command line?** Yes, use the command:

  ```
  esoreflex -n <workflow_path>/<workflow>.xml
  ```

  The -n option will set all the different options for Kepler and the workflows to avoid opening any GUI elements (including pipeline interactive windows).

  It is possible to specify workflow variables (those that appear in the workflow canvas) in the command line. For instance, the raw data directory can be set with this command:

```
esoreflex -n -RAW_DATA_DIR <raw_data_path> \
             <workflow_path>/<workflow>.xml
```

You can see all the command line options with the command `esoreflex -h`.

Note that this mode is not fully supported, and the user should be aware that the path to the workflow must be absolute and even if no GUI elements are shown, it still requires a connection to the window manager.

- **How can I add new actors to an existing workflow?** You can drag and drop the actors in the menu on the left of the `Reflex` canvas. Under `Eso-reflex -> Workflow` you may find all the actors relevant for pipeline workflows, with the exception of the recipe executer. This actor must be manually instantiated using `Tools -> Instantiate Component`. Fill in the "Class name" field with `org.eso.RecipeExecuter` and in the pop-up window choose the required recipe from the pull-down menu. To connect the ports of the actor, click on the source port, holding down the left mouse button, and release the mouse button over the destination port. Please consult the Reflex User Manual ([4]) for more information.

- **How can I broadcast a result to different subsequent actors?** If the output port is a multi-port (filled in white), then you may have several relations from the port. However, if the port is a single port (filled in black), then you may use the black diamond from the toolbar. Make a relation from the output port to the diamond. Then make relations from the input ports to the diamond. Please note that you cannot click to start a relation from the diamond itself. Please consult the Reflex User Manual ([4]) for more information.

- **How can I manually run the recipes executed by Reflex?** If a user wants to re-run a recipe on the command line he/she has to go to the appropriate reflex_book_keeping directory, which is generally reflex_book_keeping/<workflow>/<recipe_name>_<number> There, subdirectories exist with the time stamp of the recipe execution (e.g. 2013-01-25T12:33:53.926/). If the user wants to re-execute the most recent processing he/she should go to the `latest` directory and then execute the script `cmdline.sh`. Alternatively, to use a customized `esorex` command the user can execute

```
ESOREX_CONFIG="INSTALL_DIR/etc/esorex.rc"
PATH_TO/esorex --recipe-config=<recipe>.rc <recipe> data.sof
```

where INSTALL_DIR is the directory where Reflex and the pipelines were installed.

If a user wants to re-execute on the command line a recipe that used a specific raw frame, the way to find the proper data.sof in the bookkeeping directory is via `grep <raw_file> */data.sof`. Afterwards the procedure is the same as before.

If a recipe is re-executed with the command explained above, the products will appear in the directory from which the recipe is called, and not in the reflex_tmp_products or reflex_end_products directory, and they will not be renamed. This does not happen if you use the `cmdline.sh` script.

- **Can I reuse the bookkeeping directory created by previous versions of the pipeline?**

  In general no. In principle, it could be reused if no major changes were made to the pipeline. However there are situations in which a previously created bookkeeping directory will cause problems due to pipeline versions incompatibility. This is especially true if the parameters of the pipeline recipes have changed. In that case, please remove the bookkeeping directory completely.

- **How to insert negative values into a textbox?**

  Due to a bug in wxPython, the GUI might appear to freeze when attempting to enter a negative number in a parameter's value textbox. This can be worked around by navigating away to a different control in the GUI with a mouse click, and then navigating back to the original textbox. Once focus is back on the original textbox the contents should be selected and it should be possible to replace it with a valid value, by typing it in and pressing the enter key.

- **I've updated my Reflex installation and when I run esoreflex the process aborts. How can I fix this problem?**

  As indicated in Section 4, in case of major or minor (affecting the first two digit numbers) Reflex upgrades, the user should erase the `$HOME/KeplerData`, `$HOME/.kepler` directories if present, to prevent possible aborts (i.e. a hard crash) of the esoreflex process.

- **How can include my analysis scripts and algorithms into the workflow?**

  EsoReflex is capable of executing any user-provided script, if properly interfaced. The most convenient way to do it is through the Python actor. Please consult the tutorial on how to insert Python scripts into a workflow available here: `www.eso.org/sci/data-processing/Python_and_esoreflex.pdf`

- **How do I convert the source flux values given in the object catalogues to magnitudes?**
  In the object catalogues (`OBJECT_CATALOGUE_SCI` and `BASIC_CAT_STD`), the flux and error within a specified radius aperture, typically set so that R_aperture=⟨FWHM⟩ where the quantity in angle brackets is the mean FWHM of all stellar images. This is also known as the *core radius*. The apertures here correspond to (0.5, $\frac{1}{\sqrt{2}}$, 1, $\sqrt{2}$, 2, $\frac{2}{\sqrt{2}}$, 4, 5, 6, 7, 8, 10, and 12) times the core radius.

  Converting the source catalogue fluxes (here, using the $1\times$ *core radius* aperture (`Aper_flux_3`) given in the catalogue) to magnitudes can be done with the following relation:

$$\text{magnitude} = \texttt{PHOTZP} - 2.5 * \log_{10}(\texttt{Aper_flux_3}/\texttt{EFF_EXPT}) - \texttt{APCOR3} - \texttt{DRS.EXTINCT}$$

  where uppercase parameters indicate header keywords:
  `PHOTZP` = the photometric zeropoint [magnitude]
  `EFF_EXPT` = the exposure time for averaged frames [seconds]
  `APCOR3` = the stellar aperture correction for 1 * the core radius flux [magnitude]
  `HIERARCH ESO DRS EXTINCT` = the assumed extinction [magnitude]

- **I no longer see a comparison with the reference darks and flats in the interactive actors?**
  When you process your own data it is possible that you will not see a comparison with the reference darks and flats in the interactive actors for the dark and flat recipes. This was probably visible when running the tutorial data set, but not when processing your own data. During the Reflex installation, the reference images are distributed along with the tutorial data set. To include the reference images in **all** of your data processing, simply move all of the REF_*.fits files (i.e. REF_FLAT_Ks.fits etc.) to your calibration directory (CALIB_DATA_DIR as listed in the top line of your Reflex workflow canvas).

[1] R. M. Cutri and et al. VizieR Online Data Catalog: AllWISE Data Release (Cutri+ 2013). *VizieR Online Data Catalog*, 2328:0, January 2014. 12

[2] R. M. Cutri, M. F. Skrutskie, S. van Dyk, C. A. Beichman, J. M. Carpenter, T. Chester, L. Cambresy, T. Evans, J. Fowler, J. Gizis, E. Howard, J. Huchra, T. Jarrett, E. L. Kopan, J. D. Kirkpatrick, R. M. Light, K. A. Marsh, H. McCallon, S. Schneider, R. Stiening, M. Sykes, M. Weinberg, W. A. Wheaton, S. Wheelock, and N. Zacarias. VizieR Online Data Catalog: 2MASS All-Sky Catalog of Point Sources (Cutri+ 2003). *VizieR Online Data Catalog*, 2246:0, June 2003. 12

[3] ESO. *HAWK-I Pipeline Manual*, 2.5.20edition, 2015. VLT-MAN-ESO-XXXXX-XXXX. 8, 9, 12, 19, 28, 31

[4] Forchì V. *Reflex User's Manual*. ESO/SDD/DFS, 0.7 edition, 2012. VLT-MAN-ESO-19000-5037. 18, 31, 47, 48

[5] S. Zampieri and V. Forchì. *OCA User Manual*. ESO. VLT-MAN-ESO-19000-4932. 28