

European Organisation for Astronomical Research in the Southern Hemisphere

Programme: GEN

Project/WP: Science Operation Software Department

Reflex ESOTK Tutorial

Document Number: ESO-XXXXXX

Document Version: 0.9.7

Document Type: Manual (MAN)

Released on: 2024-05-15

Document Classification: Public

Prepared by: ESOTK Pipeline Team

Validated by:

Approved by:

Name

This page was intentionally left blank

Authors

Name	Affiliation
Mark Neeser	ESO
Andrea Modigliani	ESO
Armin Gabasch	ESO

This page was intentionally left blank

Change Record from previous Version

Affected Section(s)	Changes/Reason/Remarks
None	None

This page was intentionally left blank

Contents

1 Introduction to `Esoreflex` 9

2 An Introduction to Fringe Correction 10

3 Fringe-Correction Method 11

3.1 Principles and Algorithms to Characterise Fringes . . . 11

3.2 Estimating the Background and the Fringe Amplitudes . . . 12

4 Software Installation 14

4.1 Installing `Esoreflex` workflows via `macports` . . . 14

4.2 Installing `Esoreflex` workflows via `rpm/yum/dnf` . . . 14

4.3 Installing `Esoreflex` workflows via `install_esoreflex` . . . 14

5 Demo Data 16

6 Quick Start: Reducing The Demo Data 17

7 About the main `esoreflex` canvas 21

7.1 Saving And Loading Workflows . . . 21

7.2 Buttons . . . 21

7.3 Workflow States . . . 21

8 The ESOTK Workflow 22

8.1 Workflow Canvas Parameters . . . 22

8.2 Workflow Actors . . . 23

8.2.1 Simple Actors . . . 23

8.2.2 Data Organisation And Selection . . . 23

8.2.3 DataSetChooser . . . 24

8.2.4 The ProductExplorer . . . 25

8.3 Workflow Details . . . 27

8.3.1 Data Organisation and Selection Actors . . . 27

8.3.2 Editing Recipe Parameters . . . 28

8.3.3 Output Organisation 28
8.4 Interactive Windows 29
8.4.1 Create Object Mask (`esotk_mask_create`) 30
8.4.2 Compute Fringes (`esotk_masterfringe_create`) 31
8.4.3 Subtract Fringe (`esotk_masterfringe_correct`) 33
8.4.4 Lazy Mode 34
8.5 Optimising Your Results 35
8.5.1 Optimising Results Through Workflow Interaction 35

9 Frequently Asked Questions **38**

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 environmet, 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 An Introduction to Fringe Correction

The routines described in this tutorial were created during a collaboration between ESO and Austrian universities, with the goal of developing instrument-independent software to improve data processing for ESO operations. A detailed description of the fringe-correction algorithms and tests confirming how well they work is given in Neeser et al. (A&A, 2020, *in prep*). Please reference this article if you use this Reflex workflow in a scientific publication.

Fringes are interference patterns caused by multiple internal reflections within unevenly thinned charge-coupled devices (CCDs). Owing to the specific thickness of visual-band CCDs, fringing will occur in the red end of the detector sensitivity scale (i.e. most relevant in the *i* and *z* bands). Fringing is apparent in virtually all ESO instruments operating in the visual wavelength range, but is probably most apparent in the UV-sensitized e2v detectors of *OmegaCAM* and *VIMOS*. In particular, an *OmegaCAM* image taken in the z'_{SDSS} (Sloan *z*-band) filter is among the worst fringe-affected images of all ESO instruments. An example of this is given in Fig. 2.1.

A unique algorithm, based on bimodal Gaussian profiles is used to distinguish the background from the fringe pattern, and allows for a very effective means for subtracting the fringing from any astronomical image. Our method works for single or multiple astronomical images and scales the derived fringe map for subtraction.

For the best results, the routines described in this tutorial require input data that has been calibrated to a basic level. This means data that has been bias-corrected, dark-corrected, and flat-fielded. Although not limited to data reduced to this level, our experience shows that the background and fringe pattern can best be distinguished when the image has had its instrumental signatures removed. Furthermore, our method works

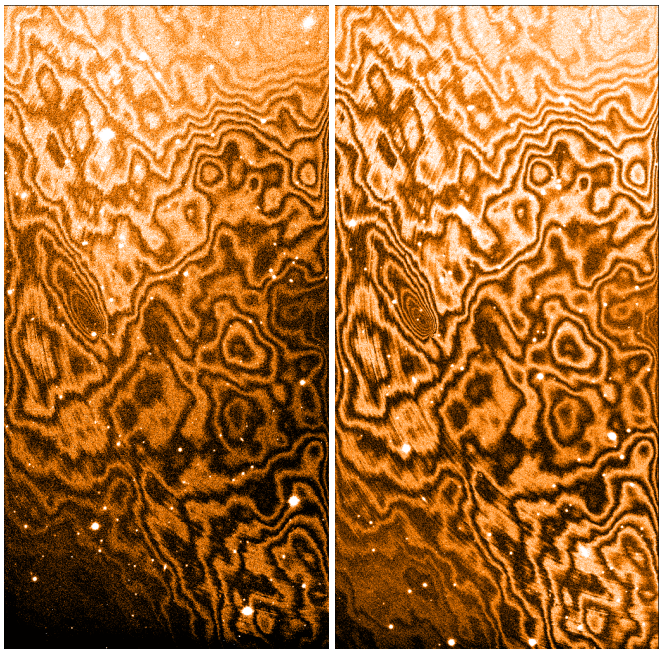


Figure 2.1: Two raw images of the same detector (ESO_CCD_#76) in *OmegaCAM* obtained in the z'_{SDSS} filter and shown at two different epochs (left: June 2013; right: May 2014). The very prominent wood grain-like fringing pattern dominates these images. The two images are similarly scaled from $\mu - \sigma$ to $\mu + 3\sigma$, and the temporal stability of the fringe pattern is apparent over long time ranges; only the relative intensity of the pattern varies with time.

best for images with discrete sources. If the image contains a large extended source, or if these algorithms are used to correct a long-slit spectra of fringing, then the extended feature must be masked using a dedicated mask image (`FRINGE_MASK`).

3 Fringe-Correction Method

3.1 Principles and Algorithms to Characterise Fringes

Assuming the input data to be calibrated to a basic level (i.e. de-biased, dark-corrected, and flat-fielded), we treat the fringing as a simple additive effect of the form:

$$D_i(x, y) = I_i(x, y) + F_i(x, y), \tag{1}$$

where D_i is the i^{th} input image, F_i is the fringe pattern in the image, and I_i is the image without fringing (consisting of a smooth sky background B_i and the flux O_i from astronomical objects), which is what we want to recover. The method proposed here is similar to the background subtraction method for near-infrared images. The fringe pattern is isolated by stacking a number of dithered images pixel-by-pixel, using a robust estimator of the mean to reject flux from objects in the field. Object masks can be created from first-pass fringe-corrected images to improve the object rejection in a second pass.

In contrast to the near-infrared background which changes its structure on short time scales, fringe patterns tend to be stable over long periods of time. This is evident in Fig. 2.1 in which two *OmegaCAM* z'_{SDSS}-band exposures, separated by almost one year, show virtually identical fringe patterns. To construct a fringe-correction image, one can therefore use images from several nights in an observational programme or even images covering different sky regions in order to obtain a high signal-to-noise ratio in the fringe-correction image. It is, therefore, assumed that all the input images to the method show the same fringe pattern $F(x, y)$. However, the amplitude a_i of the pattern will certainly change from image to image, and may even vary in intensity across the fringe pattern (e.g. due to the varying strength of the night-sky emission lines):

$$D_i = I_i + a_i F. \tag{2}$$

We now describe a method to estimate the background level $\mu_i = \overline{B_i} \simeq \overline{I_i}$ (over-lines indicate the average or median taken over one image) and the fringe level $\mu_i + a_i$ automatically for the individual images. The fringe pattern can be estimated as:

$$\hat{F} = \left\langle \frac{D_i - \mu_i}{a_i} \right\rangle_{\text{rob}} \tag{3}$$

where $\langle\,\rangle_{\text{rob}}$ indicates a pixel-wise robust mean over all input images i. The fringe-corrected images are then obtained by subtracting the scaled fringe correction from the input images:

$$\hat{I}_i = D_i - a_i \hat{F}. \tag{4}$$

3.2 Estimating the Background and the Fringe Amplitudes

To motivate the method to estimate the amplitude a of the fringe pattern in an image, we make the simplifying assumption that the image has only two values: the sky background μ_b and the fringes $\mu_f = \mu_b + a$ (this approximation assumes bright objects to have been detected and masked before hand). To this pattern, noise is added with dispersion σ_b and σ_f. As long as the images are dominated by photon noise, the noise can be reasonably well described as following a Gaussian distribution. We therefore model the pixel intensity distribution in a given image as a mixture of two Gaussian distributions, whose means are the background and the fringe amplitudes, respectively. Thus the density function $I(x)$ of the intensity of an individual pixel is modelled as follows:

$$I(x) = c_b \, \mathrm{e}^{-\frac{(x-\mu_b)^2}{2\sigma_b^2}} + c_f \, \mathrm{e}^{-\frac{(x-\mu_f)^2}{2\sigma_f^2}} \tag{5}$$

The means μ_b and μ_f (such that $\mu_b < \mu_f = \mu_b + a$) are proportional to the background level and the fringe pattern level, respectively. These values are used for normalisation of the background and fringe levels during median stacking. We refer to this procedure as the weighted median stacking. The parameters of the two Gaussian components are estimated from the density function of the pixel intensities by a nonlinear least squares fit algorithm. The algorithm requires as its input an estimated density function. Such an estimate is calculated in a preprocessing step as a truncated Hermite series:

$$I(x) \simeq \sum_{n=0}^{p} c_n \, h_n \left(\frac{x-\mu}{\sigma} \right), \tag{6}$$

where h_n is the normalized Hermite function

$$h_n(x) = \pi^{-\frac{1}{4}} \, 2^{-\frac{n}{2}} \, (n!)^{-\frac{1}{2}} \, (-1)^n \, \mathrm{e}^{\frac{x^2}{2}} \frac{d^n}{dx^n} \left(\mathrm{e}^{-x^2} \right), \tag{7}$$

and μ and σ are, respectively, the sample mean and the sample standard deviation of pixel intensities in the given image. The truncation parameter p is found experimentally, with a value of $p = 20$ giving the best results. The Hermite coefficients c_n are computed as follows:

$$c_n = \frac{1}{\sigma N} \sum_{i=1}^{N} h_n \left(\frac{I_i - \mu}{\sigma} \right), \tag{8}$$

where the summation extends over all pixel intensities
$I_1, \ldots, I_N$, N is the total number of pixels, and $n = 0, \ldots, p$.
Fig. 3.1 illustrates this Gaussian mixture and its approximation by a truncated Hermite series.

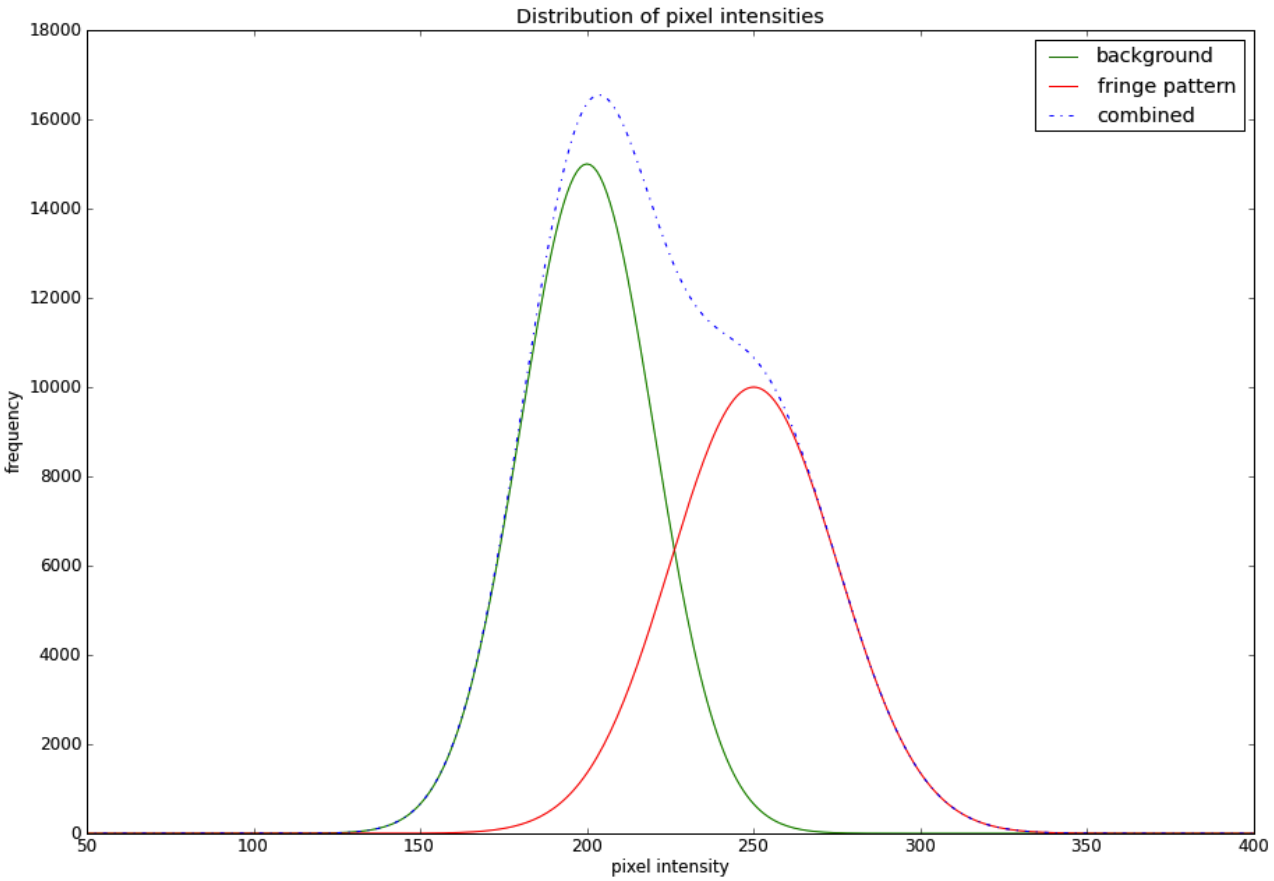


Figure 3.1: A schematic demonstrating how the routines separate the background (mean/dispersion = $\mu_b \pm \sigma_b$) from the fringe pattern (mean/dispersion = $\mu_f \pm \sigma_f$) with $\mu_b < \mu_f = \mu_b + a$, where a is the fringe pattern amplitude.

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.

Reflex 2.11.x needs java JDK 11 to be installed.

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 what 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 2.11` may be found at:
https://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.11` 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 Demo Data

The `Reflex ESOTK` workflow and pipeline kit comes with a set of demonstration data. This data is intended to be used as a means to become familiar with the pipeline and workflow. The data was selected to show the typical steps to reduce images affected by strong fringe patterns. Data taken with an uncommon or complex observing strategy may not be suitable for a straight-forward reduction with the `Reflex` workflow. In this case, a deeper understanding of the pipeline may be required.

The included fringe-correction demonstration data consists of z'_SDSS band *OmegaCAM* images that have been bias, dark, and flat-field corrected. This data consists of a small subset from the Hercules Supercluster Survey (PROG.ID = 093.A-0717(B)), obtained using the *OmegaCAM* instrument mounted on the VLT Survey Telescope (VST) at ESO's Cerro Paranal Observatory, Chile. In the interest of keeping the size of the data download modest, only four single CCD's of the 32 *OmegaCAM* detectors have been included. It should be noted, however, that the `Reflex ESOTK` workflow will work with both single and multi-extension files (MEF). This data was chosen because the UV-sensitised e2v detectors of *OmegaCAM* are amongst the worst affected by fringing.

Table 5.1: `Reflex ESOTK` Workflow *OmegaCAM* Demonstration Data Set

File	HIERARCH ESO PRO CATG	HIERARCH INS FILT1 NAME
corr_OMEGA.2014-05-11T02:20:09.900_1.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:20:09.900_2.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:20:09.900_3.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:20:09.900_4.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:30:54.661_1.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:30:54.661_2.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:30:54.661_3.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:30:54.661_4.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:42:50.213_1.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:42:50.213_2.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:42:50.213_3.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:42:50.213_4.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:55:34.905_1.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:55:34.905_2.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:55:34.905_3.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T02:55:34.905_4.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T03:06:20.236_1.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T03:06:20.236_2.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T03:06:20.236_3.fits	RAW	z_SDSS
corr_OMEGA.2014-05-11T03:06:20.236_4.fits	RAW	z_SDSS
err_OMEGA.2014-05-11T02:20:09.900_1.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:20:09.900_2.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:20:09.900_3.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:20:09.900_4.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:30:54.661_1.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:30:54.661_2.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:30:54.661_3.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:30:54.661_4.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:42:50.213_1.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:42:50.213_2.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:42:50.213_3.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:42:50.213_4.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:55:34.905_1.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:55:34.905_2.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:55:34.905_3.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T02:55:34.905_4.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T03:06:20.236_1.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T03:06:20.236_2.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T03:06:20.236_3.fits	RAW_ERROR	z_SDSS
err_OMEGA.2014-05-11T03:06:20.236_4.fits	RAW_ERROR	z_SDSS
OMEGA_bpm_1.fits	RAW_BPM	—
OMEGA_bpm_2.fits	RAW_BPM	—
OMEGA_bpm_3.fits	RAW_BPM	—
OMEGA_bpm_4.fits	RAW_BPM	—
OMEGA_bpm_5.fits	RAW_BPM	—
OMEGA_fringe_mask.fits	FRINGE_MASK	—

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 ESOTK demo data set supplied with the `esoreflex 2.11` 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 `esotk` by typing:

   ```
   esoreflex esotk&
   ```

   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 `esotk` 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[2]. 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 8.2.3.

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 8.2.4.

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 ESOTK workflow that merit a look at the rest of this tutorial.

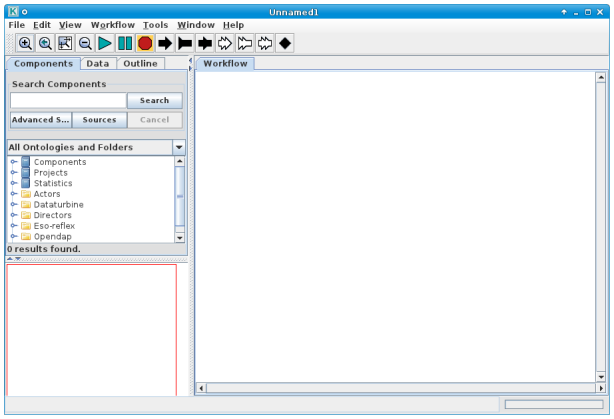


Figure 6.1: *The empty* Reflex *canvas.*

[2]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.

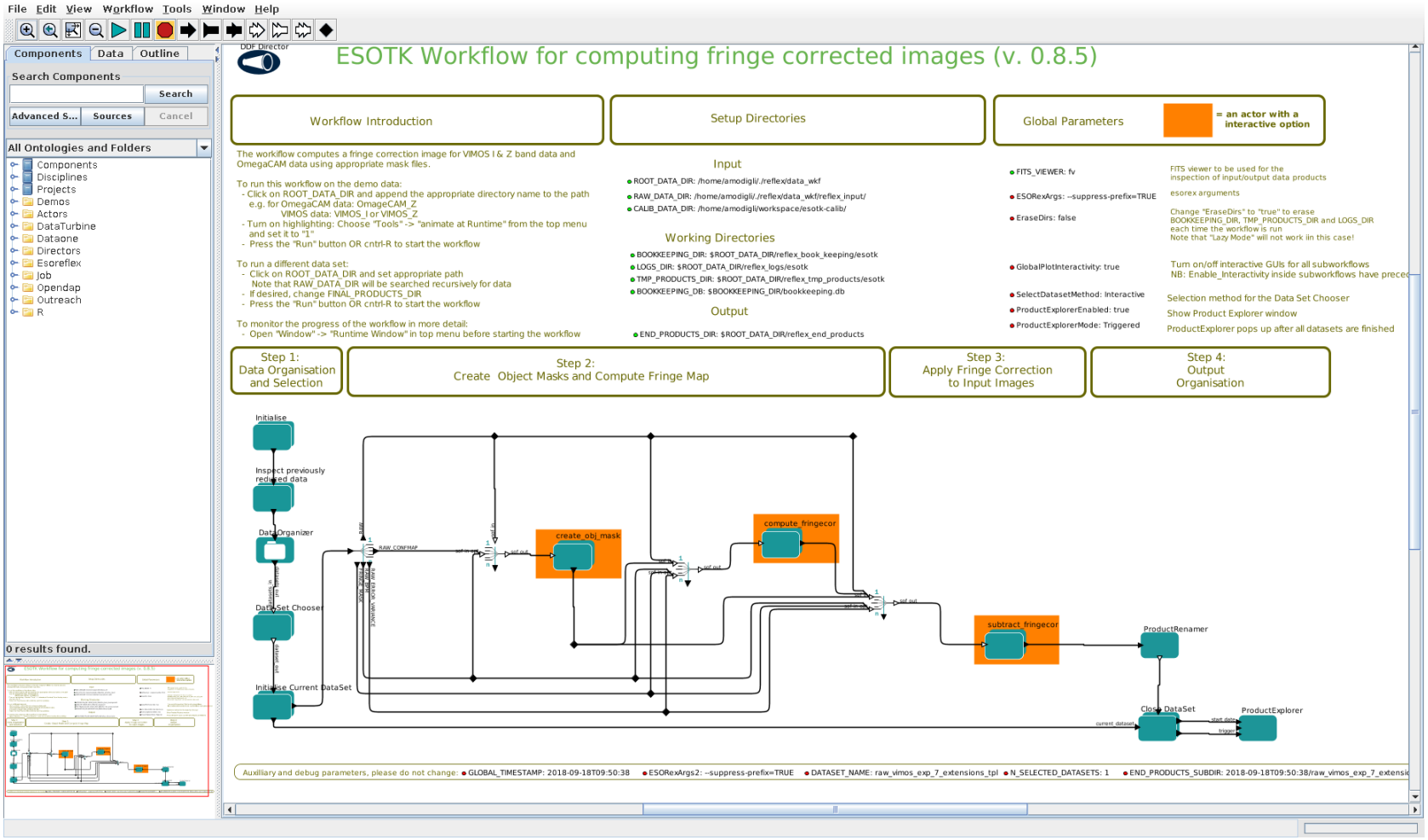


Figure 6.2: The ESOTK workflow general layout.

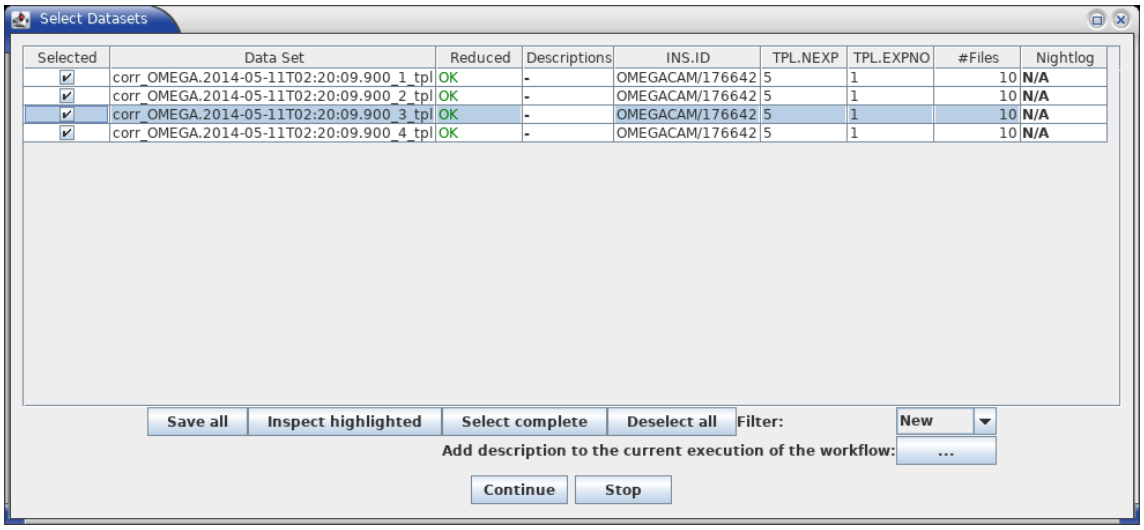


Figure 6.3: *The Select Datasets pop-up window.*

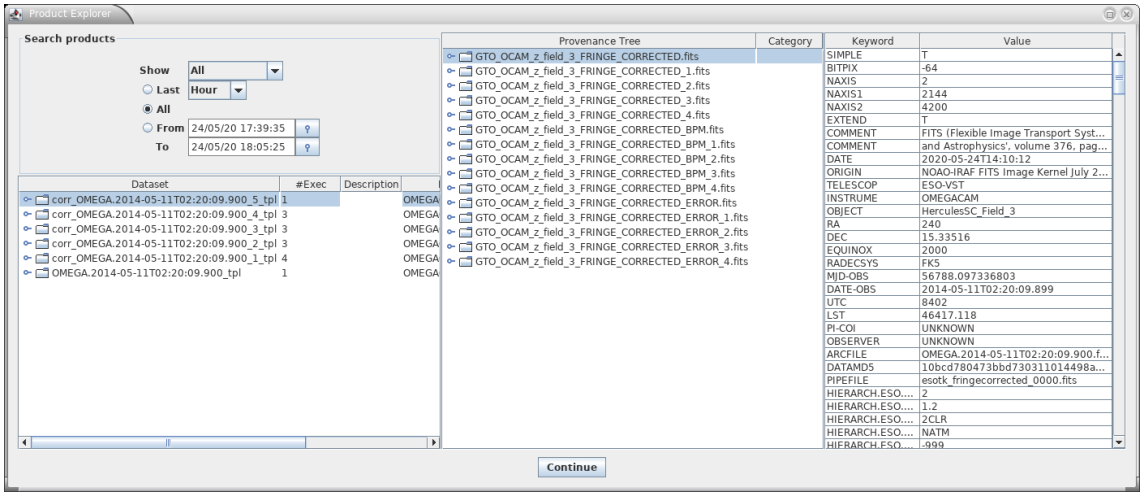


Figure 6.4: *The Product Explorer shows all datasets reduced in previous executions together with the full reduction chain for all the pipeline products..*

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 ESOTK Workflow

The ESOTK 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; [2]).

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 8.4.4), 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 Produc-

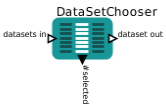
tExplorer GUI at the end of the reduction of each individual dataset. `Triggered` (default and recommended) opens the ProductExplorer GUI when all the selected datasets have been reduced. `Disabled` does not display the ProductExplorer GUI.

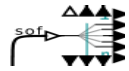
8.2 Workflow Actors

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 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[3] 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 8.2.3.

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.

8.2.3 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

[3] 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`

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 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[4], 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.

8.2.4 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:

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

- 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:
  - "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.

8.3 Workflow Details

This section describes the inner workings of the `Reflex ESOTK` workflow. It is intended for those Users who wish to delve deeper into the workflow in order to diagnose errors, understand the source of unexpected behaviour, or make modifications to the workflow.

8.3.1 Data Organisation and Selection Actors

The `Data Organiser` actor uses a special set of so-called OCA 'rules' to Organise, Classify, Associate, and define purposes for data files. These rules are defined in a file called `esotk.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. However we recommend this only to expert users as changes in the OCA rules often implies also changes in the workflow and a knowledge of their syntax. Please refer to the OCA User Manual [3] for details on the syntax and implementation of rules in an .oca file.

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.

The OCA rules of the fringe correction workflow will group together data with the same value of the primary header FITS keywords `INS.FILT1.NAME, OBS.ID, OBS.NAME, OBS.TARG.NAME, INSTRUME, NAXIS, TPL.START`.

The instrument setup information provided by the Data Set Chooser, in case of the fringe correction workflow is given by the following FITS keywords: `INS.ID, TPL.NEXP, TPL.EXPNO`. This can be customised by the user by clicking with the right hand side of the mouse on the Data Organiser actor at the left of the workflow layout and editing its configuration.

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 object mask, and compute the fringe correction master, and c) at least one raw frame on which the fringes will be corrected. 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 OBJ_MASK". 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.

8.3.2 Editing Recipe Parameters

Most processing can be done using the parameter default values. 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 8.1). This method of changing recipe parameters is *not* recommended. In the actors involved in the fringe correction workflow, all 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.

8.3.3 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

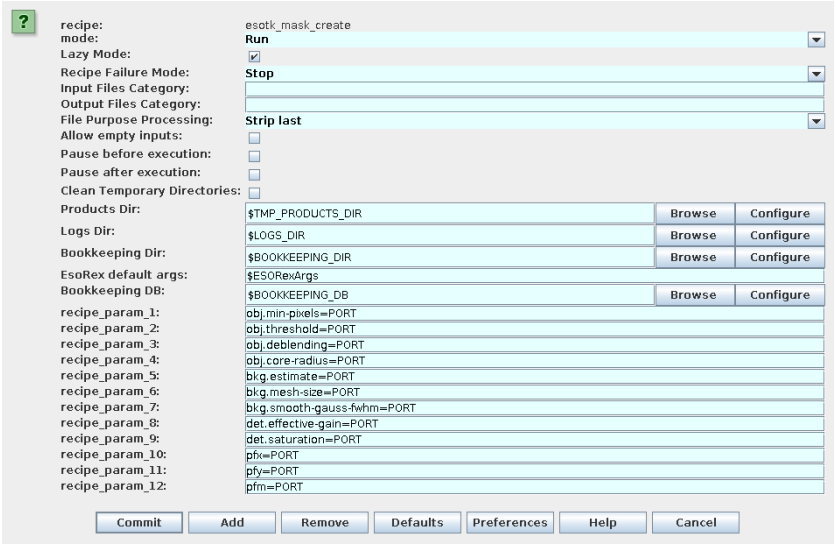


Figure 8.1: *Example window for modifying fringe mask creation parameters.*

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 incremental 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 ESOTK Pipeline Manual [1] for a description of the pipeline products and associated `HIERARCH.ESO.PRO.CATG` values.

8.4 Interactive Windows

The ESOTK workflow contains three 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[2] 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.

8.4.1 Create Object Mask (`esotk_mask_create`)

This routine detects the discrete sources in the input images and masks them. To ensure that the masks adequately cover the sources their sizes are increased by growing the detection radius by a user-defined number of pixels. Aside from the input images themselves, an optional static mask can be provided to define the areas of the input images that are used for defining the source detection threshold.

The interactive window for this recipe (Figure 8.2) opens by displaying two images. The input image with fringes is shown on the left, while the mask of the detected sources is shown on the right. At the bottom of the panel the user may choose the filename and the extension number (when the input images are multi-extension format) of the displayed images, respectively, via a menu button and a slider. Next to the filename chooser and slider are corresponding Previous and Next buttons to allow a sequential selection. The image extension identity is also shown at the top of the left panel. Moving the mouse on the images will show an explanatory text.

Hovering the mouse over the button or the round indicator of the slider will show explanatory text and allow the user to change the selection; note that there may be a short delay between clicking and seeing the displayed image change.

On the rightmost end of the panel, the controlling recipe parameters are grouped under tabs. Clicking with the mouse pointer on each tab the user can select the corresponding recipe parameters and change their value. Hovering the mouse pointer over each parameters shows an explanatory text.

The user can set the following parameter options for this recipe (see Table 8.1)

Table 8.1: Parameters for Create Object Mask

Parameter	Default Value	Description
background:		
bkg.estimate	True	Estimate background from input. If false it is assumed input is already background corrected with median 0.
bkg.mesh-size	64	Background smoothing box size.
bkg.smooth-gauss-fwhm	2	The FWHM of the Gaussian kernel used in convolution for object detection.
detector:		
det.effective-gain	1	Detector gain value to rescale convert intensity to electrons.
det.saturation	65000	Detector saturation value.
object:		
obj.core-radius	5	Value of R_{core} in pixels.
obj.deblending	True	Use de-blending?
obj.min-pixels	5	Minimum pixel area for each detected object.
obj.threshold	3	Detection threshold in sigma above sky.
post-filtering:		
pfm	closing	Post filtering mode [closing/dilation].
pfx	3	X size of the post filtering kernel.
pfy	3	Y size of the post filtering kernel.

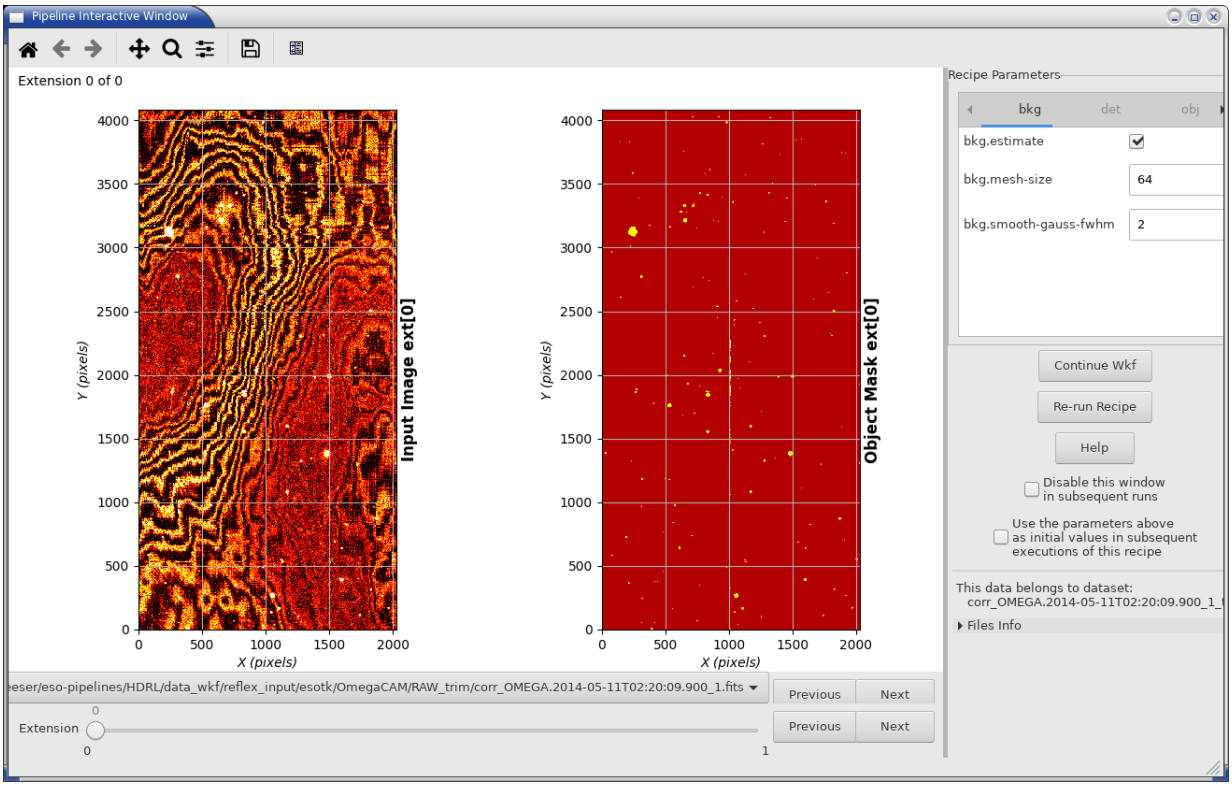


Figure 8.2: Example interactive window for creating an object mask.

8.4.2 Compute Fringes (`esotk_masterfringe_create`)

This routine detects the fringing via multiple Gaussian fits to the trough and peak of the fringe pattern, creates a master fringe map, and determines the relative scaling for each of the input images. This routine uses the source masks created by `esotk_mask_create` to remove the sources from the fringe background determination.

An optional error map can be given for each of the input images. Further options include the use of a bad-pixel map and/or a fringe mask through which the measurement of the fringe amplitude can be restricted to specified parts of the detector.

The interactive window for this recipe (Figure 8.3) opens by displaying two images. The left panel shows the original input image, which the right panel shows the computed master fringe map. At the bottom of the panel the user may choose the filename and the extension number (when the input images are multi-extension format) of the displayed images, respectively, via a menu button and a slider. Next to the filename chooser and slider are corresponding Previous and Next buttons to allow a sequential selection. The image extension identity is also shown at the top of the left panel. Moving the mouse on the images will show an explanatory text.

Hovering the mouse over the button or the round indicator of the slider will show explanatory text and allow the user to change the selection; note that there may be a short delay between clicking and seeing the displayed image change.

On the rightmost end of the panel, the controlling recipe parameters are grouped under tabs. Clicking with the mouse pointer on each tab the user can select the corresponding recipe parameters and change their value.

Hovering the mouse pointer over each parameters shows an explanatory text.

The user can set the following parameter options for this recipe (see Table 8.2)

Table 8.2: Parameters for Compute Fringes

Parameter	Default Value	Description
collapse:		
collapse.method	MEDIAN	Method used for collapsing the data: [MEAN/WEIGHTED_MEAN/MEDIAN/SIGCLIP/MINMAX]
collapse.minmax.nhigh	1	High number of pixels to reject for the minmax clipping algorithm.
collapse.minmax.nlow	1	Low number of pixels to reject for the minmax clipping algorithm.
collapse.sigclip.kappa-high	3	High kappa factor for kappa-sigma clipping algorithm.
collapse.sigclip.kappa-low	3	Low kappa factor for kappa-sigma clipping algorithm.
collapse.sigclip.niter	5	Maximum number of clipping iterations for kappa-sigma clipping.

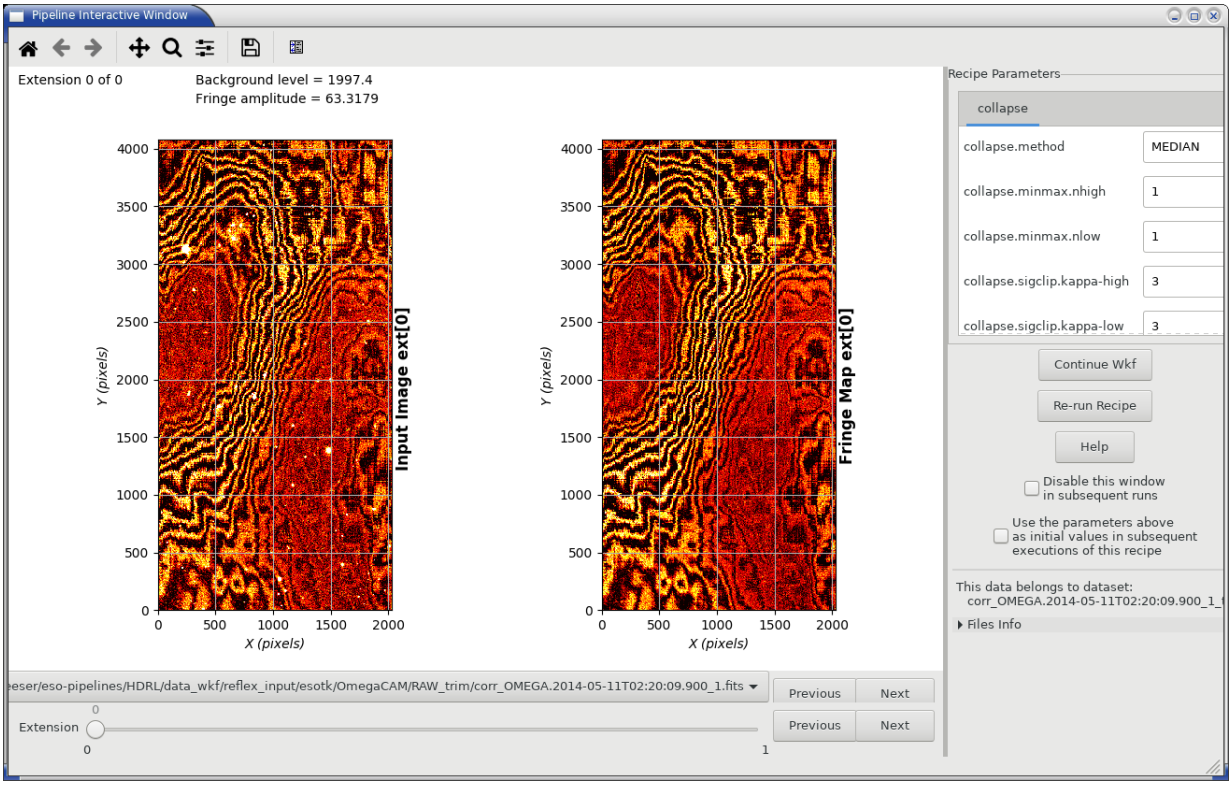


Figure 8.3: Example interactive window for computing the fringes.

8.4.3 Subtract Fringe (`esotk_masterfringe_correct`)

This routine subtracts the master fringe map from each of the input images. The amplitude of the fringes is computed for each input image and used to rescale the correction image before subtraction. This routine uses the master fringe map created by `esotk_masterfringe_create`. An optional error map can be given for each of the input images. Further options include the use of an optional bad-pixel map and/or a fringe mask by which the measurement of the fringe amplitude can be restricted to specified parts of the chip.

The interactive window for this recipe (Figure 8.4) shows opens by displaying three images. The left panel shows the original input image, the centre panel shows the computed fringe mask, and the right panel shows the input image corrected with a scaled fringe mask. At the bottom of the panel the user may choose the filename and the extension number (when the input images are multi-extension format) of the displayed images, respectively, via a menu button and a slider. Next to the filename chooser and slider are corresponding Previous and Next buttons to allow a sequential selection. The image extension identity is also shown at the top of the left panel. QC parameters computed on the input image and the final fringe-subtracted product (the background level and the fringe amplitude) are shown above these two images. Moving the mouse on the images will show an explanatory text.

Hovering the mouse over the button or the round indicator of the slider will show explanatory text and allow the user to change the selection; note that there may be a short delay between clicking and seeing the displayed image change.

The user can set the following parameter options for this recipe (see Table 8.3)

Table 8.3: Parameters for Compute Fringes

Parameter	Default Value	Description
rescale	true	measure fringe amplitude and rescale fringe correction before subtracting?

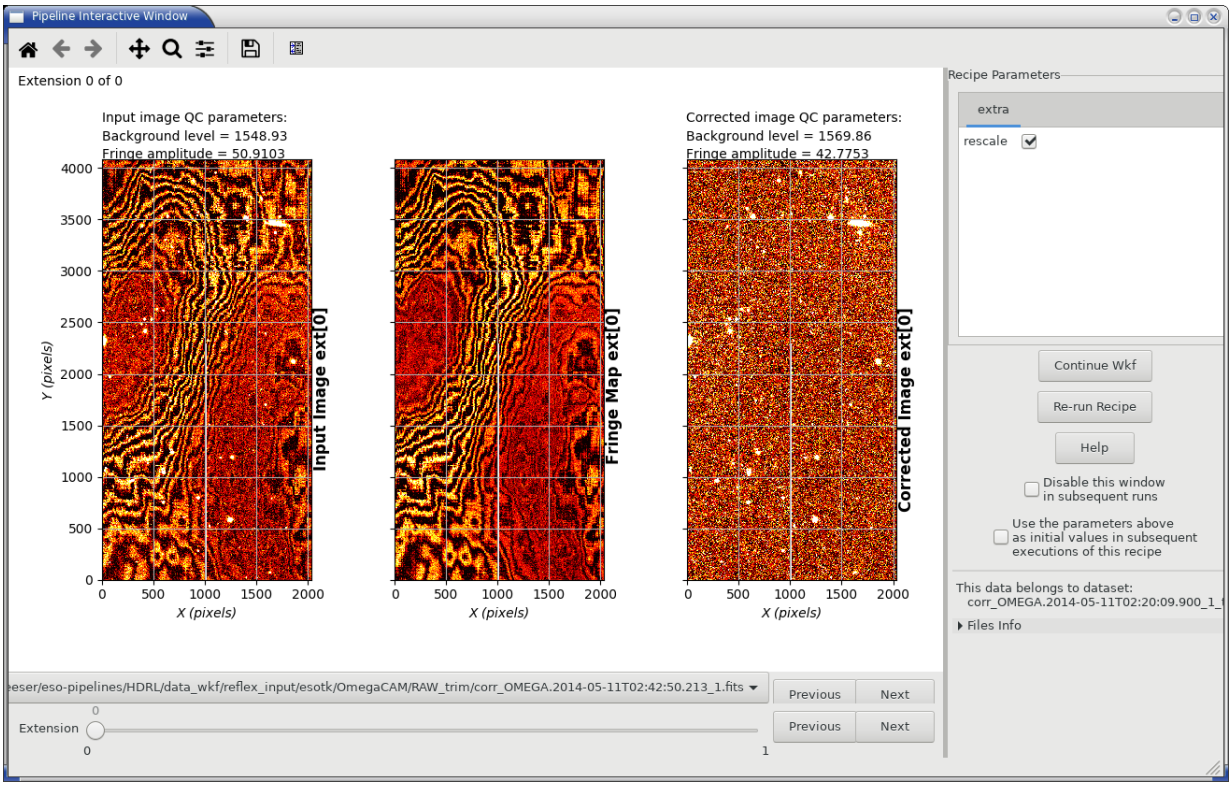


Figure 8.4: Example interactive window for checking the results after fringe-correction.

8.4.4 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 makes the workflow to crash.*

8.5 Optimising Your Results

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`.

8.5.1 Optimising Results Through Workflow Interaction

In this section we will describe a number of possible fringe-correction issues that a User may encounter.

If the fringe pattern is very strong and/or contains steep gradients, the source detection routine may confuse fringes with discrete sources. If this occurs, it will be evident in the interactive window of the `esotk_mask_create` recipe; you will see fringe patterns masked as if they were sources. An example of this is given in Figure 8.5. If this occurs, it can be corrected by increasing the `obj.threshold` parameter from its default value of 3. Alternatively, the `bkg.mesh-size` can be increased.

If the input data frame has strong background gradients or discontinuous edges, this can hinder the determination of the fringe pattern and will cause either a fringe over- or under-estimate. This will become apparent in the final `esotk_masterfringe_correct` recipe when the fringe pattern has been either over-subtracted or under-subtracted. An example of this is shown in Figure 8.6 in which the fringe pattern has been underestimated. To remedy such a case, the User should either trim the discontinuous edges from the image, or create a mask having the `HIERARCH ESO PRO CATG` label `FRINGE_MASK`. Such a mask should have a value of 0 in all regions where the fringe pattern should be determined, and 1 in all regions that should be ignored (i.e. those of the discontinuous edges).

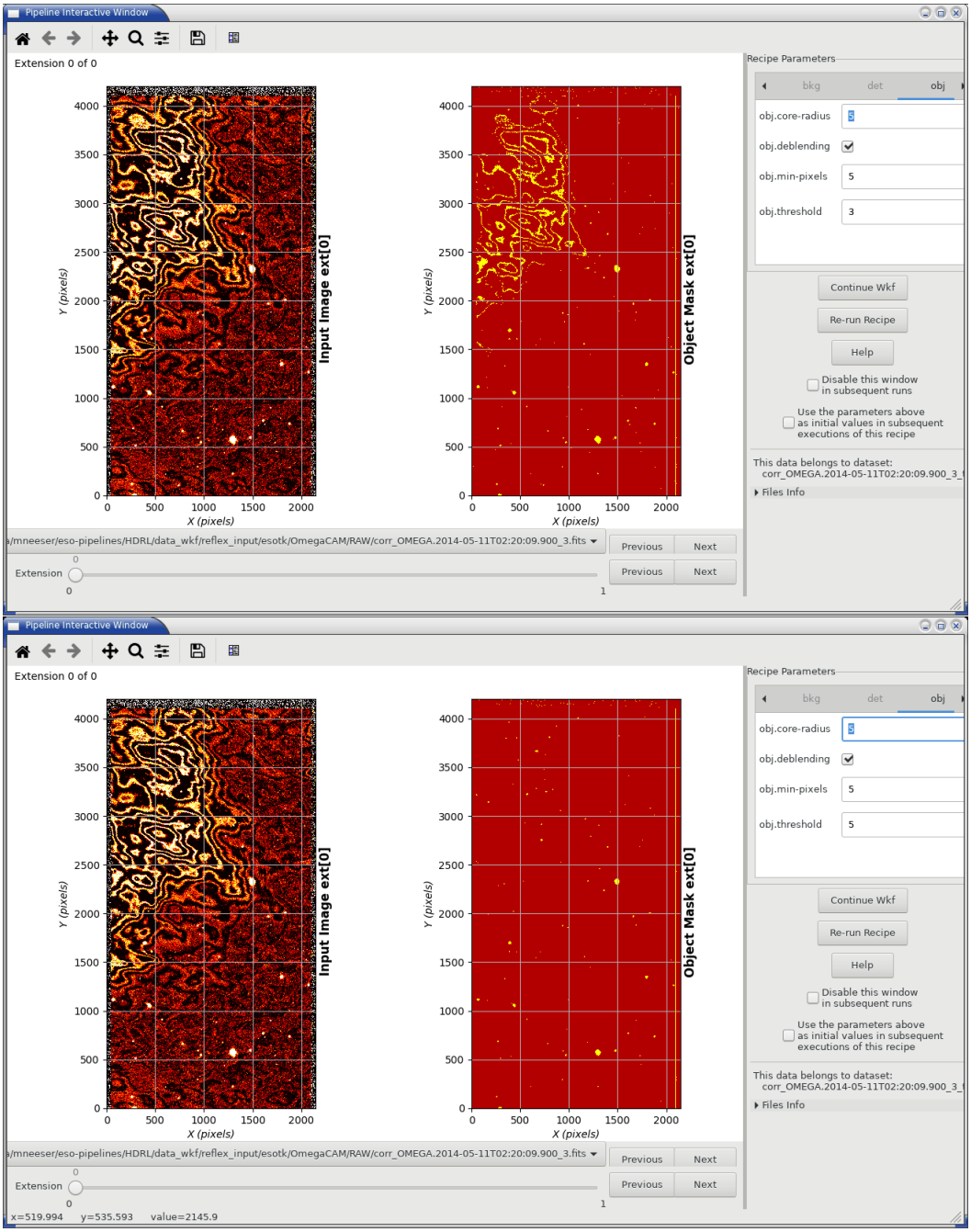


Figure 8.5: *Upper Panel:* An example of a strong fringe pattern being confused with astronomical sources. In this iteration of the workflow, the `obj.threshold` parameter was set to its default value of 3.
Lower Panel: In a second iteration, the `obj.threshold` parameter was set to 5 and the Re-run Recipe button was pressed.

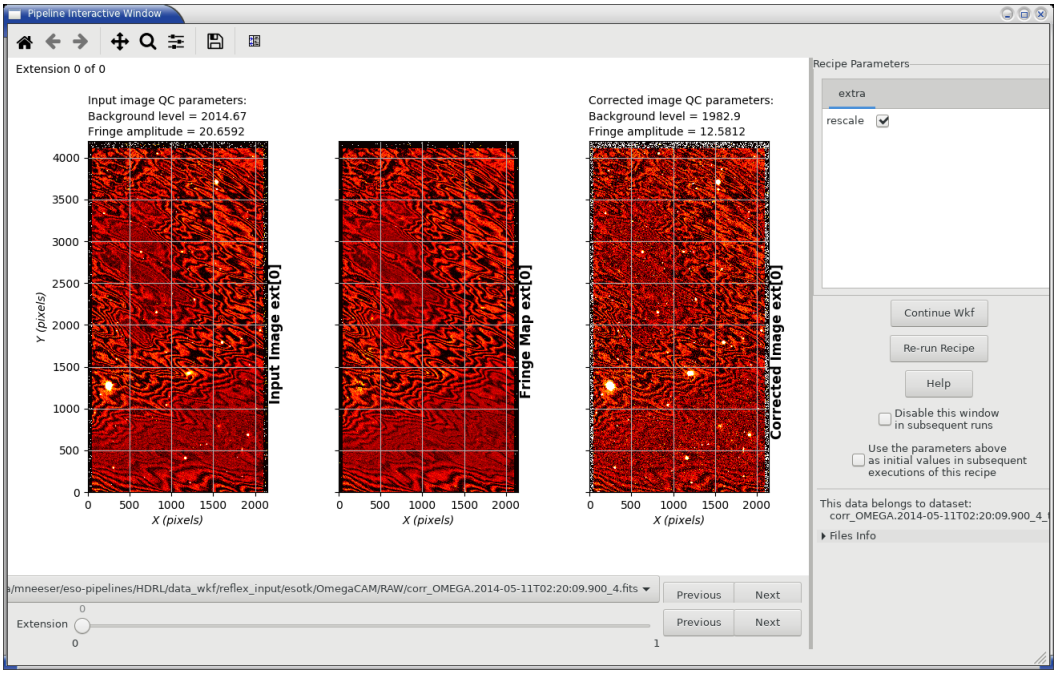


Figure 8.6: An example of an input image with strong discontinuous edges. In the results shown for the interactive window for the `esotk_masterfringe_correct` recipe the fringe pattern has been inadequately subtracted.

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 ([2]). 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 ([2]) 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 ([2]) 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`

[1] ESO. *ESOTK Pipeline Manual*, 0.9.7edition, 2020. VLT-MAN-ESO-XXXXX-XXXX. 29

[2] Forchì V. *Reflex User's Manual*. ESO/SDD/DFS, *http://www.eso.org/gasgano/*, 0.7 edition, 2012. VLT-MAN-ESO-19000-5037. 22, 29, 38, 39

[3] S. Zampieri and V. Forchì. *OCA User Manual*. ESO. VLT-MAN-ESO-19000-4932. 27