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

Mime Pipeline User Manual

Issue 0.9

Date 28 February 2011

Prepared:	`O. Czoske, W. Zeilinger, S. Das, T. Hrycak`	`not yet`	
	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
0.1	13/12/2010	All	Layout initialised
0.5	10/01/2011	All	acceptance release
0.9	28/02/2011	Sect. 8, A	pre-final release

This page was intentionally left blank

Contents

1 Introduction

1.1 Purpose

The Multiobject Image Mosaicing pipelinE (MIME) is a collection of instrument-independent recipes that convert the basic calibrated products of instrument pipelines into final scientifically usable products.

This manual gives a complete description of the imaging data reduction recipes reflecting the status of the MIME pipeline as of 28 February 2011 (mime-0.2.0).

1.2 Acknowledgments

The MIME pipeline is based on the Common Pipeline Library (CPL) developed by the ESO/SDD/PSD.

1.3 Scope and references

This document describes the MIME pipeline.

Updated versions of the present document can be found on [20]. Quality control information is at [8].

Additional information on the Common Pipeline Library (CPL), *Esorex* and *Gasgano* can be found at [5], [4], [11], [12], and [13].

1.4 Reference and applicable documents

[1] I. Appenzeller, K. Fricke, W. Fürtig, W. Gässler, R. Häfner, R. Harke, H.-J. Hess, W. Hummel, P. Jürgens, R.-P. Kudritzki, K.-H. Mantel, W. Meisl, B. Muschielok, H. Nicklas, G. Rupprecht, W. Seifert, O. Stahl, T. Szeifert, and K. Tarantik. Successful commissioning of FORS1 - the first optical instrument on the VLT. *ESO Messenger*, 94:1–6, December 1998. 14

[2] D. Baade, K. Meisenheimer, O. Iwert, J. Alonso, T. Augusteijn, J. Beletic, H. Bellemann, W. Benesch, A. Böhm, H. Böhnhardt, J. Brewer, S. Deiries, B. Delabre, R. Donaldson, C. Dupuy, P. Franke, R. Gerdes, A. Gilliotte, B. Grimm, N. Haddad, G. Hess, G. Ihle, R. Klein, R. Lenzen, J.-L. Lizon, D. Mancini, N. Münch, A. Pizarro, P. Prado, G. Rahmer, J. Reyes, F. Richardson, E. Robledo, F. Sanchez, A. Silber, P. Sinclaire, R. Wackermann, and S. Zaggia. The Wide Field Imager at the 2.2-m MPG/ESO telescope: first views with a 67-million-facette eye. *ESO Messenger*, 95:15–16, March 1999. 14

[3] Åke Björck. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, 1996. 48

[4] *CPL home page*. http://www.eso.org/cpl/. 9

[5] *Common Pipeline Library User Manual*. VLT-MAN-ESO-19500-2720. 9

[6] J. G. Cuby, C. Lidman, and C. Moutou. Isaac: 18 months of paranal science operations. *ESO Messenger*, 101:2–8, sep 2000. 46

[7] Anirban DasGupta. *Asymptotic Theory of Statistics and Probability*. Springer, New York, 2008. 43

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

[9] ESO. *Data Interface Control Document*. GEN-SPE-ESO-19400-0794. 40

[10] ESO. *Reflex User Manual*. VLT-MAN-ESO-19000-5037. 15

[11] *ESOREX home page*. http://www.eso.org/cpl/esorex.html. 9, 15, 17

[12] *GASGANO home page*. http://www.eso.org/gasgano. 9

[13] *Gasgano User's Manual*. VLT-PRO-ESO-19000-1932. 9, 15

[14] M. Kissler-Patig, J.-F. Pirard, M. Casali, A. Moorwood, et al. HAWK-I: the high-acuity wide-field K-band imager for the ESO Very Large Telescope. *Astron. and Astroph.*, 491:941–950, December 2008. 13

[15] O. Le Fèvre, M. Saisse, D. Mancini, S. Brau-Nogue, O. Caputi, L. Castinel, S. D'Odorico, B. Garilli, M. Kissler-Patig, C. Lucuix, G. Mancini, A. Pauget, G. Sciarretta, M. Scodeggio, L. Tresse, and G Vettolani. Commissioning and performances of the vlt-vimos instrument. In M. Iye and A. F. M. Moorwood, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 4841 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 1670=–1681, March 2003. 13

[16] J. S. Maritz and R. G. Jarrett. A note on estimating the variance of the sample median. *Journ. Am. Stat. Assoc.*, 73:194–196, mar 1978. 43

[17] A. Moorwood, J.-G. Cuby, P. Ballester, P. Biereichel, J. Brynnel, R. Conzelmann, B. Delabre, N. Devillard, A. van Dijsseldonk, G. Finger, H. Gemperlein, C. Lidman, T. Herlin, G. Huster, J. Knudstrup, J.-L. Lizon, H. Mehrgan, M. Meyer, G. Nicolini, A. Silber, J. Spyromilio, and J. Stegmeier. ISAAC at the VLT. *ESO Messenger*, 95:1–5, March 1999. 13

[18] A. Moorwood, J.-G. Cuby, P. Biereichel, J. Brynnel, B. Delabre, N. Devillard, A. van Dijsseldonk, G. Finger, H. Gemperlein, R. Gilmozzi, T. Herlin, G. Huster, J. Knudstrup, C. Lidman, J.-L. Lizon, H. Mehrgan, M. Meyer, G. Nicolini, M. Petr, J. Spyromilio, and J. Stegmeier. ISAAC sees first light at the VLT. *ESO Messenger*, 94:7–9, December 1998. 13

[19] Ferdinando Patat. Night sky brightness during sunspot maximum at paranal. *ESO Messenger*, 115:18–21, mar 2004. 46

[20] *Pipelines Web Page*. http://www.eso.org/pipelines. 9

[21] Paul R. Rider. Variance of the median of small samples from several special populations. *Journ. Am. Stat. Assoc.*, 55:148–150, mar 1960. 43, 44

1.5 Abbreviations and acronyms

DFS	Data Flow System
FITS	Flexible Image Transport System
FORS	FOcal Reducer and low dispersion Spectrograph
HAWK-I	High Acuity Wide field K-band Imager
ISAAC	Infrared Spectrometer And Array Camera
MIME	Multiobject Image Mosaicing pipelinE
OB	Observing Block
SOF	set of frames
SOFI	Son OF ISAAC
UT	Unit Telescope
VIMOS	VIsible MultiObject Spectrograph
VLT	Very Large Telescope
WFI	Wide-Field Imager

2 Overview

With more than ten instruments permanently installed at the four UTs of the VLTs and many instruments at other telescopes, ESO's task of developing and maintaining software for the data processing and reduction is formidable. So far, all of ESO's data processing pipelines have been specific to particular instruments. Once the instrument-specific signatures in the data have been removed, however, the further processing can be done largely independently of what particular instrument the data came from. Consequently, there is some redundancy amongst the instrument pipelines.

In order to simplify the data flow system and to reduce the number of pipeline modules that have to be maintained, an instrument-independent pipeline for the reduction of imaging data beyond the basic reduction stages of bias and dark subtraction and flat-field correction was desired. The goal of MIME is to provide the modules for this purpose.

MIME has been developed at the Institute for Astronomy and the Numerical Harmonic Analysis Group (NuHAG) of the Faculty of Mathematics at the University of Vienna as part of the Austrian in-kind contribution after Austria joined ESO in July 2008.

The MIME recipes are applicable to data from a number of ESO instruments. The input data are products from the instrument-specific pipelines which apply the calibration images (bias, dark, flat-field) obtained with the instrument at the telescope during, or shortly before or after the science images were taken. MIME performs further calibration tasks that can be done from the basic calibrated data themselves or using external reference data. The tasks include estimation and subtraction of the sky background from the images, astrometric and (relative) photometric calibration, regridding of dithered exposures of the same field to a common pixel grid and stacking to the final output image. Error maps are propagated through the recipes. If no input error maps are available, MIME attempts to estimate them from the science data using a photon noise model.

Like the instrument pipelines, MIME is based on the Common Pipeline Library (CPL). CPL is a collection of C functions that perform basic reduction and image operation tasks and ensure the proper integration of recipes into the data flow system. Many utility functions have been developed within MIME to do the actual work underneath the recipe level. While not primarily aimed at developers, it is hoped that they may be useful for users who wish to develop recipes for their own particular needs.

For the astrometric calibration and regridding of the images, MIME relies on the well tested programmes `SExtractor`, `scamp` and `SWarp` from the AstroMatic (formerly Terapix) software suite developed by Emmanuel Bertin. Recipes are provided that wrap these programmes and integrate them according to the requirements of ESO's data flow system.

Apart from the software modules, MIME provides workflows for the REFLEX environment which automatize the data organisation and the flow of data through to the final products.

This manual is organized as follows: The instruments for which the MIME recipes apply are briefly described in Sect. 3. Sect. 4 provides an overview of the various interfaces that can be used to work with the MIME recipes. Also, an example reduction using ESOREX is described in some detail. The general data reduction cascade which guided the development of MIME is described in Sect. 5. Here one can find information on which data frames are needed and how they are passed from one recipe to the next. The interfaces to the recipes are described in Sect. 6 with details on the input and output data and the recipe parameters. The REFLEX workflows provided by MIME are described in Sect. 7. Details on the algorithms used in the MIME recipes are given in Sect. 8. Appendix A explains how to install the MIME recipes and workflows on the user's computer.

3 Instruments

In this section, we give a brief description of the instruments mounted at the VLT[1] and other ESO telescopes [2] for which the MIME pipeline recipes have been tested. For more details about the instruments please refer to the instrument user manuals and the ESO web pages.

3.1 ISAAC

ISAAC [18, 17] is an infrared imager and spectrograph installed at the Nasmyth A focus of UT 3. It has two arms, a short- and a long-wavelength arm (SW and LW, respectively). The MIME pipeline has been tested with imaging data from the SW arm.

The SW arm is equipped with a 1024×1024 Hawaii Rockwell array and operates at wavelengths $1 - 2.5\,\mu$m. The pixel scale is 0.148 arcsec, giving a field of view of $2.5 \times 2.5\,\mathrm{arcmin}^2$.

The instrument is offered with 21 filters: five broad-band filters (SZ, Js, J, H and Ks) and 16 narrow-band filters.

3.2 SOFI

SOFI is an infrared spectrograph and imager mounted at the Nasmyth A focus of the NTT at La Silla. The instrument is very similar to ISAAC. It is equipped with a 1024×1024 Hawaii HgCdTe detector giving a field of view of $4\overset{\prime}{.}92 \times 4\overset{\prime}{.}92$ at a pixel scale of $0\overset{\prime\prime}{.}288$ (SOFI can also be operated at a pixel scale of $0\overset{\prime}{.}144$).

3.3 HAWK-I

HAWK-I [14] is a cryogenic wide-field imager installed at the Nasmyth A focus of UT 4. The on-sky field of view is $7.5 \times 7.5\,\mathrm{arcmin}^2$ with a cross-shaped gap of 15 arcsec between the four HAWAII 2RG 2048×2048 pixel detectors. The pixel scale is 0.106 arcsec. The instrument is offered with 10 filters placed in two filter wheels: four broad-band filters (Y, J, H and K) and six narrow-band filters (Bracket gamma, CH_2, H_2, $1.061\,\mu$m, $1.187\,\mu$m and $2.090\,\mu$m).

3.4 VIMOS

VIMOS [15] is a multi-purpose optical instrument mounted at the Nasmyth B focus of UT 3, working in imaging and two spectroscopic modes (multi-object and integral-field spectroscopy). The MIME pipeline works with imaging data from VIMOS.

The detector consists of four quadrants, each with a field of view of $\sim 7 \times 8\,\mathrm{arcmin}^2$, with a cross-shaped gap of width 2 arcmin between the quadrants. Until June 2010, each quadrant was equipped with four 2048×4096 EEV 44-82, backside-illuminated single-layer CCDs (in imaging mode only 2048×2350 were used).

[1] `http://www.eso.org/sci/facilities/paranal/instruments/index.html`
[2] `http://www.eso.org/public/teles-instr/lasilla.html`

In the course of the VIMOS Improvement Project, the detectors have been replaced with four E2V 44-820-1-D42 devices of the same size which give improved red efficiency and reduced fringing. The pixel scale is 0.205 arcsec.

VIMOS is offered with six broad-band filters (U', B, V, R, I and z).

3.5 FORS2

FORS2 [1] is a multi-mode optical instrument mounted at the Cassegrain focus of UT 1. It can be operated with pixel scales of $0.''25$, $0.''125$ and $0.''0632$, giving fields of view of $6.'8 \times 6.'8$ and $4.'2 \times 4.'2$. FORS2 is equipped with a mosaic of either two red-sensitive $2k \times 4k$ MIT CCDs or two blue-sensitive $2k \times 4k$ E2V CCDs (these are the original FORS1 detectors). The standard filter set of FORS includes seven broad-band filters (u, b, v, g, R, I and z) and many medium- and narrow-band filters.

3.6 WFI

WFI [2] is a wide-field imager mounted at the Cassegrain focus of the 2.2-m MPG/ESO telescope at La Silla. It is equipped with eight $2k \times 4k$ EEV-44 CCDs arranged in two rows with gaps of $\sim 2''$ between the chips, resulting in a field of view of $34' \times 33'$ at a pixel scale of $0.''238$. The standard filter set includes broad-, medium- and narrow-band filters covering the wavelength range between 3500 Å and $\sim$ 10 000 Å.

4 Quick start

4.1 MIME pipeline recipes

mime_estimate_error_map: Estimate the rms photon noise in an image

mime_compute_bkg: Create sky correction images for NIR imaging OBs

mime_subtract_bkg: Subtract sky correction images from NIR images

mime_shift_and_add: Create a simple stacked image for the purpose of creation of an object mask

mime_create_obj_mask: Create object masks

mime_compute_smooth_bkg: Computate a polynomial estimate of the background in (optical) images

mime_run_sextractor: Run SExtractor on a set of images

mime_run_scamp: Run Scamp on a set of image catalogues

mime_run_swarp: Run SWarp on a set of images, creating a final mosaiced image

mime_flip_binary_mask: Convert a mask from CPL convention to Terapix/astromatic convention

mime_copy_files: Copy FITS image files from one directory to another

4.2 An introduction to ESOREX, GASGANO and REFLEX

The MIME pipeline modules can be used with several user interfaces provided by ESO. ESOREX is the application that actually executes the pipeline recipes. It can be used directly from the command line or in scripts. GASGANO is a graphical data management tool that allows automatic data classification by instrument-specific rules. REFLEX is a graphical workflow tool that automates classification and processing of data through to the final data product.

These interfaces are briefly described in this section. For more information, refer to the user manuals [11], [13] and [10].

4.2.1 Using ESOREX

ESOREX is a command-line utility for running pipeline recipes. ESOREX can be embedded easily by users into data-reduction scripts for the automation of processing tasks. The user has to classify and associate the data based on information contained in the FITS header keywords.

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

Here is an example SOF, valid for the recipe `mime_compute_bkg` applied to a set of HAWK-I images:

```
hawki_step_basic_calib_obj001.fits BASIC_CALIBRATED\\
hawki_step_basic_calib_obj002.fits BASIC_CALIBRATED\\
hawki_step_basic_calib_obj003.fits BASIC_CALIBRATED\\
hawki_step_basic_calib_obj004.fits BASIC_CALIBRATED\\
../HawkI-mask.fits STATIC_MASK
```

Esorex syntax: The basic syntax for using ESOREX is as follows:

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

To obtain more information on how to customise ESOREX run the command:

```
esorex -- help
```

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

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

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

```
esorex --recipes
```

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

```
esorex --params recipe_name
```

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

```
esorex --help recipe_name
```

To obtain more details about a given recipe, run the command

```
esorex --man-page recipe_name
```

Recipe configuration: An ESOREX configuration file can be assigned to each pipeline recipe, containing user-specified values of the parameters related to that recipe. The configuration files are normally generated in the directory `$HOME/.esorex` and have the same name as the recipe to which they are related, with the file name extension `.rc`.

The command

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

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

A recipe configuration file other than the default file can be specified on the command line:

```
esorex --recipe-config=my_recipe_config
```

More than one configuration file can be maintained for the same recipe, but a configuration file that is not located under `$HOME/.esorex` or has a name different from the recipe name has to be specified explicitly when a recipe is executed.

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

Recipe execution: A recipe can be run by specifying its name to ESOREX together with the name of a set of frames. For instance, the following command would be used to run the recipe `mime_compute_bkg` to create sky correction images for the files specified in the SOF file (set of frames) `compute_bkg.sof`:

```
esorex mime_compute_bkg compute_bkg.sof
```

The recipe parameters may be modified by editing the recipe configuration file or by specifying new parameter values on the command line using the command line options defined for that purpose. Such recipe options are placed after the recipe name and before the SOF name, and they will take precedence over the recipe defaults and the configuration file settings. For instance, to set the sky-estimation method to `kappa-sigma-clip` and the corresponding lower threshold to 5σ, the following command is used:

```
esorex mime_compute_bkg --method="kappa-sigma-clip" --kappa_low=5 \
    compute_bkg.sof
```

For more information on ESOREX see [11].

4.2.2 Using REFLEX

REFLEX is based on the workflow engine Kepler. In this section, we provide a brief general description of the use of Kepler and REFLEX. Detailed descriptions of the MIME workflows can be found in Sect. 7. The installation procedure is described in Appendix A.

The main components of REFLEX are workflows and actors. Workflows show the interdependence of pipeline recipes from the initial data organization to the final data product. Workflows are configured through variables that contain for instance path names to directories where the input data or data products reside. Actors are modules in the workflow which perform specific operations such as launching pipeline recipes. Other generic actors such as the `DataOrganizer` or the `FitsRouter` are used to manage the data files and the data flow through the workflow.

Once installed, REFLEX can be started from the shell with the command

```
reflex &
```

Workflows are created and executed interactively via the graphical user interface of REFLEX. After launching REFLEX the user is presented with the empty canvas shown in Fig. 4.2.1. A workflow can be loaded through the toolbar via File→Load and selecting the appropriate xml file from the file selection window.

Actors can be configured graphically by right-clicking on the symbols associated with them.

4.3 Example of NIR data reduction

A typical example of the reduction of a set of near-infrared imaging data using ESOREX is described here. We consider a set of HAWK-I exposures of the cluster of galaxies Abell 1689, taken in the K_s band. The data are from one OB and comprise 15 exposures. To obtain the output file names set by the MIME recipes the ESOREX parameter `suppress-prefix` has to be set to `TRUE` (e.g. permanently in the file `esorex.rc`).

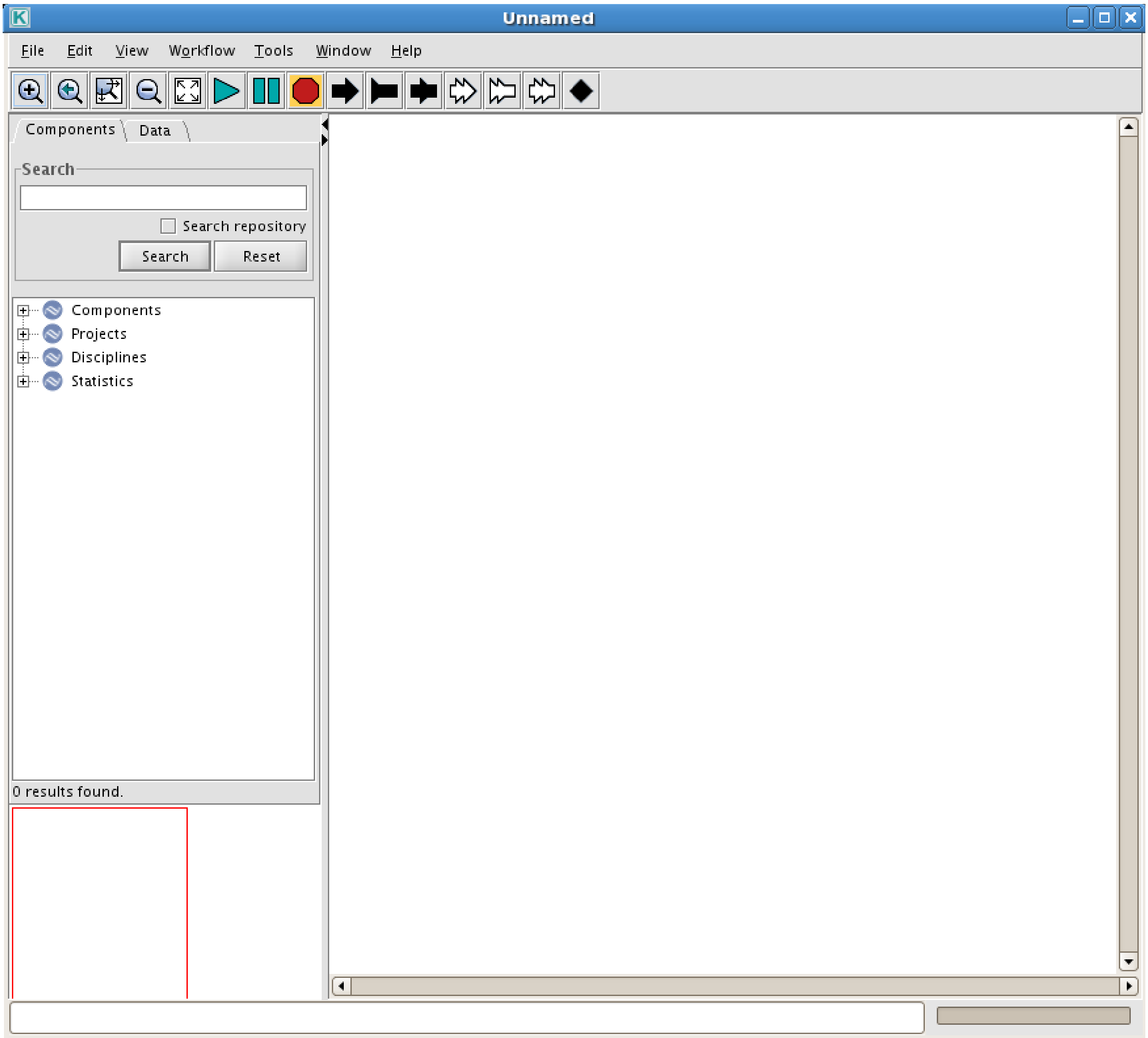

Figure 4.2.1: Fresh REFLEX canvas

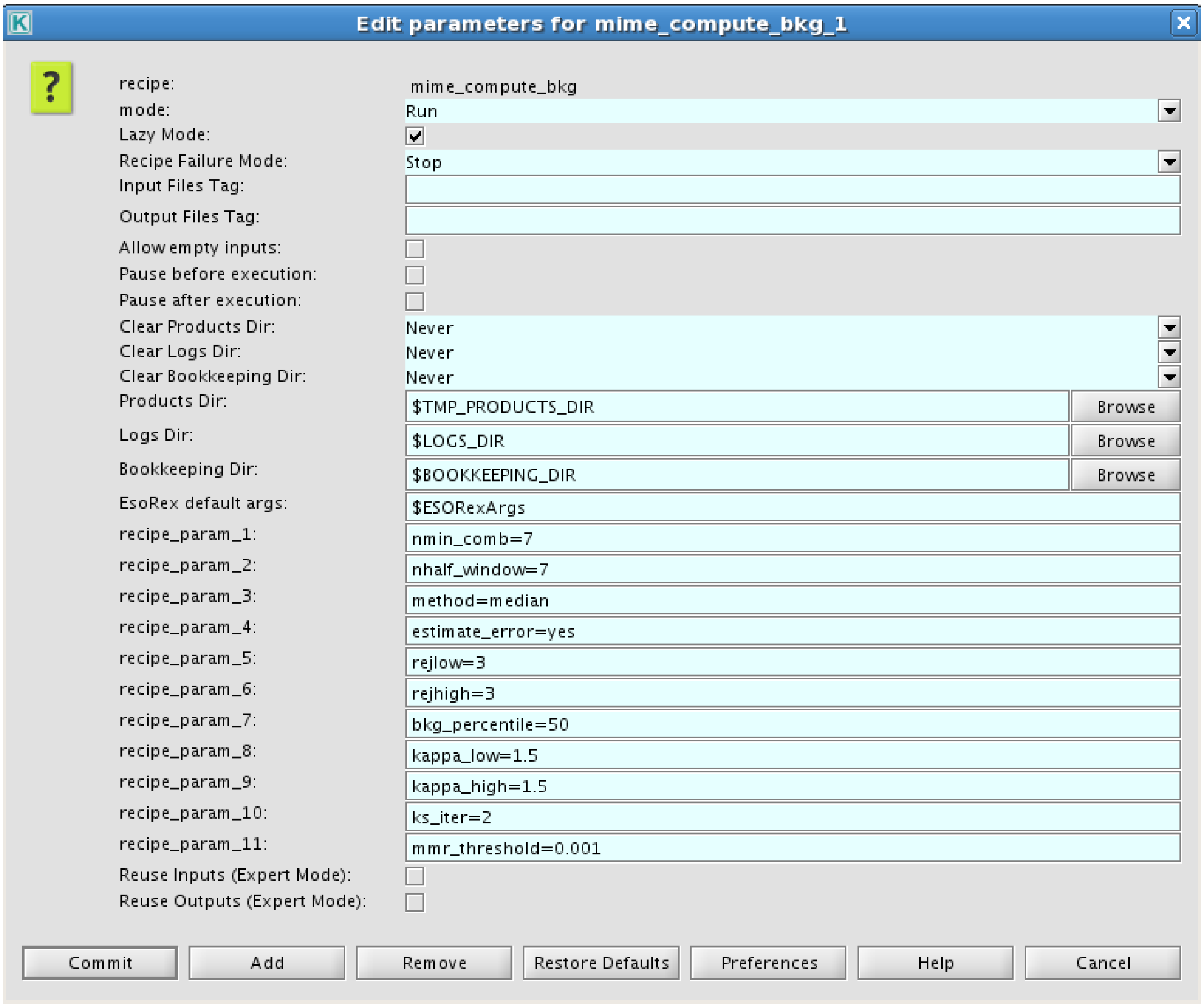

Figure 4.2.2: Parameters of a recipe actor. The parameters in the upper part of the list affect the execution of the recipe by ESOREXand the data organisation by REFLEX. Parameter values starting with a dollar sign are variables that are defined on the workflow level. The recipe parameters are listed in the lower part.

Since the HAWK-I instrument pipeline does not provide error maps, the first step is to create these from the exposures themselves using `mime_estimate_error_map`.

The input SOF looks like this:

File: `01-estimate_error.sof`

```
hawki_step_basic_calib_obj001.fits BASIC_CALIBRATED
hawki_step_basic_calib_obj002.fits BASIC_CALIBRATED
...
hawki_step_basic_calib_obj015.fits BASIC_CALIBRATED
```

Since `mime_estimate_error_map` does not have any recipe parameters, the command to execute the recipe is simply

```
esorex mime_estimate_error_map 01-estimate_error.sof
```

The recipe creates 15 new files named `mime_estimated_error_map_001.fits` etc.

The next step is to estimate the background for each exposure by taking the median over a number of exposures taken immediately before and after. This is done by the recipe `mime_compute_bkg`. The input consists of the basic calibrated exposures and their respective error maps. In addition, we include a static mask that marks the outermost five rows and columns of each quadrant as useless. The SOF is thus:

File: `02-compute_bkg.sof`

```
hawki_step_basic_calib_obj001.fits BASIC_CALIBRATED
...
hawki_step_basic_calib_obj015.fits BASIC_CALIBRATED
mime_estimated_error_map_001.fits ERROR_MAP
...
mime_estimated_error_map_015.fits ERROR_MAP
HawkI-mask.fits                 STATIC_MASK
```

The sky subtraction for near-infrared images is designed as a two-pass process in MIME. For simplicity, we create the background estimate in the first pass by taking the median of the relevant exposures. This is the default and also the fastest method provided by the recipe. The command to launch the recipe is

```
esorex mime_compute_bkg 02-compute_bkg.sof
```

The background images are subtracted from the basic calibrated images by `mime_subtract_bkg`. The error maps are not required in the first pass, hence the SOF is:

File: `03-subtract_bkg.sof`

```
hawki_step_basic_calib_obj001.fits BASIC_CALIBRATED
...
hawki_step_basic_calib_obj015.fits BASIC_CALIBRATED
mime_sky_bkg_001.fits           SKY_BKG_IM
...
mime_sky_bkg_015.fits           SKY_BKG_IM
```

The recipe is launched by

```
esorex mime_subtract_bkg 03-subtract_bkg.sof
```

The second pass will use a mask that removes all the objects from the estimate of the sky background. The mask is created from a stack of the exposures, taking into account the telescope offsets made between the background-corrected exposures. The SOF for the recipe to create the stack is:

File: `04-shift_and_add.sof`

```
mime_sky_bkg_corrected_001.fits BKG_CORRECTED
...
mime_sky_bkg_corrected_015.fits BKG_CORRECTED
```

The recipe is launched with

```
esorex mime_shift_and_add 04-shift_and_add.sof
```

Next, objects are detected in the combined image and an object mask is created. The SOF contains just the combined image:

File: `05-create_obj_mask.sof`

```
mime_stacked_image.fits COMBINED
```

The recipe is launched with

```
esorex mime_create_obj_mask 05-create_obj_mask.sof
```

For the second pass of the sky estimation and subtraction, the object mask just created is added to the SOF for `mime_compute_bkg`, which thus reads:

File: `06-compute_bkg.sof`

```
hawki_step_basic_calib_obj001.fits BASIC_CALIBRATED
...
hawki_step_basic_calib_obj015.fits BASIC_CALIBRATED
mime_estimated_error_map_001.fits ERROR_MAP
...
mime_estimated_error_map_015.fits ERROR_MAP
HawkI-mask.fits                STATIC_MASK
mime_obj_mask_001.fits         OBJ_MASK
```

Since the objects are now masked, we can take a milder approach to outlier rejection and employ $\kappa\sigma$-clipping using the recipe parameter `--method` in order to identify cosmic ray hits and other defects. This method gives a slightly higher signal-to-noise ratio than the median. We set $\kappa = 3$ for both high and low outliers. The command to execute the recipe is thus

```
esorex mime_compute_bkg --method=kappa_sigma_clip --kappa_low=3 \
   --kappa_high=3 06-compute_bkg.sof
```

The SOF for `mime_subtract_bkg` now contains the error maps for the basic calibrated and the background images in addition to the images themselves. This ensures that the errors are properly propagated to the next step.

File: `07-subtract_bkg.sof`

```
hawki_step_basic_calib_obj001.fits BASIC_CALIBRATED
...
hawki_step_basic_calib_obj015.fits BASIC_CALIBRATED
mime_estimated_error_map_001.fits ERROR_MAP
...
mime_estimated_error_map_015.fits ERROR_MAP
mime_sky_bkg_001.fits           SKY_BKG_IM
...
mime_sky_bkg_015.fits           SKY_BKG_IM
mime_sky_bkg_error_001.fits     BKG_ERROR_MAP
...
mime_sky_bkg_error_015.fits     BKG_ERROR_MAP
```

The recipe is executed by

```
esorex mime_subtract_bkg 07-subtract_bkg.sof
```

The next stage is the astrometric calibration of the images as a preparation to resampling the images to a common pixel grid and combining them into a final deep mosaic image. Up to this point we have worked with data from a single OB. Now, all available exposures of the given field should be combined.

MIME relies on the programmes `SExtractor`, `Scamp` and `SWarp` from the Terapix/Astromatic software suite by E. Bertin for the astrometric calibration and remapping and provides wrapper recipes for their execution in a way that is compliant with the requirements of the ESO data flow system.

First, `SExtractor` is used to create object catalogues of the exposures, containing measurements of centroids and fluxes. A default configuration file is provided which should provide reasonable results in most cases, but the user can create a separate configuration file or specify values for individual parameters that override the defaults. The input SOF lists the background-corrected images and their error maps:

File: `08-run_sextractor.sof`

```
mime_sky_bkg_corrected_001.fits BKG_CORRECTED
...
mime_sky_bkg_corrected_015.fits BKG_CORRECTED
mime_sky_bkg_corrected_error001.fits BKG_CORRECTED_ERROR
...
mime_sky_bkg_corrected_error015.fits BKG_CORRECTED_ERROR
```

With the default configuration, the recipe is launched by

```
esorex mime_run_sextractor 08-run_sextractor.sof
```

`scamp` is used to derive the astrometric solutions for the exposures. Configuring `scamp` can be tricky and it is hardly possible to provide a default configuration which works for all or even most cases and it will almost always be necessary to run the recipe repeatedly to find parameter values tuned to the particular data set that is being worked on. `scamp` produces check plots that are helpful in this process.

The SOF lists the catalogues created by `SExtractor`:

File: `09-run_scamp.sof`

```
mime_sky_bkg_corrected_001.cat CATALOGUE
...
mime_sky_bkg_corrected_015.cat CATALOGUE
```

For the HAWK-I K_s-band data of Abell 1689, we obtained good results with the following parameters:

```
esorex mime_run_scamp --extra_config="ASTREF_CATALOG:SDSS-R7|DISTORT_DEGREES:3|
   ASTREF_WEIGHT:1000|ASTREFMAG_LIMITS:17,21|FWHM_THRESHOLDS:1,20|CROSSID_RADIUS:1.0"
    09-run_scamp.sof
```

The choice of the reference catalogue can be quite critical, in particular for instruments with small fields of view. The field considered here was covered by the Sloan Digital Sky Survey which provides the densest set of reference objects available. The astrometry is modelled by a cubic polynomial. The match to the reference catalogue has been given large weight in relation to internal matches between HAWK-I exposures. The magnitude limits for the reference objects have been set so as to include only objects with reasonably high signal-to-noise ratio, and limits on object sizes have been set so as to select only fairly compact objects. The search radius for object matches has been decreased compared to the default to obtain robust matches in this moderately crowded field. With these settings the internal positional residuals are $\sigma_x = 0.''036$ and $\sigma_y = 0.''050$ and the external residuals are $\sigma_{x,\mathrm{ref}} = 0.''071$ and $\sigma_{y,\mathrm{ref}} = 0.''069$ (the pixel scale of HAWK-I is $0.''1064$).

Finally, the images are ready to be remapped onto a common output grid and to be combined into the final deep mosaic image. The SOF contains the background-corrected images, their error maps and the header files created by `scamp`, which contain the astrometric solutions:

File: `10-run_swarp.sof`

```
mime_sky_bkg_corrected_001.fits BKG_CORRECTED
...
mime_sky_bkg_corrected_015.fits BKG_CORRECTED
mime_sky_bkg_corrected_error001.fits BKG_CORRECTED_ERROR
...
mime_sky_bkg_corrected_error015.fits BKG_CORRECTED_ERROR
mime_sky_bkg_corrected_001_head.fits HEADER
...
mime_sky_bkg_corrected_015_head.fits HEADER
```

The command to launch the recipe is

```
esorex mime_run_swarp 10-run_swarp.sof
```

This creates the final image `mime_coadd.fits` and its corresponding error map `mime_coadd_weight.fits`.

5 Data reduction

The MIME pipeline recipes are meant to bring basic calibrated (or pre-reduced) data sets to final, scientifically useable products. The input data consists of a set of exposures of a field, taken in one or more OBs, that have undergone the basic reduction procedures, such as bias and dark subtraction and flat-field correction. They are assumed to be products of one of ESO's instrument pipelines.

MIME provides recipes that deal with background subtraction, astrometric calibration, homogenization of the data set, and the creation of a final mosaiced and stacked output image that can be used for scientific analysis. In this section, we present the reduction cascade, i.e. the sequence in which the recipes are typically applied and the flow of data and data products through that sequence.

REFLEX workflows for the most common situations are distributed with the MIME/REFLEX pipeline. The application of individual recipes is described in Sect. 6. The workflows are described in Sect. 7.

The reduction cascade and file association map for the NIR data reduction is shown in Fig. 5.0.1. This includes the case where the background is computed from dedicated sky exposures and the case where the background is computed from the science data. Error maps are listed as input data when they are available as products from the instrument pipeline, or as products of the recipe `mime_estimate_eror_map`, when they are not.

The corresponding cascade for optical data is shown in Fig. 5.0.2.

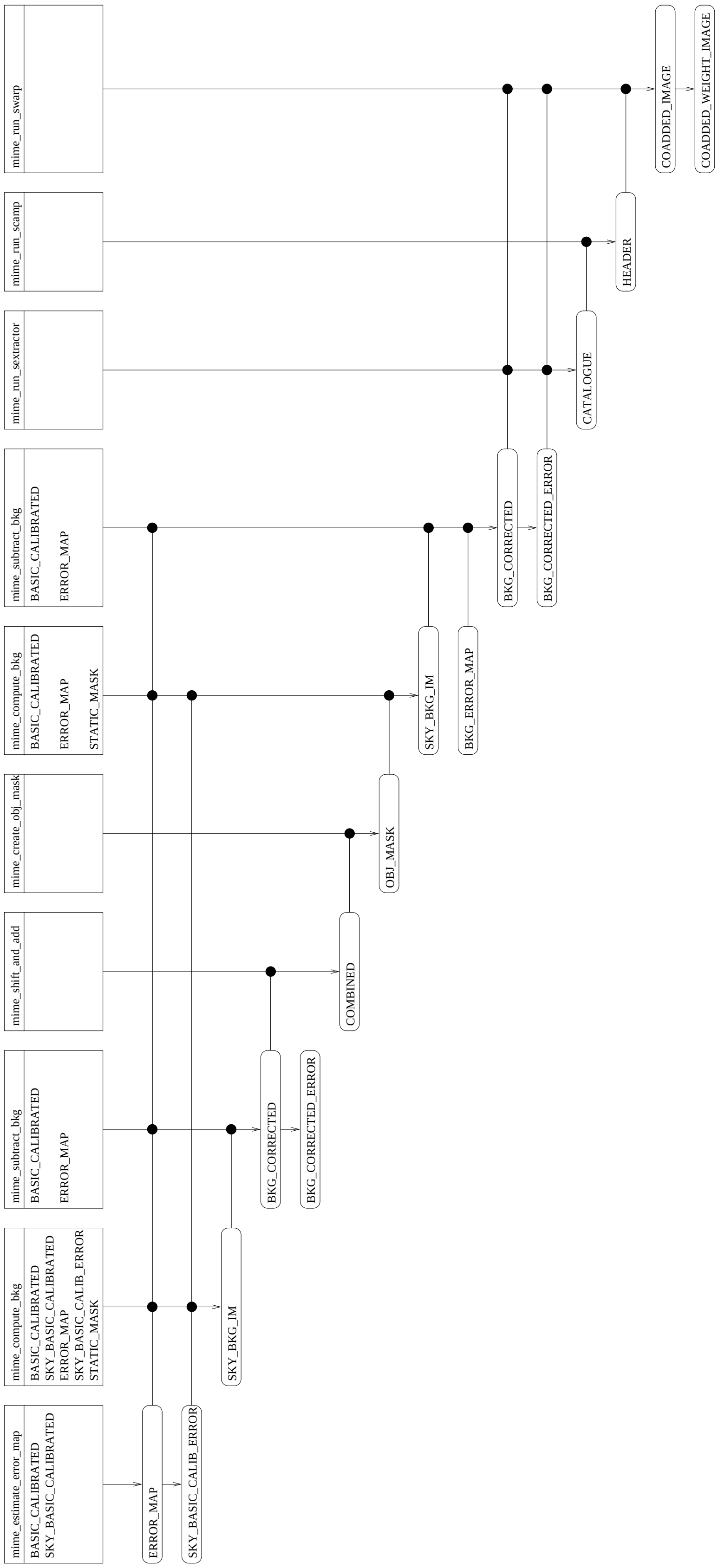

Figure 5.0.1: Reduction cascade for NIR data.

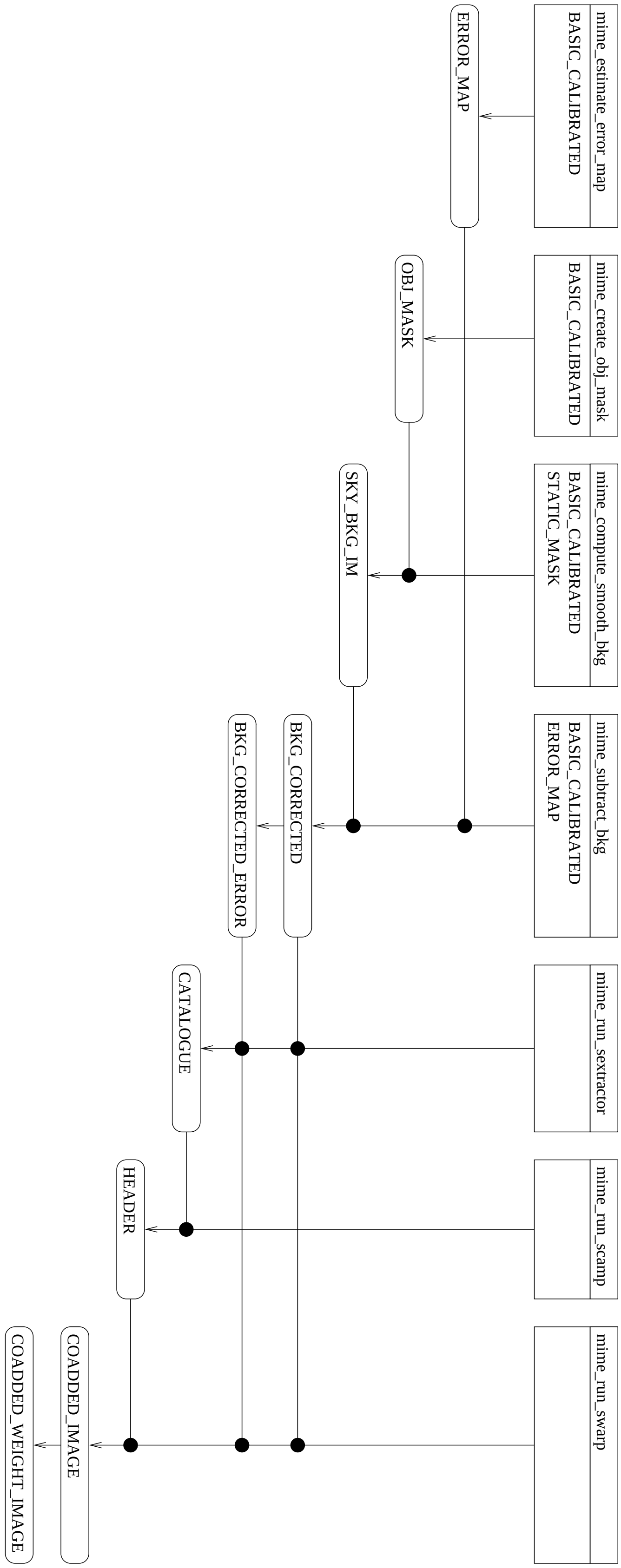

Figure 5.0.2: Reduction cascade for optical data.

6 Pipeline recipe interfaces

In this section, the required input data for the MIME recipes, their products and the recipe parameters are described. The recipes are listed in alphabetical order.

6.1 mime_compute_bkg

This recipe creates background images by computing a robust running mean over a set of dithered science exposures or by stacking exposures of blank sky. The input images should come from a single OB and be the basic calibrated products of an instrument pipeline. In the first case, a background image is created for each science exposure in the set, in the second case, all sky exposures are combined into a single background image. The second case applies if at least one sky exposure is listed in the SOF.

The recipe provides the possibility to derive error estimates for the input images internally. This is the identical functionality as with the recipe `mime_estimate_error_maps`.

6.1.1 Input

- Input frames classified as `BASIC_CALIBRATED`, if the background is to be computed from science images. Optionally, one noise image for each input image, classified as `ERROR_MAP`.
- Input frames classified as `SKY_BASIC_CALIBRATED`, if the background is to be computed from sky images. Optionally, one noise image for each sky image, classified as `SKY_BASIC_CALIB_ERROR`.
- Optionally, a static mask excluding bad pixels or bad regions, classified as `STATIC_MASK`.
- Optionally, an object mask excluding regions covered by astronomical sources, classified as `OBJ_MASK`. Such a mask can be created by the recipe `mime_create_obj_mask`.

6.1.2 Products

- one background image for each input image if computed from science exposures, or a single background image if computed from sky exposures. The background images are named `mime_sky_bkg_`*xxx*`.fits` (`PRO.CATG = SKY_BKG_IM`).
- a noise image for each background image, named `mime_sky_bkg_error_`*xxx*`.fits` (`PRO.CATG = BKG_ERROR_MAP`)
- If error maps are estimated internally, these are saved as `mime_estimated_error_map_`*xxx*`.fits` (`PRO.CATG = ERROR_MAP`).

6.1.3 Parameters

- `estimate_error`: option to estimate error maps internally (default is `yes` if no error maps are provided in the SOF, otherwise it is `no`).

- nmin_comb: minimum number of dithered images used to compute the running mean (default is 7).
- nhalf_window: half length of the symmetric window around the exposure to compute the running mean (default is 7)
- bkg_percentile: percentile of the image histogram to be used as scaling factor before computing the running mean (default is 50, i.e. images are scaled by their median level)
- method: robust estimator or outlier rejection method for computing the running mean. Default is median. The other options are min_max_rejection, kappa_sigma_clip and mean_median_ratio_constraint.
- rejlow, rejhigh: the numbers of low/high pixel values to reject from the running mean if method is min_max_rejection (default is 3 for both parameters)
- kappa_low, kappa_high: low and high sigma clipping factors if method is kappa_sigma_clip (default is 1.5 for both parameters)
- ks_iter: number of iterations if method is kappa_sigma_clip (default is 2)
- mmr_threshold: threshold for the deviation of the mean-median ratio from 1 if method is mean_median_ratio_constraint (default is 0.001)

6.2 mime_compute_smooth_bkg

6.2.1 Input

This recipe creates a smooth background image from a science image by fitting a polynomial surface. Image regions with astronomical sources can be excluded from the fit by applying an object mask.

- basic calibrated science images classified as BASIC_CALIBRATED
- for each image an object mask to exclude astronomical sources from the fit, classified as OBJ_MASK
- an optional static mask to exclude bad pixels or detector regions, classified as STATIC_MASK

6.2.2 Products

- for each input image a background image named mime_smooth_bkg_*xxx*.fits (PRO.CATG = SKY_BKG_IM)

6.2.3 Parameters

- dim_X, dim_Y: dimensions along the x and y axes of the polynomial to be fitted (default is 2 for both parameters). Note that the degree of a polynomial is equal to its dimension minus one.

6.3 mime_copy_files

This recipe copies FITS images from one directory to another. It is a utility recipe used in REFLEXworkflows to permit running in lazy mode. It should not be necessary to use it interactively with ESOREX.

6.3.1 Input

- FITS images with any classification

6.3.2 Products

- Copies of the input images with `_copy_`*xxx*`.fits` appended to the file name. The `PRO.CATG` is preserved.The files are written to the current working directory or the directory given by the ESOREX parameter `output_dir`.

6.3.3 Parameters

This recipe has no parameters. To specify a target directory other than the current working directory use the ESOREXparameter `output_dir`.

6.4 mime_create_obj_mask

This recipe creates a mask that flags pixels having a value higher than a threshold above the background. The masked regions are grown to include low surface-brightness wings of objects in the field.

6.4.1 Input

- Input images classified as `BASIC_CALIBRATED` or `COMBINED`

6.4.2 Products

- Binary masks named `mime_obj_mask_`*xxx*`.fits` (`PRO.CATG = OBJ_MASK`). Masked regions have value 1, unmasked regions 0.

6.4.3 Parameters

- `sigma_det`: sets the threshold for masking pixels to `sigma_det` times the background standard deviation above the background level (default is 3.0)
- `growing_radius`: growing radius for masked regions. If a pixel is within this distance from a pixel detected as having object flux, it is also masked (default is 5).

6.5 mime_estimate_error_maps

This recipe estimates error maps from input images using a simple Poisson model for photon noise. For pixels that have value 1 in a optional static binary mask, the error is set to a very large value (`FLT_MAX`).

6.5.1 Input

- input images classified as `BASIC_CALIBRATED` or `SKY_BASIC_CALIBRATED`
- optionally, a static mask excluding bad pixels or bad regions, classified as `STATIC_MASK`.

6.5.2 Products

- error maps named `mime_estimated_error_map_`*xxx*`.fits` (`PRO.CATG = ERROR_MAP` or `PRO.CATG = SKY_BASIC_CALIB_ERROR`).

6.5.3 Parameters

This recipe has no parameters.

6.6 mime_flip_binary_mask

This recipe swaps values of 0 and 1 in a binary mask, turning it into a weight map. The recipe makes masks usable by `SExtractor` and `SWarp`.

6.6.1 Input

- binary masks classified as `BPM`

6.6.2 Products

- binary masks named `mime_flipped_BPM_`*xxx*`.fits` (`PRO.CATG = BPM`). These have value 1 for good pixels and 0 for bad pixels, which is the opposite of the CPL convention.

6.6.3 Parameters

This recipe has no parameters.

6.7 mime_run_scamp

This recipe runs `scamp` on a set of input catalogues created by `mime_run_sextractor` and computes astrometric solutions as well as photometric scaling factors between the images. The default configuration parameters are obtained from the file `mime_default.scamp` which is distributed with the MIMEpipeline.

6.7.1 Input

- a set of catalogues in FITS-LDAC format, classified as `CATALOGUE`. The files must have extension `.cat`.

6.7.2 Products

- FITS header files containing WCS keywords derived by `scamp`. The files have the same name as the input catalogues but with the extension `.cat` replaced by `_head.fits`, e.g. `mime_sky_bkg_corrected_`*xxx*`_head.fits`.
- an ASCII file `mime_run_scamp_config.scamp` with the full list of configuration parameters used by `scamp`.

6.7.3 Parameters

- `config_fname`: name of a file with configuration parameters for `scamp`. These parameters will override the ones in the default configuration file.
  - `extra_config`: additional configuration parameters for `scamp`. The format is `--extra_config="PARAMETER1:VALUE1|PARAMETER2:VALUE2|..."`.
    These parameter values will override those in the configuration files.

6.8 mime_run_sextractor

This recipe runs `SExtractor` on a set of input images to create catalogues of objects detected in the images. The default configuration is obtained from the file `mime_default.sex` distributed with the MIME pipeline. The measurements required for the analysis of the object catalogues by `scamp` are defined in `mime_default.param`.

6.8.1 Input

- background corrected image frames classified as `BKG_CORRECTED`
- optionally, error maps for the input images, classified as `BKG_CORRECTED_ERROR`. The error maps are used as type `MAP_RMS` by `SExtractor`.
- optionally, bad pixel masks for each input images or one bad pixel mask to be applied to all input images, classified as `BPM`. The masks are used as type `MAP_WEIGHT` by `SExtractor`. If both error maps and bad pixel masks are given in the SOF, only the error maps are used.

6.8.2 Products

- object catalogues in FITS-LDAC format. The files have the same name as the input images, with the extension `.fits` replaced by `.cat` (`PRO.CATG = CATALOGUE`).
- an ASCII file `mime_run_sextractor_config.sex` with the full list of configuration parameters used by `SExtractor`.

6.8.3 Parameters

- `config_fname`: name of a file with configuration parameters for `SExtractor`. These parameters will override the ones in the default configuration file.
  - `extra_config`: additional configuration parameters for `SExtractor`. The format is `--extra_config="PARAMETER1:VALUE1|PARAMETER2:VALUE2|..."`.
    These parameter values will override those in the configuration files.

6.9 mime_run_swarp

This recipe runs `SWarp` on a set of input images to create a regridded, mosaicked and stacked output image. The default configuration is obtained from the file `mime_default.swarp` distributed with the MIME pipeline.

6.9.1 Input

- background-corrected images classified as `BKG_CORRECTED`
- optionally, error maps for the input images, classified as `BKG_CORRECTED_ERROR`. The error maps are used as type `MAP_RMS` by `SWarp`.
- optionally, (flipped) bad pixel masks for each input images or one bad pixel mask to be applied to all input images, classified as `BPM`. The masks are used as type `MAP_WEIGHT` by `SWarp`. If both error maps and bad pixel masks are given, only the error maps are used.
- FITS header files classified as `HEADER`. These should contain WCS keywords specifying the astrometric solutions for the images. If no header files are given, the WCS keywords from the images are used.

6.9.2 Products

- a coadded, mosaiced output image named `mime_coadd.fits` (`PRO.CATG = COADDED_IMAGE`)
- a weight image for the output image, named `mime_coadd_weight.fits` (`PRO.CATG = COADDED_WEIGHT_IMAGE`)
- a masked, mosaiced output image named `mime_coadd_masked.fits` (`PRO.CATG = COADDED_IMAGE`). This image is identical to `mime_coadd.fits`, but pixels with very low weight are assigned a value of 0.

- an ASCII file `mime_run_swarp_config.swarp` with the full list of configuration parameters used by `SWarp`.

6.9.3 Parameters

- `config_fname`: name of a file with configuration parameters for `SWarp`. These parameters will override the ones in the default configuration file.
  - `extra_config`: additional configuration parameters for `SWarp`. The format is `--extra_config="PARAMETER1:VALUE1|PARAMETER2:VALUE2|..."`.
    These parameter values will override those in the configuration files.

6.10 mime_shift_and_add

This recipe creates a stacked image from a set of dithered images using integer pixel shifts to correct for telescope offsets. The output image is not suited for scientific use but is meant to serve for creation of an object mask.

6.10.1 Input

- background-corrected image frames classified as `BASIC_CALIBRATED`
- optionally, bad pixel masks for each input images or one bad pixel mask to be applied to all input images, classified as `BPM`.

6.10.2 Products

- a combined image named `mime_stacked_image.fits`
- a bad pixel mask named `mime_stacked_bpm.fits` (`PRO.CATG = BPM`)

6.10.3 Parameters

- `nmin_comb`: minimum number of pixel values to combine (default is 1). If the number of pixel values after outlier rejection is less than `nmin_comb` the pixel is flagged in the output bad pixel mask.
- `method`: robust estimator or outlier rejection method to use for image stacking. Default is `median`. The other options are `min_max_rejection`, `kappa_sigma_clip` and `mean_median_ratio_constraint`.
- `rejlow`, `rejhigh`: the numbers of low/high pixel values to reject if `method` is `min_max_rejection` (default is 3 for both parameters)
- `kappa_low`, `kappa_high`: low and high sigma clipping factors if `method` is `kappa_sigma_clip` (default is 1.5 for both parameters)

- `ks_iter`: number of iterations if `method` is `kappa_sigma_clip` (default is 2)
- `mmr_threshold`: threshold for the deviation of the mean-median ratio from 1 if the parameter `method` is set to `mean_median_ratio_constraint` (default is 0.001)

6.11 mime_subtract_bkg

This recipe subtracts background images from basic calibrated images. There should be one background image for each basic calibrated image or a single background image which is subtracted from all basic calibrated images.

6.11.1 Input

- basic calibrated images classified as `BASIC_CALIBRATED`
- background images classified as `SKY_BKG_IM`.
- optionally, error maps for the basic calibrated images, classified as `ERROR_MAP`
- optionally, error maps for the background images, classified as `BKG_ERROR_MAP`

6.11.2 Products

- background-corrected images named `mime_sky_bkg_corrected_`*xxx*`.fits`
  (`PRO.CATG = BKG_CORRECTED`)
- error maps for the background-corrected images named `mime_sky_bkg_corrected_error`*xxx*`.fits`
  (`PRO.CATG = BKG_CORRECTED_ERROR`)

6.11.3 Parameters

This recipe has no parameters.

7 Reflex workflows

MIME provides three REFLEX workflows which cover the most common types of astronomical data sets for optical and near-infrared imaging. They implement different background subtraction strategies:

`mime_mosaicing_nir_dither.xml` : images were taken in a near-infrared waveband where the sky background shows significant variation over the field. The background is estimated using the two-pass method.

`mime_mosaicing_nir_sky.xml` : in addition to the science images taken in a near-infrared waveband, additional exposures were obtained on a nearly empty piece of sky close to the target field. The background is estimated by stacking the sky exposures.

`mime_mosaicing_optical.xml` : images were taken in an optical waveband where the sky background varies smoothly over the field and can be modelled by fitting low-order polynomial surfaces to the sky background of the images.

In general, each workflow consists of three parts:

- Management of input data: the basic calibrated input images are grouped based upon instrument and detector characteristics and observing blocks.
- Background subtraction: different strategies of estimating the background are employed depending on the type of input data.
- Astrometric and photometric correction of the individual data sets and final mosaicing.

7.1 MIME workflow for dithered near-infrared images

This workflow implements the two-pass strategy for the background estimation. It is applicable for near-infrared data from instruments like HAWK-I, ISAAC and SOFI. It can also be used for optical images in red bands that are affected by fringing, if a sufficient number of exposures from the same OB are available to permit robust removal of all object contributions.

7.2 MIME workflow for near-infrared images with separate sky images

This workflow requires exposures of a piece of nearly empty sky to be taken in the same OB as the science data. This observing strategy is typically used when the target field is crowded and the two-pass method for sky estimation cannot be expected to be able to remove all contribution from objects from the science images. It is applicable (but not restricted) to data from the same NIR instruments as the workflow described in Sect. 7.1.

7.3 MIME workflow for optical images

This workflow applies to data taken in the optical wavebands in which the sky background varies smoothly across the field. It can be used with data from VIMOS or WFI.[4]

[4]It cannot at present be applied to data from FORS because the instrument pipelines creates products in a format that differs from that of the other instrument pipelines.

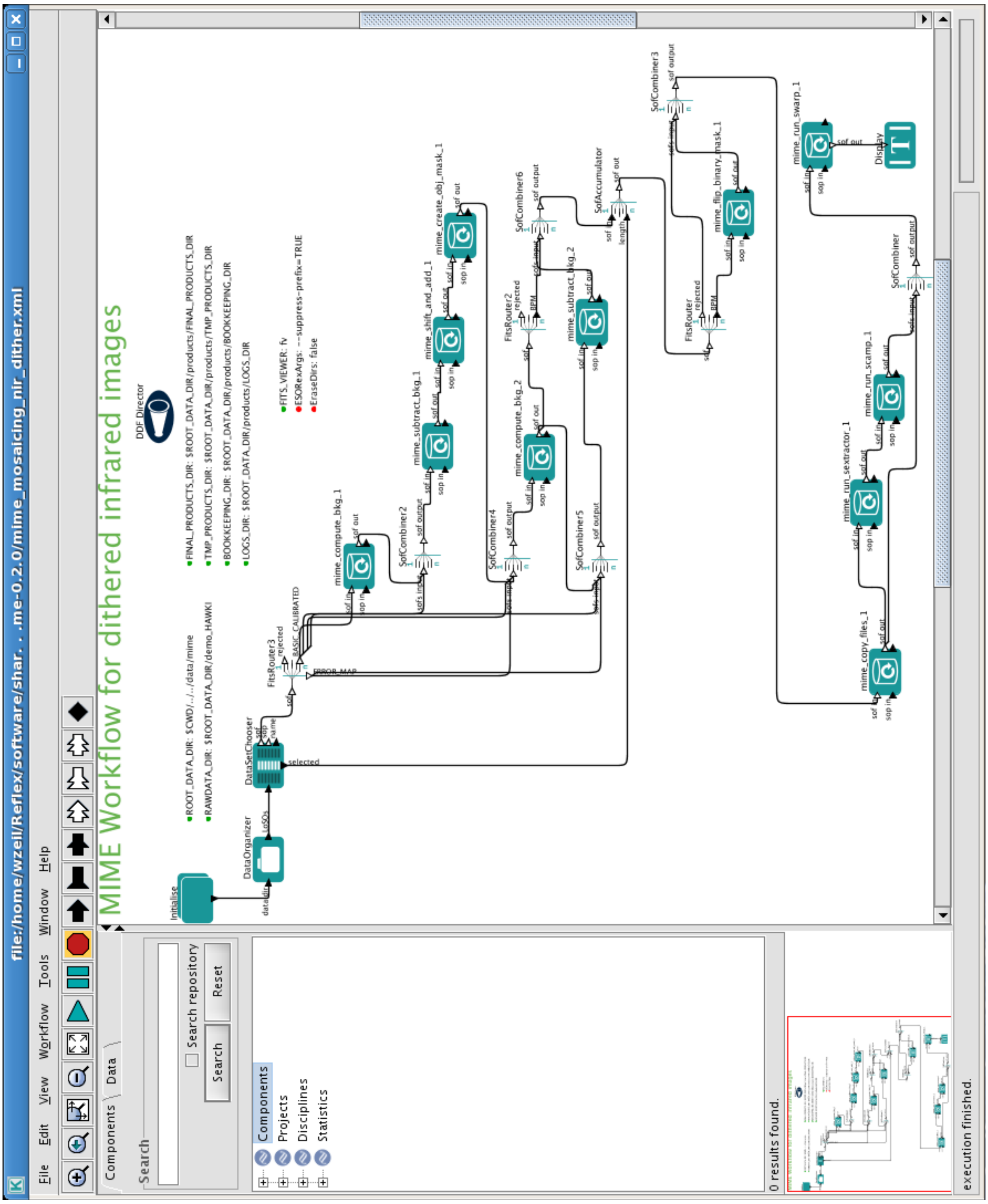

Figure 7.3.1: MIME REFLEX workflow for dithered NIR images

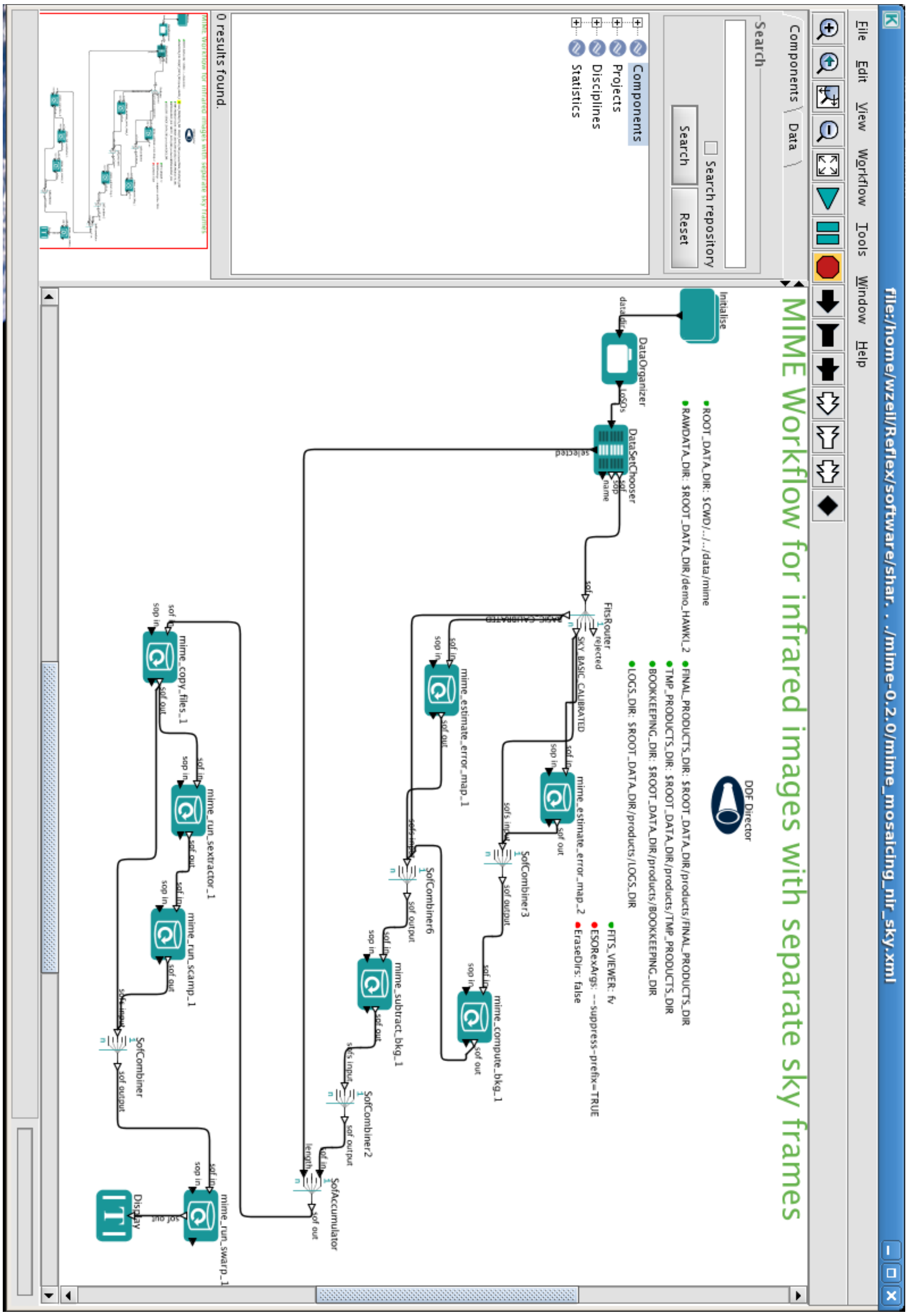

Figure 7.3.2: MIME REFLEX workflow for NIR images with separate sky images

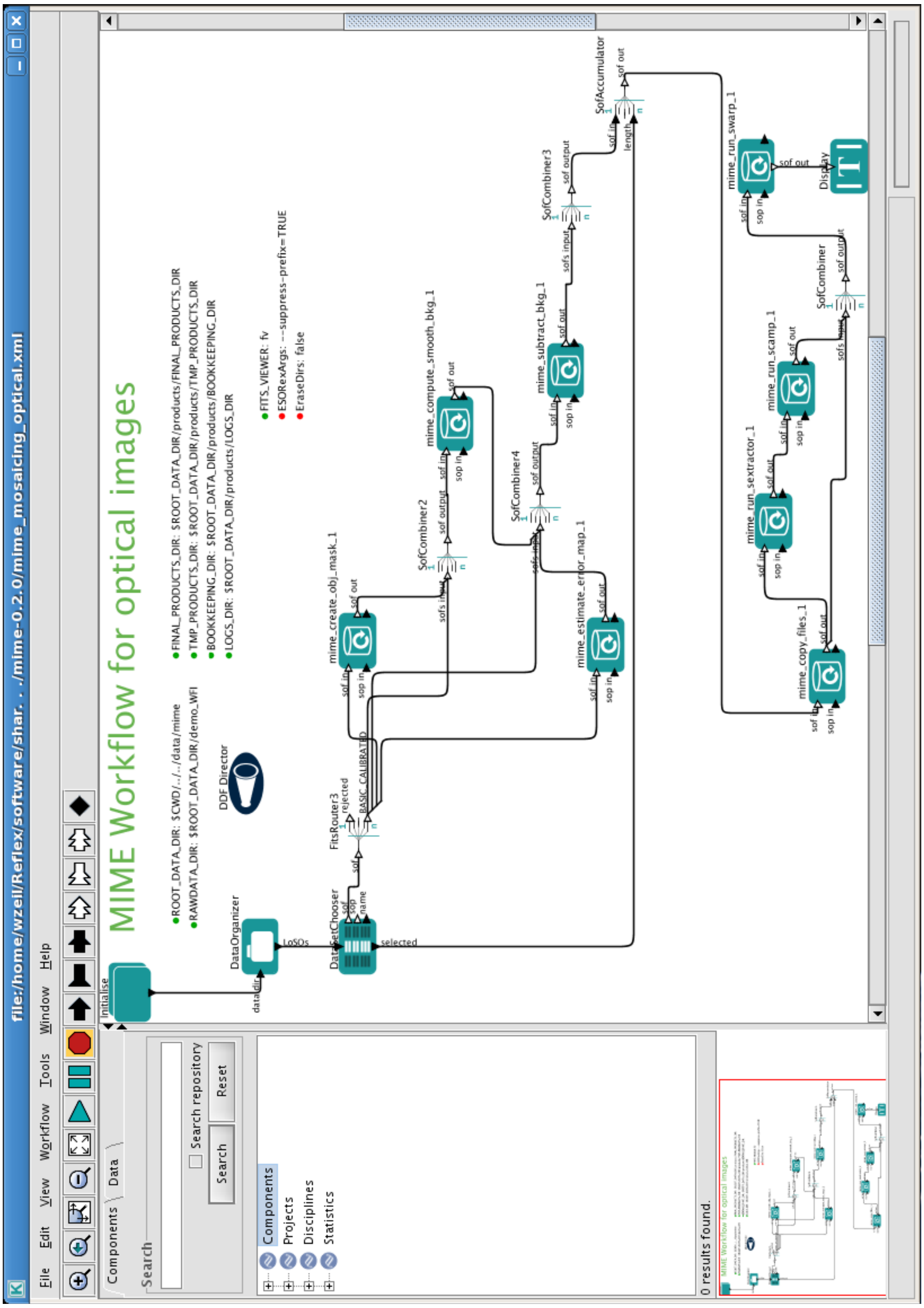

Figure 7.3.3: MIME REFLEX workflow for NIR images with optical images

8 Algorithms

In this section we describe the main algorithms implemented in the MIMEpipeline recipes.

8.1 Estimation of pixel noise

Astronomical images should be accompanied by error maps giving the likely noise contribution at each pixel of the image. This makes it possible to assign significances to detections and to estimate errors on photometric and other measurements in the course of the scientific analysis of the images. The noise includes contributions from photon noise associated with the photon counting process over the integration time of an exposure, dark current noise, CCD read-out noise, digitization noise, and possibly other sources.

The basic reduction procedures, such as bias and dark subtraction and flat-field correction, add the noise from calibration images to the noise of the science exposures. In the ideal case, these noise contributions should be tracked during the reduction process and the instrument pipelines should provide error (noise) maps in addition to the basic calibrated images that are used as input to MIME.

Unfortunately, not all instrument pipelines offered by ESO currently provide such error maps. The MIMEpipeline therefore includes a recipe, `mime_estimate_error_map` that estimates the noise in the basic calibrated input images based on a simple model. Error estimation can also be done within the recipe `mime_compute_bkg`.

It is assumed that the input images are photon-noise dominated. The number of electrons, N_e, liberated in a pixel of the CCD over the exposure time then follows a Poisson distribution with variance $\sigma^2 = \langle N_e \rangle$, where $\langle N_e \rangle$ is the expected number of electrons.

Pixel values F in a raw image are given in ADU and are related to the number of electrons via the conversion factor g (in $\mathrm{e}^-/\mathrm{ADU}$) which specifies the gain setting of the instrument's analog-to-digital converter and which we refer to as the "gain"[5]:

$$F = gN_e. \tag{1}$$

The expected noise in the image is thus

$$\sigma_F = g\sqrt{\langle N_e \rangle} = \sqrt{\frac{\langle F \rangle}{g}}. \tag{2}$$

The expected count rate $\langle F \rangle$ in a given pixel is estimated by the actual count rate, hence the estimated noise is

$$\hat{\sigma}_F = \sqrt{\frac{F}{g}} \tag{3}$$

Note that the gain factor can fluctuate or drift with time. At the processing stage to which the MIME recipes apply, it is not possible to determine the gain factor empirically, so that the recipes have to rely on external information. It has to be kept in mind that the error maps determined by `mime_estimate_error_map` or

[5] There is some confusion regarding the terminology in ESO instrument manuals and quality control web pages, some of which use the term "gain" for the quantity g, and others for the inverse quantity, $1/g$. According to the Data Interface Control Document [9] (Appendix B), the value for g is given by the FITS keyword `ESO DET OUT CF` as "conversion factor from ADUs to e^- (e^-/ADU)" or `ESO DET OUT CONAD`.

Table 8.1.1: Conversion factors ("gain") for various ESO instruments as used by recipe `mime_estimate_error_map`. Basic calibrated images from these instruments do not record their gain values in the FITS headers.

Instrument	Detector	g (e^{-}/ADU)
ISAAC	SW – Rockwell	4.500
HAWKI	#66 (Q1)	1.705
	#78 (Q2)	1.870
	#79 (Q3)	1.735
	#88 (Q4)	2.110
WFI	#50	2.001
	#51	2.052
	#52	2.283
	#53	1.981
	#57	2.174
	#56	2.185
	#55	2.048
	#54	1.994

`mime_compute_bkg` are fairly rough estimates and the user has to judge whether using internally estimated error maps are useful for the data in hand or whether a conservative approach using constant weights is to be preferred.

The recipes first check for the presence of the keyword

```
HIERARCH ESO DET OUT1 CONAD
```

which is the conversion factor "g". If this is not available, the recipe identifies the detector used in the data set based on the keyword `HIERARCH ESO DET CHIP ID` and assigns the corresponding conversion factor as listed in Table 8.1.1.

The conversion factors obtained in this way apply to a single exposure and can be used as is for optical instruments, such as VIMOS or WFI. Near-infrared images, on the other hand, are created by averaging over two (*double-correlated read*) or more (*non-destructive read*) short exposures. The values in the resulting raw image thus correspond to the integration time of a single exposure (as listed in `HIERARCH ESO DET DIT`), whereas the noise is reduced by the square root of the number of exposures as compared to a single exposure:

$$\hat{\sigma}_F = \sqrt{\frac{F}{g \cdot \mathrm{NDIT}}} \qquad (4)$$

The number of exposures is recorded in `HIERARCH ESO DET NDIT`. The quantity $g \cdot \mathrm{NDIT}$ represents the effective gain factor for the exposure.

8.2 Robust estimators of the mean – outlier rejection

Astronomical data are often affected by cosmic ray hits, satellite or airplane trails, reflections etc., i.e. data values which do not arise from the usual noise distribution of the astronomical image and which should not appear in the stacked image. In the course of the two-pass method for NIR sky subtraction, images are stacked in the image frame with the goal of removing light from stars and other astronomical sources so that a clean estimate of the sky background is obtained. In photon-noise dominated images, the noise in the background arises from a Poisson process which can be well approximated asymptotically by a Gaussian process with the constraint that $\sigma = \mu$. In the list of pixel values to be combined for a given location in the image frame, values that are affected by object light (or cosmic ray hits and the like) will appear as outliers from the Gaussian distribution and have to be identified and removed from the list. We discuss here a number of robust estimators of the mean of a sample of data values and outlier rejection methods which have been implemented in the MIMEmodules.

8.2.1 Arithmetic mean

The arithmetic mean of a sample $x_1, x_2, \ldots, x_N$ is defined as

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i\,. \tag{5}$$

For a random variable X that is normally distributed ($N(\mu,\sigma)$), the sample mean is the most efficient estimator of the population mean μ. By the central limit theorem this statement holds asymptotically for many noise distributions encountered in practice.

If the noise distribution of X is unknown, its variance can be estimated from the sample as

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2\,. \tag{6}$$

If X has variance σ then the average (5) has variance

$$\sigma_{\mathrm{av}}^2 = \frac{\sigma^2}{N}. \tag{7}$$

This relation can be used in place of Eq. (6) if an *a priori* estimate of the noise in the images is available and is the same for all images (homescedastic case).

When each image has its own noise image $\sigma_i^2(x,y)$ (heteroscedastic case), then the weighted mean

$$\bar{x} = \left(\sum \frac{x_i}{\sigma_i^2}\right)\left(\sum \frac{1}{\sigma_i^2}\right)^{-1} \tag{8}$$

can be used, giving higher weight to values with lower noise. The variance of the average value at a given pixel is in this case given by

$$\frac{1}{\sigma_{\mathrm{av}}^2} = \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\,. \tag{9}$$

While the average is efficient, it is also sensitive to the presence of outliers, i.e. sample values which arise from a process which is not included in the $N(\mu,\sigma)$ model of the background noise. Cosmic ray hits and the like that affect single exposures in the set therefore show up clearly in the stacked image. The average fails in the presence of a single outlier in the sample.

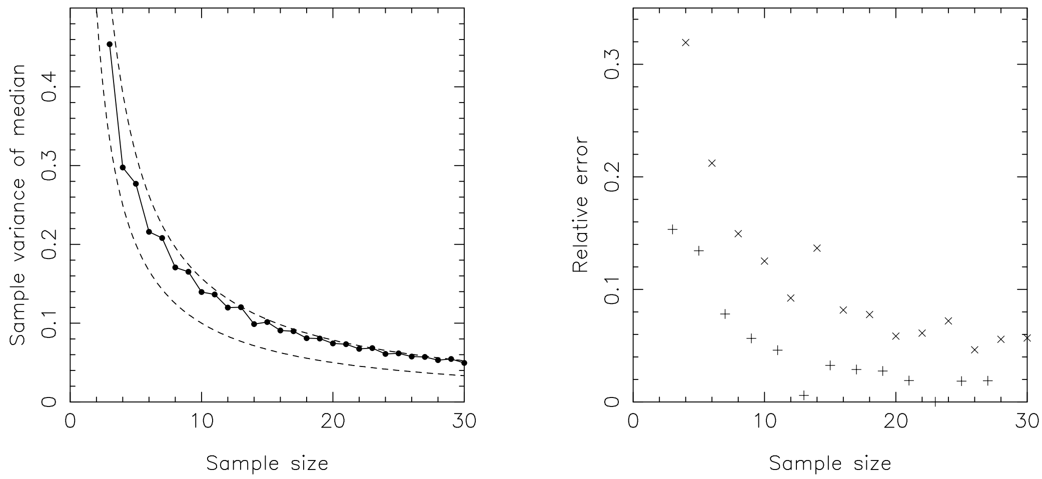

Figure 8.2.1: Sample variance of the median as a function of sample size (left panel). The lower dashed curve is the expected variance of the arithmetic mean, the upper dashed curve the asymptotic variance of the median. The right panel shows the relative error compared to the asymptotic variance. Crosses are for even sample sizes, pluses for odd sample sizes.

8.2.2 Median

For the computation of the sample median, the sample $\{x_i, i = 1 \ldots N\}$ is first sorted such that $x_1 < x_2 < \cdots < x_N$. The median is then the 50th percentile of the ordered set, i.e.

$$x_{\mathrm{med}} = x_{\frac{N+1}{2}} \quad \text{if } N \text{ is odd} \tag{10}$$

and

$$x_{\mathrm{med}} = \frac{1}{2}(x_{\frac{N}{2}} + x_{\frac{N}{2}+1}) \quad \text{if } N \text{ is even.} \tag{11}$$

This definition for the median for even-sized samples ensures that the median is an unbiased estimator of the population mean μ for a Gaussian error distribution or indeed any distribution that is symmetric about its mean.

The sample median has a larger variance than the sample average, i.e. it is less efficient as an estimator of the population mean. It is however robust against outliers in the sample since only the central one or two sample values are actually used for the computation. The median only fails if half or more of the sample values are outliers, i.e. if a given pixel is affected by a cosmic ray hit in half or more of the images in the stacking list.

While the sample distribution of the median (which is not Gaussian) can be written down easily, the computation of its moments and hence its variance is difficult. A scheme to compute it analytically has been described in [16] and exact values for a few sample sizes and a number of parent distributions are given in [21]. However, these are hardly useful for practical purposes. The asymptotic value for the ratio of the variances of mean and median (the *asymptotic relative efficiency* of the median, [7]) in the Gaussian case is

$$\sigma^2(\mathrm{Median}) = \frac{\pi}{2}\sigma^2(\mathrm{Mean}) = \frac{\pi}{2N}. \tag{12}$$

In Fig. 8.2.1, we determine the variance of the median from simulations by drawing for any given sample size 10000 samples from a standard normal distribution. The asymptotic value is approached from below, which makes it a conservative choice for an estimate of the median's variance. The relative error, defined as

$$\frac{\pi/2N - \hat{\sigma}^2(\text{Median})}{\pi/2N}, \tag{13}$$

is plotted in the right hand panel of Fig. 8.2.1. For even sample sizes, the relative error is significantly larger than for odd sizes because the variance is actually lower. For odd-sized samples, the error is below 10% for $N \geq 7$ ($N \geq 5$ according to [21]), whereas for even sized samples this threshold is not crossed until $N \gtrsim 12$. However, as mentioned above, the asymptotic value in Eq. (12) is conservative for any N and it is therefore used in MIME.

When each image has its own noise image $\sigma_i^2(x,y)$ (heteroscedastic case), then a weighted median can be defined as the point where cumulative weights is equal to $1/2$. Specifically, the weighted median of an ordered sample of N values x_i with weights w_i is

$$x_{\text{wmed}} = \begin{cases} \dfrac{x_j + x_{j+1}}{2}, & \text{if} \quad \sum_{i=1}^{j} w_i = 0.5 \\ x_j, & \text{if} \quad \sum_{i=1}^{j} w_i > 0.5 \quad \text{and} \quad \sum_{i=1}^{j-1} w_i < 0.5 \end{cases} \tag{14}$$

8.2.3 Min-max rejection

In the min-max rejection algorithm, the N_{low} lowest and N_{high} highest values are removed from the set and the arithmetic mean of the remainder is computed. For the ordered set x_i, the estimator is thus

$$x_{\text{min-max}} = \frac{1}{N - N_{\text{low}} - N_{\text{high}}} \sum_{i=N_{\text{low}}+1}^{N-N_{\text{high}}} x_i \,. \tag{15}$$

If the rejected values are interpreted as true outliers, i.e. as useless values or values that do not arise from the background noise distribution then one effectively only has $N - N_{\text{high}} - N_{\text{low}}$ observations of the distribution. The standard error of (15) is then

$$\sigma_{\text{minmax}} = \frac{1}{\sqrt{N - N_{\text{high}} - N_{\text{low}}}} \tag{16}$$

The extreme case of min-max rejection where all but the central one or two data values are rejected leads to the same estimate as the median; however, the sample error given by (16) differs from that given by (12). This is due to the fact that when applying the sample median, all values are considered good values.

8.2.4 Kappa-sigma clipping

In the min-max rejection algorithm, a fixed number of sample values are rejected, regardless whether they can be identified as outliers or not. If one wants to retain all "good" sample values and only reject true outliers, one has to employ an adaptive method which compares each sample value to the distribution of the entire sample.

In the $\kappa\sigma$ clipping algorithm, all values that deviate from the mean by more than κ standard deviations are rejected as outliers. Typically, $\kappa = 3$. The mean is usually estimated from the data, as is the standard deviation if no independent error estimate is available. One can introduce an iteration which stops once no more values are rejected.

Standard $\kappa\sigma$ clipping is not very efficient in removing low-significance outliers because the estimates of the mean and the standard deviation are themselves affected by the outliers. Using more robust estimators of location and scale, like the median and the inter-quartile range, improves the situation and makes $\kappa\sigma$ clipping a robust and easy-to-use adaptive outlier-rejection algorithm.

$\kappa\sigma$ clipping has one parameter, the clipping threshold κ. The value of κ is not critical if outliers are expected to lie far from the sample distribution, as is the case for cosmic ray hits. Weaker effects, such as airplane trails, may require fine-tuning of κ.

The method might fail if more than a quarter of the input images are affected by outliers at a given pixel because then the estimate of the inter-quartile range might be affected and cause outlier rejection to fail. The method also requires a sufficient number of input images to be able to obtain reasonably accurate estimates of the mean and standard deviation.

The variance of the estimator now varies from pixel to pixel depending on the number of values that are rejected as outliers. Since outliers are not drawn from the same distribution as the "good" data values, the cleaned sample is equivalent to a smaller sample drawn from the noise distribution. In the general heteroscedastic case, the variance is therefore estimated as

$$\sigma_{\kappa\sigma}^2 = \frac{1}{N_{\mathrm{good}}^2} \sum_{\mathrm{good}} \sigma_i^2 \,. \tag{17}$$

8.2.5 Mean-median ratio

MIME introduces another adaptive outlier rejection method that exploits the different robustness of the mean and the median. It will be seen that the method suffers from difficulties in setting the parameters that make a general use of the method problematic. Its inclusion here should be seen as experimental.

Both the (arithmetic) mean and the median are estimators of the expectation value of a random variable X based on a sample x_i. In the presence of outliers, the mean reacts very sensitively whereas the median is hardly affected as long as the number of outliers is less than half the sample size. On the other hand, the mean is the most efficient and hence the preferred estimator of the expectation value if the distribution of X is Gaussian.

We make use of the different robustness of the mean and the median in order to test for the presence of outliers in a given sample. The test statistic is their ratio, whose sample distribution can in principle be computed under the null hypothesis H_0: "The sample is drawn from a Gaussian distribution with mean μ and dispersion σ." The expectation of the ratio is 1, hence if the value of the ratio deviates significantly from 1, the null hypothesis is rejected and it is assumed that outliers are present. In this case, we remove the highest (lowest) value from the sample if the ratio is larger (smaller) than 1. The ratio is recomputed on the cleaned sample and the procedure repeated until no more values are rejected.

The standard error of the estimator is the standard error of the arithmetic mean of the pruned sample, as in (17).

Unfortunately, the sample distribution and the, say, 95% confidence limits for the mean-median ratio depend on the sample size (and due to the implication of the median it matters whether the sample size is odd or even) and the population mean and dispersion. Fig. 8.2.2 shows results from simulations demonstrating the complexity of the confidence limit.

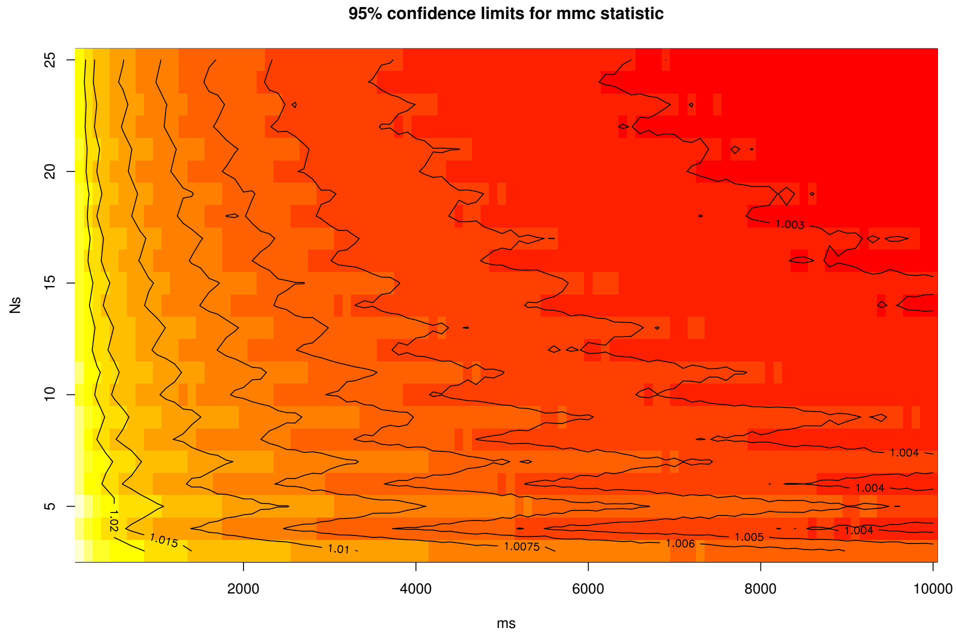

Figure 8.2.2: 95% confidence limits (single-tailed) for the mean-median ratio, as a function of population mean (variance is assumed to be equal to the mean) and sample size. The limits are derived from simulations, with 10000 samples drawn for each combination of m_s and N_s.

Table 8.3.0: Typical sky brightness (in mag/arcsec^2) at Paranal. UBVRI data are from [19], JHKL from [6]

Band	U	B	V	R	I	J	H	K_s	L
Magnitude	22.3	22.6	21.6	20.9	19.7	16.5	14.4	13.0	3.9

8.3 Sky subtraction algorithms

8.3.1 NIR images: pixel-based sky estimation

Sky emission in the near-infrared wavebands (Y, J, H, K) is mainly due to vibrational-rotational transitions of OH, as well as O_2 and H_2O in the K band. The sky emission is bright (cf. Table 8.3.0)) and varies significantly over space and time, so it has to be carefully subtracted before NIR images can be used for photometry of astronomical objects. As gravity waves pass through the upper atmosphere at an altitude of 80 to 105 km, both the strength and the structure of the sky emission vary over short time scales on the order of a few minutes to an hour. Spatially, variations are seen on angular scales of several degrees down to the arcmin level, which means that significant variation can be seen in typical NIR imager fields of view.

NIR observations typically consist of a large number of dithered exposures with integration times of a few seconds to avoid saturation by the sky background. This makes it possible to build a sky correction frame for each individual exposure by stacking a certain number of exposures taken within a time window such that the sky does not vary significantly. Due to the dithering, object flux can be effectively removed from the combined sky image.

Object rejection works only if the dithering shifts are larger than the largest object sizes in the field. If the observations target an object which covers a large fraction of the field, one typically observes a blank "sky field" some distance away from the target, alternating sky and object exposures in a certain pattern to achieve good temporal sampling of the sky behaviour.

The `mime` recipes implement a two-pass strategy that constructs a sky correction image for each exposure I_i ($i = 1,\ldots,N_{\rm exp}$) from either a set of dithered object exposures $\{\mathsf{I}_j\}$ taken immediately before and after in the observing sequence or a sequence of sky exposures $\{\mathsf{J}_j\}$ taken in close temporal vicinity to the exposure that is to be corrected. Algorithmically, there is no difference between these two approaches and we will simply refer to the set of exposures used for the correction as $\{\mathsf{I}_i\}$.

The basic assumption is that each exposure I_i can be represented as the sum of flux from astronomical objects T_i, a scaled version of a common sky structure S and noise ε:

$$\mathsf{I}_i = \mathsf{T}_i + c_i\mathsf{S} + \varepsilon\,. \tag{18}$$

The task of the sky subtraction is to estimate and isolate the scientifically interesting part T_i.

The scale c_i is estimated by the median μ_i of the image I_i. Each image can thus be considered an estimator of S after division by its median if the object contribution T_i can be removed. In order to decrease the statistical noise and to fill in gaps left by object removal, a pixelwise average of several images within a window is used:

$$\hat{\mathsf{S}}(x,y) = \frac{1}{N_{\rm good}}\sum_i \frac{\mathsf{I}_i(x,y) - \mathsf{T}_i(x,y)}{\mu_i}\,. \tag{19}$$

In practice, the subtraction of T_i in Eq. (19) is done by identifying and discarding images from the sum in which the pixel (x,y) under consideration is affected by flux from an astronomical object. The number of remaining images for a given pixel (i.e. those that contain only sky flux at this position) is then $N_{\rm good} \le N_{\rm exp}$.

The two-pass strategy is visualised in the left panel of Fig. 8.3.1. In the first pass, images are stacked (in the image frame) with a strict outlier rejection method (e.g. by forming the median of the images). The resulting set of sky estimates is subtracted from the exposures to form preliminary sky-corrected images.

In order to create a more sophisticated mask based on application of an object-detection algorithm, the first-pass sky-corrected images are stacked into a combined image which allows detection of fainter objects than would be possible from individual exposures (see Sect. 8.4).

The combined image is converted into a binary object mask by applying a threshold of a certain number of standard deviations above the background level. The mask is then morphologically opened with a 3×3 structuring element that removes all structures from the mask that are smaller than the structuring element; it is used here to remove single-pixel detections which are mostly extreme values of the background noise distribution. The masked regions are then grown in order to capture the low surface-brightness wings of the objects in the field (see Sect. 8.5).

8.3.2 Optical images: smooth background

Images taken in the visible wavebands usually show a background which varies slowly over the field of view. In cases where objects are fairly sparse and small so that the majority of pixels in an image carries pure sky

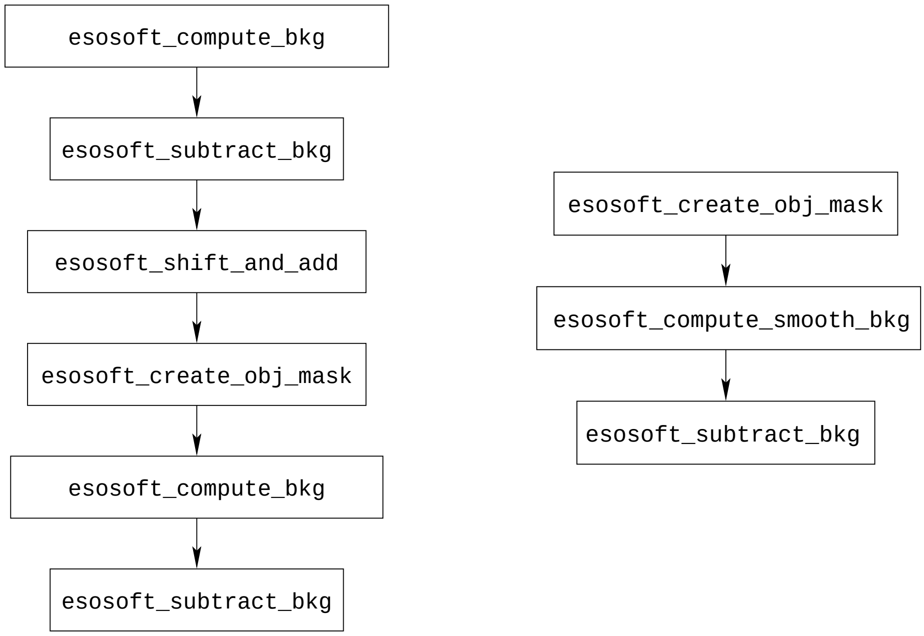

Figure 8.3.1: Sequence of recipes to be used for (left) the two-pass correction for NIR images and (right) the smooth correction for optical images.

signal it is possible and customary to obtain a smooth estimate of the sky background from the exposure to be corrected.

The recipe `mime_compute_smooth_bkg` estimates the sky background by fitting orthogonal Legendre polynomials in pixel coordinates x and y. The procedure for the background correction for optical images is as follows (see right panel in Fig. 8.3.1:

1. detect objects and create an object mask (`mime_create_obj_mask`)
2. fit a smooth background model (`mime_compute_smooth_bkg`)
3. subtract the background model (`mime_subtract_bkg`)

It is applied to each exposure individually.

The first step attempts to clean the set of pixels over which the polynomial is fitted from object contamination (see Sect. 8.5).

Next, the noisy sky background is regularised, and partially denoised, by means of a least-squares fit with the polynomials in two variables.

The least-squares fit is computed using the Wiener filter; the method is also known as the Tikhonov regularisation or the ridge regression (e.g. [3]). The method requires the solution of the normal equations with the diagonal incremented by a constant reflecting the conditioning of the problem and the amount of noise (e.g. the variance of the Gaussian noise).

The resulting polynomial approximation is a linear combination of the basis polynomials. For reasons of numerical stability, the basis consists of the tensor products of the Legendre polynomials. Specifically, each basis polynomial has the form $P_i(x)P_j(y)$, where P_i is the Legendre polynomial of degree i rescaled according to the size of the image. When used with rectangular images, the Legendre polynomials in the variable x have a different scaling than those in the variable y. The most practical basis consists of all tensor products $P_i(x)P_j(y)$ with $i+j \leqslant k$, where k is a fixed integer.

8.4 Stacking of sky-subtracted images

In order to create a more sophisticated mask based on application of an object-detection algorithm, the first-pass sky-corrected images are stacked into a combined image which allows detection of fainter objects than would be possible from individual exposures.

As the purpose of stacking at this stage is only to create an object mask, distortion correction based on a precise astrometric solution is not necessary. The recipe `mime_shift_and_add` uses simple integer pixel shifts to align the images, hence no rebinning of the images is required. The shifts are obtained from the headers of the input images.

For most instruments, the telescope offsets applied during the execution of an observing block are stored in the fits header as `HIERARCH ESO SEQ CUMOFFSETX` and `HIERARCH ESO SEQ CUMOFFSETY`. Note that these are *telescope* offsets and therefore have opposite sign to the shifts of the image frames.

The situation is illustrated in Fig. 8.4.1. The size of the individual exposures is $n_x \times n_y$, the size of the stacked image encompassing all the exposures is $N_x \times N_y$. The position of the lower left corner of the reference frame within the stacked frame is then:

$$(1,1)_{\mathrm{ref}} \longmapsto (1-\min_i \Delta x_i\,,\, 1-\min_i \Delta y_i)_{\mathrm{stack}} \tag{20}$$

and the position of exposure i is

$$(1,1)_{\mathrm{exp},i} \longmapsto (\Delta x_i + 1 - \min_i \Delta x_i\,,\, \Delta y_i + 1 - \min_i \Delta y_i)_{\mathrm{stack}} \tag{21}$$

The size of the output image is

$$(N_x, N_y) = (n_x + \max_i \Delta x_i - \min_i \Delta x_i\,,\, n_y + \max_i \Delta y_i - \min_i \Delta y_i) \tag{22}$$

To obtain the value of pixel (X,Y) in the stacked image, the average of the values of the input pixels shifted to that position is computed. For exposure i, the relevant pixel is

$$(x_i, y_i) = (X + \min_j \Delta x_j - \Delta x_i\,,\, Y + \min_j \Delta y_j - \Delta y_i) \tag{23}$$

8.5 Creation of object masks

The recipes `mime_compute_bkg` and `mime_compute_smooth_bkg` permit the use of object masks when creating the background images. Such an object mask can be created from the stacked image with the recipe `mime_create_obj_mask`, which applies an object detection algorithm to the image. The power of this approach over relying on the robust outlier rejection methods described in Sect. 8.2 lies in the fact that it detects

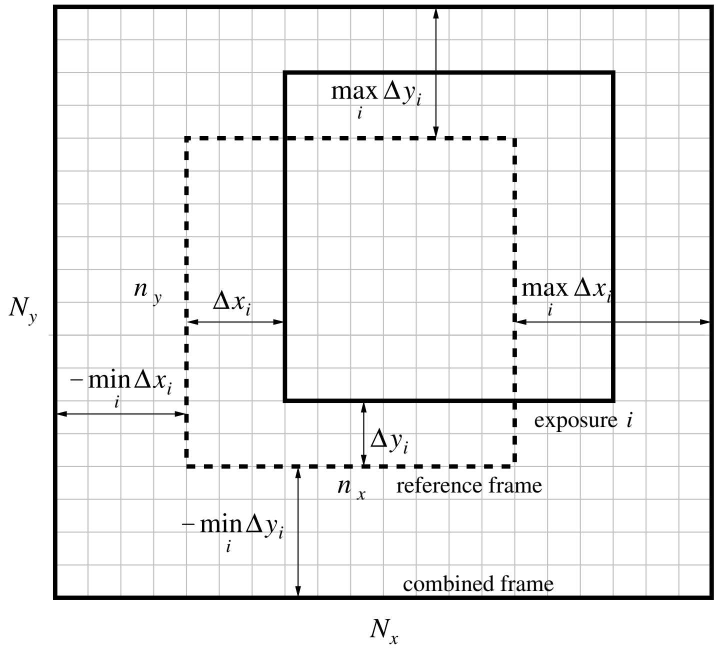

Figure 8.4.1: Position of an individual exposure within the stacked image frame. The shifts $(\Delta x_i, \Delta y_i)$ are given with respect to a reference position which is here represented by a corresponding reference frame.

coherent structures and is thus able to identify wings of objects where the flux from the object is on the order of the background noise.

The algorithm first determines the background level μ of the input image by its median, and the noise σ via the robust inter-quartile range ($\sigma = 1.349\,\mathrm{IQR}$ for a Gaussian distribution). The threshold is then set to

$$t = \mu + k\sigma\,, \tag{24}$$

where the k is a recipe parameter and defaults to 3.0. The initial object mask is then created with value 1 at all pixels with a flux above the threshold, and 0 elsewhere.

The mask is treated in two steps: First it is morphologically opened with the structuring element

$$\mathsf{K}_{\text{morph}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{25}$$

The effect of this operation is that masked regions that are smaller than the structuring element (3×3 pixels) are removed from the mask. In most cases where a single isolated pixel is at a value higher than the threshold will be from the normal noise distribution and by unmasking these pixels we avoid truncating the Gaussian distribution.

Next, the mask is grown (dilated) by additionally masking a certain number r_{grow} of pixels around a masked region. This is done by applying a kernel of size $(2r_{\text{grow}}+1) \times (2r_{\text{grow}}+1)$ with values

$$\mathsf{K}_{\text{grow},ij} = \max\left(0, 1 - \frac{\sqrt{(i-i_0)^2 + (j-j_0)^2}}{r_{\text{grow}}}\right)\,, \tag{26}$$

where $i_0 = j_0 = (2r_{\text{grow}}+1)/2$. The growing radius r_{grow} is a recipe parameter and defaults to 5 pixels.

A Installation

The MIME/REFLEXpipeline is distributed as a kit which contains all the software required as well as demonstration data for the three REFLEX workflows.

After downloading the MIME/REFLEX/ distribution kit, unpack it into an empty directory:

```
tar xvf mime-reflex-110228.tar.gz
```

The newly created directory `Reflex` contains several installation scripts which serve to install all or part of the components of the kit:

- `install_mime_reflex`: This script installs the MIME pipeline, REFLEX, ESOREX, GASGANO as well as the required libraries (in particular the CPL library).
- `install_terapix` installs binaries of the programmes `sextractor`, `scamp` and `swarp`. This can be skipped if recent versions of the programmes are already installed on your computer.
- `install_demodata` installs one or all of the demo data sets included in the kit.

The entire intallation procedure is covered by the script `install_all`. To install the MIME/REFLEX kit, change to the directory `Reflex` and launch the installation script:

```
cd Reflex
./install_all
```

This installs all software modules required to run the MIME pipeline modules via ESOREX, GASGANO and REFLEX into the subdirectory `software` (except for GASGANO which insists on being installed in the user's home directory). In order to avoid the newly installed libraries and executables from interfering with the normal operation of your computer, it is recommended to add the directories to the paths only when working with the MIME/REFLEX environment. If you use the `bash` shell, do

```
export PATH=/path/to/Reflex/software/bin:$PATH
export LD_LIBRARY_PATH=/path/to/Reflex/software/lib:$LD_LIBRARY_PATH
```

If you use the `csh` or `tcsh` shell, do

```
setenv PATH /path/to/Reflex/software/bin:$PATH
setenv LD_LIBRARY_PATH /path/to/Reflex/software/lib:$LD_LIBRARY_PATH
```

The script asks which binary version of the Terapix programmes to install:

- `terapix-fc11-i686.tar.gz`: appropriate for 32-bit systems
- `terapix-sl53-x86_64.tar.gz`: appropriate for 64-bit systems

Finally, the script asks whether to install demo data sets. The included data sets are:

- `data_WFI`: five *R*-band exposures of the GOODS field obtained with WFI@2.2m. This data set is meant to be reduced with the workflow `mime_mosaicing_optical.xml`, demonstrating the smooth background subtraction.
- `data_HAWKI`: fifteen *J*-band exposures of the cluster of galaxies Abell 1689 obtained with HAWK-I. This data set is meant to be reduced with the workflow `mime_mosaicing_nir_dither.xml`, demonstrating the two-pass method for sky subtraction.
- `data_HAWKI_2`: 26 K_s-band exposures of the spiral galaxy NGC 7793 with six exposures of a nearby empty region obtained during the same OB. These data are meant to be reduced with the workflow `mime_mosaicing_nir_sky.xml`, demonstrating the NIR sky subtraction using dedicated sky exposures.

The demo data are installed into the directory `data/mime`. They include scripts which demonstrate how to call the recipes directly via ESOREX.