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

AMBER Pipeline User Manual

VLT-MAN-ESO-19500-4221

Issue 4.4.4

Date 2023-05-19

	Name	Date	Signature
Prepared:	A. Gabasch	2023-05-19	
Approved:	P. Ballester		
Released:	M. Peron		

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	03/05/2007	All	First version
1.1	01/04/2008	All (restructured)	Update to 2.4.3
1.2	01/06/2008	3, 7.6	Update to 2.4.4
1.3	01/08/2009	3, 7.5, 7.6	Update to 2.5.2
2.9.1	01/05/2010	6.1, 7.6, 7.5, 7.7 , 7.8	Update to 2.9.1
3.0.0	01/08/2010	3, 6.2, 7, 6.9, 7.9, A	Update to 3.0.0
3.0.1	01/08/2010	3, 7.5, 7.6, 7.7, 7.8, D	Update to 3.0.1
3.1.0	09/08/2010	3, 7.5	Update to 3.1.0
3.3.0	01/04/2011	3, 7.5	Update to 3.3.0
4.1.2	01/04/2012	1.2, 3, 7.4 7.5	Update to 4.1.2
4.2.2	01/04/2013	3, 7.11	Update to 4.2.2
4.3.1	01/12/2013	3, 7.1	Update to 4.3.1
4.3.2	22/08/2014	3, 7.6	Update to 4.3.2
4.3.4	01/04/2016	3,	Update to 4.3.4
4.3.9	01/05/2019	3,	Update to 4.3.9
4.4.0	01/06/2020	3,	Update to 4.4.0
4.4.3	19/05/2022	3,	Update to 4.4.3
4.4.4	19/05/2023	3,	Update to 4.4.4

This page was intentionally left blank

Contents

1 Introduction **11**

1.1 Purpose . . . 11

1.2 Acknowledgments . . . 11

1.3 Scope . . . 11

1.4 Reference and applicable documents . . . 11

1.5 Abbreviations and acronyms . . . 13

2 Overview **14**

3 What's new **15**

3.1 What's new in pipeline release 4.4.4 . . . 15

3.2 What's new in pipeline release 4.4.3 . . . 15

3.3 What's new in pipeline release 4.4.0 . . . 15

3.4 What's new in pipeline release 4.3.9 . . . 15

3.5 What's new in pipeline release 4.3.4 . . . 15

3.6 What's new in pipeline release 4.3.2 . . . 15

3.7 What's new in pipeline release 4.3.1 . . . 15

3.8 What's new in pipeline release 4.2.2 . . . 16

3.9 What's new in pipeline release 4.1.2 . . . 16

3.10 What's new in pipeline release 3.3.0 . . . 16

3.11 What's new in pipeline release 3.1.0 . . . 16

3.12 What's new in pipeline release 3.0.1 . . . 17

3.13 What's new in pipeline release 3.0.0 . . . 17

3.14 What's new in pipeline release 2.8.0 . . . 17

3.15 What's new in pipeline release 2.5.2 . . . 18

3.16 What's new in pipeline release 2.4.4 . . . 18

3.17 What's new in pipeline release 2.4.3 . . . 19

4 AMBER Instrument Description **20**

4.1 Overview . . . 20

4.2 Observation Modes 20
4.3 Angular Resolution 21
4.4 Flux Sensitivity 21
4.5 Field Of View 22

5 Instrument Data Description 23
5.1 Calibration Frames 23
5.2 Scientific Frames 25

6 Data Reduction 27
6.1 Reduction Cascade 27
6.2 AMBER pipeline recipes 27
6.3 The set of frames (SOF) 28
6.4 Recipe configuration 29
6.5 Running a recipe using EsoRex 30
6.6 Running a complete data reduction using GASGANO 31
6.7 Creating a primary FITS-Image of the Channels 32
6.8 Calculating the Pixel-To-Visibility-Matrix 33
6.9 Calculating the scientific results 35

7 Pipeline Recipes Interfaces 37
7.1 amber-raw-to-fitsimage 37
7.2 amber-detector 38
7.3 amber-beampos 40
7.4 amber-p2vm 42
7.5 amber-SciCal 44
7.6 amber-selector 48
7.7 amber_trf 52
7.8 amber_calibrate 53
7.9 amber_oimerge 54
7.10 amber-ascii-export 54
7.11 amber_spectral_calibration 56

A Installation **57**

B Algorithms **58**

B.1 General Algorithms . . . 58

B.1.1 Dark subtraction . . . 58

B.1.2 Bad Pixel Map and Flat Field Generation . . . 58

B.1.3 Particle Events . . . 61

B.1.4 Electronic Bias . . . 64

B.1.5 Pixel Statistics, Pixel Bias, and Photon Noise . . . 65

B.1.6 Dark Current . . . 68

B.1.7 Bad Pixel Map . . . 68

B.1.8 Flatfield Map . . . 70

B.1.9 Flat field correction . . . 72

B.2 AMBER-specific Algorithms . . . 72

B.2.1 P2VM calculation . . . 72

C Changes in the Amdlib **74**

C.1 Amdlib changes in the amber_p2vm recipe . . . 75

C.2 Amdlib changes in the amber_SciCal recipe . . . 76

D QC1 Dictionary **79**

D.1 QC parameter of the amber_SciCal recipe . . . 79

D.2 QC parameter of the amber_selector recipe . . . 81

D.3 QC parameter of the amber_trf recipe . . . 83

D.4 QC parameter of the amber_calibrate recipe . . . 85

E Troubleshooting Guide **87**

List of Tables

6.7.1 *Input files for the amber_raw_to_fitsimage recipe.* . . . 32
6.8.1 *Input files for the amber_p2vm recipe.* . . . 33
6.9.1 *Input files for the amber_SciCal recipe.* . . . 35
7.1.1 *Input files for the amber_raw_to_fitsimage recipe.* . . . 37
7.1.2 *Output of the amber_raw_to_fitsimage recipe.* . . . 37
7.2.1 *Input files for the amber_detector recipe.* . . . 38
7.2.2 *Product of the amber_detector recipe.* . . . 38
7.2.3 *QC values written by amber_detector recipe.* . . . 39
7.3.1 *Input files for the amber_beampos recipe.* . . . 40
7.3.2 *Product of the amber_beampos recipe.* . . . 40
7.3.3 *QC values written by the amber_beampos recipe.* . . . 40
7.3.4 *QC value prefixes written by the amber_beampos recipe.* . . . 41
7.4.1 *Input files for the amber_p2vm recipe.* . . . 42
7.4.2 *Product of the amber_p2vm recipe.* . . . 42
7.4.3 *QC values written by the amber_p2vm recipe.* . . . 43
7.5.1 *Frame selection criteria of the amber_SciCal recipe: selection is done on the piston value* . . . 45
7.5.2 *Input files for the amber_SciCal recipe.* . . . 46
7.5.3 *Products of the amber_SciCal recipe.* . . . 46
7.6.1 *Input files for the amber_selector recipe.* . . . 48
7.6.2 *Example of the file /tmp/amber_exclude.txt.* . . . 50
7.7.1 *Input files for the amber_trf recipe.* . . . 52
7.7.2 *Product of the amber_trf recipe.* . . . 52
7.8.1 *Input files for the amber_calibrate recipe.* . . . 53
7.8.2 *Product of the amber_calibrate recipe.* . . . 53
7.9.1 *Input files for the amber_oimerge recipe.* . . . 54
7.9.2 *Product of the amber_oimerge recipe. Please note, that the filename chances to merged_unknown.fits if the waveband can not be identified properly, i.e. if the QC BAND header parameter is missing.* 55
7.9.3 *QC values inherited by the amber_oimerge recipe.* . . . 55
7.11.1 *Input files for the amber_spectral_calibration recipe.* . . . 56

7.11.2 *Product of the amber_spectral_calibration recipe.* 56
7.11.3 *QC values written by the amber_spectral_calibration recipe.* 56
B.1.0 Parameters for the bad pixel map generation 70
D.1.0 *QC values written by the amber_SciCal recipe.* 80
D.2.0 *QC values written by the amber_selector recipe.* 82
D.3.0 *QC values written by the amber_trf recipe.* 84
D.4.0 *QC values written by the amber_calibrate recipe.* 86

1 Introduction

1.1 Purpose

The AMBER pipeline is a subsystem of the *VLTI Data Flow System* (DFS). Its target user is ESO *Data Flow Operations* (DFO) in the generation of master calibration data, in the reduction of scientific exposures and in the data quality control. It should also serve as a quick look tool for *Paranal Science Operations* (PSO). Additionally, the AMBER pipeline recipes are made public to the user community, to allow a more personalized processing of the data from the instrument.

This manual is a complete description of the data reduction recipes used by the AMBER pipeline, reflecting the status of the AMBER pipeline as of version 4.4.4.

1.2 Acknowledgments

The software package on which the AMBER pipeline is based on was in large parts developed by the AMBER Consortium, and it is still the foundation of the current AMBER interferometry data reduction.

If you use the AMBER data reduction software, JMMC (Jean-Marie Mariotti Center) asks you to add this sentence in the acknowledgement section of your articles:

```
This research has made use of the \texttt{AMBER data reduction package}
of the Jean-Marie Mariotti Center \footnote{Available at
http://www.jmmc.fr/amberdrs}.
```

Furthermore, as user of the AMBER data reduction software, you are kindly requested to cite the two following refereed papers in the data reduction section of your article(s):

Tatulli et al. (2007, available at http://cdsads.u-strasbg.fr/abs/2007A&A...464...29T) and Chelli et al. (2009, available at http://cdsads.u-strasbg.fr/abs/2009A&A...502..705C).

1.3 Scope

This document describes the AMBER pipeline used at ESO-Garching and ESO-Paranal for the generation of master calibration data and data quality control.

1.4 Reference and applicable documents

[1] ESO. *AMBER Data Reduction Plan*. VLT–PLA–AMB–15830–6004.

[2] ESO. *VLT Data Flow System Specifications for Pipeline and Quality Control*. VLT-SPE-ESO-19600-1233. 14

[3] ESO. *Deliverables Specification*. VLT-SPE-ESO-19000-1618 (2.0).

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

[5] ESO. *ESO DICB – Data Interface Control Document*. GEN-SPE-ESO-19400-0794 (3.0).

[6] ESO. *Common Pipeline Library User Manual*. VLT-MAN-ESO-19500-2720. 29

[7] ESO. *Gasgano User's Manual*. VLT-PRO-ESO-19000-1932. 14, 23, 28, 29, 30

[8] ESO. *AMBER Calibration Plan*. VLT–PLA–AMB–15830–0004.

[9] ESO. *AMBER product data specification*. VLT-SPE-AMB-15830-3984. 19

[10] ESO. *AMBER data reduction software design description document*. VLT-TRE-AMB-15830-4107.

1.5 Abbreviations and acronyms

ADF	Aperture Definition File
ADM	Aperture Definition in mm
AMBER	Astronomical Multi BEam combineR
AT	Auxilliary Telescope
CPL	Common Pipeline Library
CRV	Spectral curvature model
DFS	Data Flow System
DMD	Data Management Division
DO	Data Organiser
DRS	Data Reduction System
ESO–MIDAS	ESO's Munich Image Data Analysis System
FITS	Flexible Image Transport System
HR	High Resolution
ICS	Instrument Control Software
IDS	Inverse Dispersion Solution
IRAF	Image Reduction and Analysis Facility
IWS	Instrument WorkStation
LR	Low Resolution
MMU	Mask Manufacturing Unit
MOS	Multi Object Spectroscopy
MR	Medium Resolution
P2VM	Pixel To Visibility Matrix
PAF	VLT PArameter File
PWS	Pipeline WorkStation
QC	Quality Control
RB	Reduction Block
RBS	Reduction Block Scheduler
SAO	Smithsonian Astrophysical Observatory
SNR	Signal to Noise Ratio
SOF	Set Of Frames
TCS	Telescope Control Software
UT	Unit Telescope
VLT	Very Large Telescope
VLI	Very Large Telescope Interferometer
WCS	World Coordinate System

2 Overview

The AMBER pipeline is a collection of data reduction recipes administrated by the higher level programs of the *VLT Data Flow System* (DFS [2]).

The *VLT Data Flow System* provides an environment where the definition and the execution of the data reduction steps can be made entirely automatic. The fundamental DFS software components making up the required functionality are the *Data Organizer* (DO) and the *Reduction Block Scheduler* (RBS). The DO applies some defined instrument-dependent rules for the classification of the incoming frames, for their association to suitable calibration data, and for launching the appropriate recipe for their standard reduction. The product of the DO is the *Reduction Block* (RB), an ASCII file containing all the necessary information for the specific data reduction. The RB is executed by the RBS, which calls the assigned pipeline recipe with the appropriate parameters and makes the data reduction products available to the *Data Handling Server* (DHS).

The AMBER pipeline recipes can also be used outside the pipeline context, as stand-alone programs. In this way they may be easily used by astronomers at their home institutes, or taken as building blocks for different data reduction strategies. However, the pipeline recipes, being designed to operate within the DFS environment, do not repeat the checks already performed by the DO. In other words, data classification and appropriate association are not re-verified within a recipe. The *Gasgano* [7] data browser, capable of classifying the data frames in the same way as the DO, greatly assists in the administration of the different data frames.

An overview of the improvements achieved with this new pipeline version is given in Section 3. The AMBER instrument and the different types of AMBER raw frames are briefly described in Sections 4 and 5, while the usage of the available reduction recipes is presented in Section 6. In Section 7 we present the pipeline recipe interface. In Appendix A the installation of the AMBER pipeline recipes is described, whereas a more detailed descriptions of the data reduction algorithms used by the individual pipeline recipes can be found in Appendix B. Appendix C gives a detailed comparison between the products derived by the previous and the recent pipeline version.

3 What's new

3.1 What's new in pipeline release 4.4.4

- The pipeline has been updated to work with cpl version 7.3

3.2 What's new in pipeline release 4.4.3

- The pipeline has been updated to work with cpl version 7.2

3.3 What's new in pipeline release 4.4.0

- The pipeline has been updated to work with cpl version 7.1.3
- The pipeline latex files encoding were standardized.

3.4 What's new in pipeline release 4.3.9

- The pipeline has been updated to work with cpl version 7.1.1
- The pipeline installation routines were standardized.

3.5 What's new in pipeline release 4.3.4

- The pipeline has been updated to work with cpl version 7.
- The pipeline installation routines were standardized.

3.6 What's new in pipeline release 4.3.2

- A new parameter `AverageFrames` has been added to the recipe *amber_selector*. If activated, the recipe will average all frames after performing the selection and save the averaged frame as recipe product.

3.7 What's new in pipeline release 4.3.1

- The recipe *amber_raw_to_fitsimage* has been rewritten. It creates two products with the DO.CATG AMBER_RAW_CUBE and AMBER_RAW_SUM, the first transforms and saves the raw file into a cube structure whereas the second contains the sum of the cube. Moreover, the automatic triggering of the recipe was removed from the Paranal oca files.
- Various modifications to the autotool build system have been performed in order to support as many platforms as possible.

3.8 What's new in pipeline release 4.2.2

- A new recipe *amber_spectral_calibration* has been added. The recipe calculates the real Y offset between the different photometric and the interferometric channels in the 3-telescope mode. It uses the same algorithm as the *amber_p2vm* recipe. Moreover, a amdlib-bug present on 32 bit computers was fixed in the above described calculation/recipes.

3.9 What's new in pipeline release 4.1.2

- The pipeline has been updated to amdlib version 3.0.3.
- An acknowledgment to JMMC has been added to the various pipeline products.
- The QC parameter <QC USEDVALUE ANDselection> of the amber_SciCal recipe has been renamed to <QC USEDVALUE AND>
- The midas plotting program amber_OI_FITS_plot.prg / amber_OI_FITS_plot.sh has been removed as incompatible with the new amdlib product structure
- New calibratorlist:

```
FILE                                    PRO.CATG
calibrator_database_amber_H.fits        CALIB_DATABASE_H
calibrator_database_amber_J.fits        CALIB_DATABASE_J
calibrator_database_amber_K.fits        CALIB_DATABASE_K
```

3.10 What's new in pipeline release 3.3.0

- two classification tags are added to the amber_SciCal recipe to identify the dark frames: AMBER_DARK_CALIB and AMBER_DARK_SCIENCE
- two classification tags are added to the amber_SciCal recipe to identify the sky frames: AMBER_SKY_CALIB and AMBER_SKY_SCIENCE
- An error in the check for the overlapping wavelength interval of the transfer function and science observation has been fixed.

3.11 What's new in pipeline release 3.1.0

- The calibrator database and the tagging of the calibrator database has changed. The DO category has been changed from
  `AMBER_CALIB_DATABASE_J` to `CALIB_DATABASE_J`,
  `AMBER_CALIB_DATABASE_H` to `CALIB_DATABASE_H`, and
  `AMBER_CALIB_DATABASE_K` to `CALIB_DATABASE_K`

3.12 What's new in pipeline release 3.0.1

- New Quality Control (QC) parameters for the amber_SciCal, amber_selector, amber_trf, and amber_calibrate recipe. The standard deviation of the squared visibilities with respect to the frames (and averaged over the various wavelength bins) for all wavelength bins is calculated and added as QC parameter. See Appendix D for the new QC parameters.

3.13 What's new in pipeline release 3.0.0

- A new recipe amber_oimerge has been written. It can be used to merge OI-fits files of the following PRO CATG:
  SCIENCE_REDUCED SCIENCE_REDUCED_FILTERED CALIB_REDUCED
  CALIB_REDUCED_FILTERED AMBER_TRF_J AMBER_TRF_H AMBER_TRF_K
  SCIENCE_CALIBRATED (see section 7.9).

- The recipe amber_calibrate has been expanded in order to calibrate science observations even if the wavelength coverage is not identical with the one of the transfer function. In this case the recipe tries to find a common overlapping wavelength range and derives calibrated visibilities for the overlap.

- In the amber_selector recipe the possible values for the `selection-method` method have been renamed (see section 7.6).

- A function counting the number of zeroframes in the p2vm raw files has been implemented. The number of frames with zero values is written as QC parameter to the p2vm file: HIERARCH ESO QC ZEROFRAMES

3.14 What's new in pipeline release 2.8.0

- A new recipe amber_trf has been written in order to calculate the transfer function of a calibrator (see section 7.7).

- A new recipe amber_calibrate has been written in order to calculate calibrated squared visibilities and closure phases by applying a transfer function (see section 7.8).

- The recipe amber_SciCal (see section 7.5) has been rewritten. The recipe can automatically perform a frame selection and derive transfer functions (for calibrators) as well as calibrate squared visibilities and closure phases (for science targets). New Quality Control parameters are derived and added to each product.

- The recipe amber_ascii_export has been updated.

- Flatfield and Bad Pixel Mask have been updated (2009-11-05)

- The pipeline has been upgraded to CPL 5.1

3.15 What's new in pipeline release 2.5.2

- The Pipeline has been upgraded from amdlib version 2.0 to amdlib version 2.2:
  - Correction of bugs happening when appending OI-FITS files and memory leaks in the C code part.
  - This version corrects the problem of wrong estimate of spectral resolution elements invalidated by the spectral displacement between the photometric and interferometric beams. This caused a of loss of valid spectral channels during the P2VM computation, only noticeable in Low resolution JHK mode, when the spectral displacement was large, which, to our knowledge, occurred only between September 2007 and January 2008. However, this bug correction may be beneficial for all observations, providing a few (1 to 2) more wavelengths available in the reduced datasets, in the long-wavelength end of LR and HR modes, and short-wavelength end in MR modes.
- The Flatfield and Bad Pixel Mask has been updated with data taken on 2008-09-14
- About 30 new quality control parameter were added to the amber_SciCal (see section 7.5) and amber_selector (see section 7.6) recipe.
- The recipe amber_ascii_export has been updated.
- The pipeline recipe amber_SciCal can now handle up to 5000 science files in the SOF. If more are given, a warning will be issued and only the firsts 5000 files will be reduced.
- The recipe amber_raw_to_fitsimage has been updated in order to have an ESO compliant product header.
- Improved recipe stability.

3.16 What's new in pipeline release 2.4.4

- The amber_selector recipe has been updated and new selection methods have been implemented. At the moment 10 different methods can be used to select the frames:
  - "First x Frames"
  - "Fringe SNR > x"
  - "Fringe SNR percentage x"
  - "Flux > x"
  - "Flux percentage x"
  - "Exclude Frames by ASCII File"
  - "Include Frames by ASCII File"
  - "IO-Test: no filtering"
  - "Absolute piston value < x"
  - "Absolute piston value percentage x"

  Moreover, after selecting a method one can impose a selection threshold for every baseline by defining the variables X1, X2, and X3 as well as the way how the single thresholds should be applied to a frame by setting the Boolean variable ANDselection. See section 7.6 for details.

3.17 What's new in pipeline release 2.4.3

- The major change in the AMBER pipeline is the integration of the new amdlib version 2.0 (cmm version 2.29). The new amdlib version has improved error handling, corrects software bugs and extracts the p2vm matrix for each individual waveband (J,H,K).
- The following new quality control (QC) parameters are calculated and added for every baseline XX to the product of the amber_p2vm recipe:
  QC.P2VM.J.VISXX – QC.P2VM.J.ERRVISXX – QC.P2VM.H.VISXX – QC.P2VM.H.ERRVISXX – QC.P2VM.K.VISXX – QC.P2VM.K.ERRVISXX.
- The AMBER_SPECTRUM extension has been added in the amber_SciCal recipe product file.
- The AMBER product data specifications [9] have changed (see also Appendix C)
- Update of the amber_detector recipe to support the new detector 247. A new flatfield and bad pixel mask has been derived and included in the distribution. Due to the detector upgrade for data taken before 2007-09-13 the old bad pixel map (BadPixelMap_oldDetector.fits) and the old flatfield (FlatFieldMap_oldDetector.fits) must be used. From 2007-09-13 until 2008-02-07 the following flatfield and bad pixel map should be used: BadPixelMap_oldDetector1.fits and FlatFieldMap_oldDetector1.fits. Data taken after 2008-02-07 should use the newest calibration files (BadPixelMap.fits, FlatFieldMap.fits). All fits-files are automatically installed by the pipeline installation procedure (see Appendix A).
- The pipeline has been upgraded to CPL 4.1.
- The Midas plotting program amber_OI_FITS_plot.prg has been adapted to the output of the new amdlib 2.0
- Integration of the new amber_ascii_export provided by Klara Shabun and adapted to the new amdlib 2.0
- amber_raw_to_fitsimage has been expanded in order to be also able to handle detector calibration files.

Details about the changes in the product files due to the amdlib upgrade can be found in Appendix C.1 for the amber_p2vm and in Appendix C.2 for the amber_SciCal recipe.

4 AMBER Instrument Description

AMBER has been developed under ESO contract by the AMBER Consortium, headed by the Laboratoire d'Astrophysique de Grenoble with contributions from Observatoire de la Côte d'Azur, Laboratoire Universitaire d'Astrophysique de Nice, MPI fuer Radioastronomie - Bonn and Osservatorio Astrofisico di Arcetri.

The instrument has been made available to the community and started Service Mode Operations in Paranal on October 1st, 2005.

In this chapter a brief description of the AMBER instrument is given. A more complete documentation can be found in the AMBER User Manual, which can be retrieved from:

`http://www.eso.org/instruments/amber/`

4.1 Overview

AMBER is the first-generation near-infrared three-way beam combiner. This instrument (see Figure 4.1.1) provides simultaneously spectrally dispersed visibility for three baselines and a closure phase at three different spectral resolutions. AMBER has been designed to investigate the milli-arcsec surrounding of astrophysical sources like young and evolved stars, active galactic nuclei and to detect exoplanets.

The main new feature of AMBER if compared to other interferometric instruments is the simultaneous use of modal filters (optical fibers) and a dispersed fringe combiner using a spatial coding (see Figure 4.1.2). The AMBER team has therefore carefully investigated a data processing strategy for this instrument and is providing a new type of data reduction method.

4.2 Observation Modes

AMBER offers three different modes of resolutions and three different bands in the following combinations:

Resolution Mode	Abbreviation	Description
Low Resolution	JHK-LR	The J, H and K bands are covered simultaneously
Medium Resolution	JH-MR	The J and H bands are covered simultaneously
Medium Resolution	HK-MR	The H and K bands are covered simultaneously
High Resolution	J-HR	The J band is covered
High Resolution	H-HR	The H band is covered
High Resolution	K-HR	The K band is covered

Please note that the spectral coverage of the high spectral resolution does not cover the entire bands.

Depending on the type of science required, one can choose to observe with AMBER in a classical way or in differential mode. The differential mode involves slight differences in the observing procedures but allows to obtain higher precision on the differential phase. The expected accuracy are typically 1% for the high sensitivity mode, 0.1% for the high precision mode, and 0.01% for the phase in the high precision mode in the differential configuration. This can be used either to obtain angular information on very unresolved objects or to aim at very high accuracy like in the extrasolar planets programs.

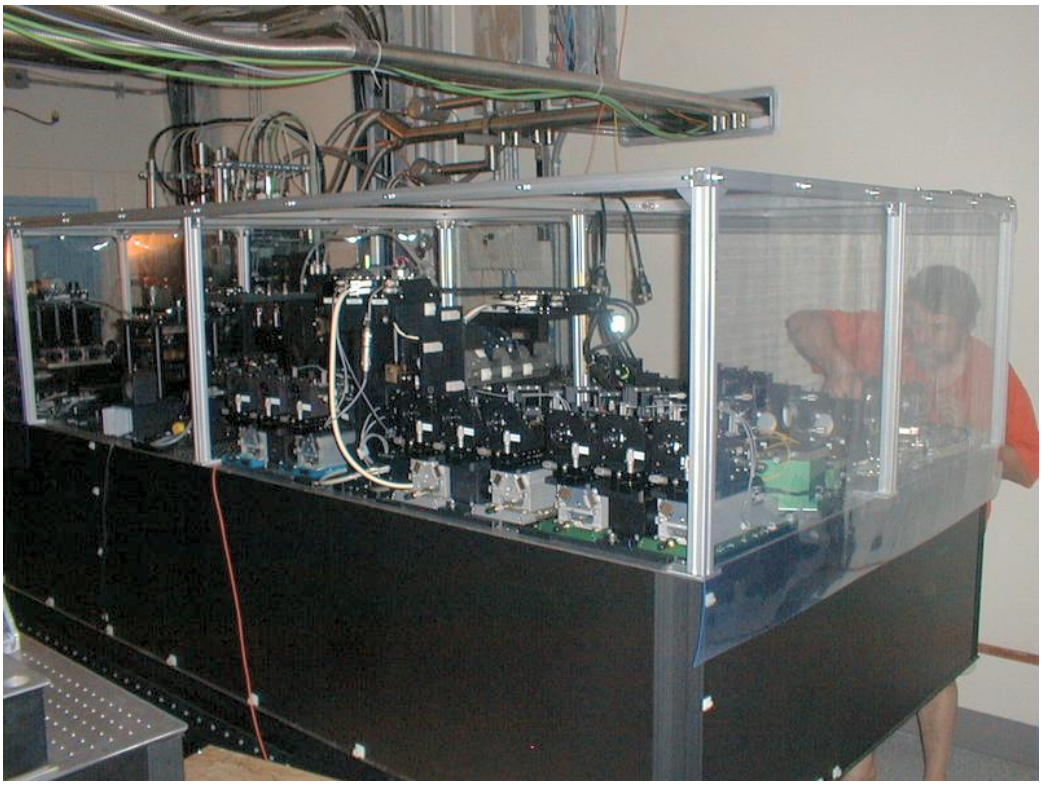

Figure 4.1.1: *The complex instrument AMBER consists of various optical and mechanical components to allow interferometric and spectroscopic observations at the same time.*

4.3 Angular Resolution

The angular resolution is set by the available baseline, which can reach about 200 meters for the ATs and about 130 meters for the UTs. Accordingly, the limit will be about 2 milliarcsecond (mas) for the ATs, and about 3 mas for the UTs, in the K band. These values are roughly halved for the J-band. Of course, the actual resolution will depend also on the signal-to-noise ratio (SNR) available, which is in turn a function of the source brightness, as well as of many factors mentioned below. Under conditions of very high SNR, it is in principle feasible to obtain higher resolutions than the broad limit given above. This applies in particular to cases in which a source model is available and determinations of the visibility before the first zero are sufficient to constraint the size of the object (for instance, angular diameters and binary stars). Reversely, conditions of low SNR (for instance, for very faint sources), will limit the actual maximum resolution to values which can be substantially worse than the figures mentioned above.

4.4 Flux Sensitivity

The limits in sensitivity depend on a large number of factors: observing modes that correspond to the elementary exposure time (high sensitivity: 50ms; high precision: 10ms; long exposures: any but with fringe tracking), the type of telescopes, the spectral resolution, the seeing, etc. Some of these are relatively well known and identified

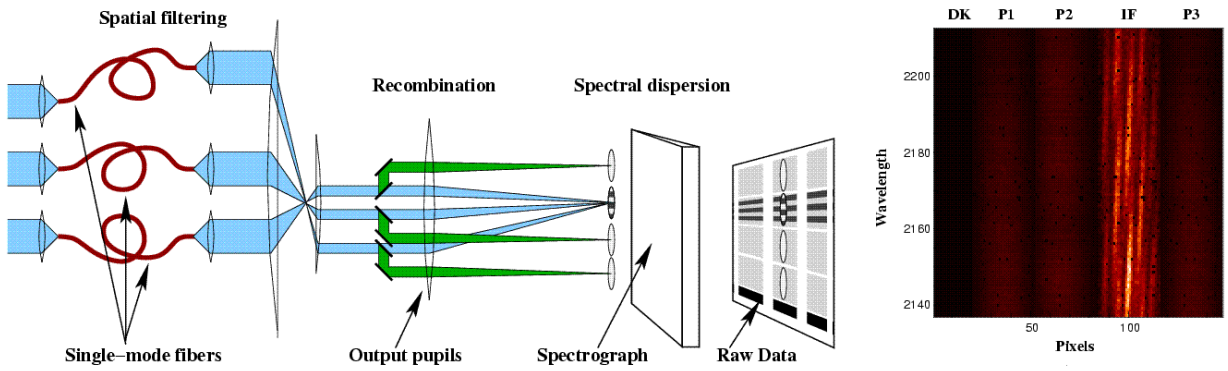

Figure 4.1.2: *Left panel: Sketch of the AMBER instrument. The light enters the instrument from left. Right panel: Reconstituted image from AMBER raw data in three-telescopes mode. DK corresponds to a dark region of the detector. Pk are vertically dispersed spectra from each telescope and IF is the spectrally dispersed interferogram.*

by the design of AMBER (e.g. transmission efficiency, fiber coupling, detector characteristics), while many others are more difficult to characterize at this time (e.g. the quality of fringe tracking and the adaptive optics correction).

4.5 Field Of View

AMBER is a single-mode instrument, therefore the field of view is theoretically limited to the Airy disk of each individual aperture, i.e. 250 mas for the ATs and 60 mas for the UTs in the K band.

5 Instrument Data Description

AMBER data can be separated into *raw* frames and *product* frames. Raw frames are the unprocessed output of the AMBER instrument observations, while product frames are the result of the AMBER pipeline processing (e.g. reduced frames, master calibration frames, p2vm, etc.).

Any raw or product frame can be classified on the basis of a set of header keywords. Data classification is typically carried out by the DO or by *Gasgano* [7], that applies the same set of classification rules. The association of a raw frame with calibration data (*e.g.*, of a science frame with a master flat field frame) can be obtained by matching the values of a different set of header keywords.

Each kind of raw frame is typically associated to a single AMBER pipeline recipe, *i.e.*, the recipe assigned to the reduction of that specific frame type. In the pipeline environment this recipe would be launched automatically.

A product frame may be input to more than one AMBER pipeline recipe, but it may be created by just one pipeline recipe (with the same exceptions mentioned before). In the automatic pipeline environment a product data frame alone wouldn't trigger the launch of any recipe.

In the following Section all raw and product AMBER data frames are listed, together with the keywords used for their classification and association. The indicated *DO category* is a label assigned to any data type after it has been classified, which is then used to identify the frames listed in the *Set of Frames* (see Section 6.3, page 28).

Please note that the actual rules used to classify the AMBER raw frames do contain more types of raw frames than listed below. This is for backwards compatibility with old AMBER data. Please refer to the AMBER.rul file (GASGANO) or the file amber.oca (online pipeline) for more details.

Raw frames can be distinguished in *calibration* frames and *science* frames. Their intended use is implicitly defined by the assigned recipe.

5.1 Calibration Frames

These frames are used for instrument calibration.

- **Pixel To Visibility Matrix and Spectral Calibration:** These frames are for calculation of a Pixel-To-Visibility-Matrix. Optionally the 2WAVE and 3WAVE files can be send to the recipe. In that case the the recipe will also perform a new spectral calibration.
  - DO category: `AMBER_2P2V`
    Processed by: `amber_p2vm`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = 2P2V`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

  - DO category: `AMBER_2WAVE`
    Processed by: `amber_p2vm`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = WAVE,2TEL`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

– DO category: `AMBER_3P2V`
  Processed by: `amber_p2vm`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = 3P2V`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

– DO category: `AMBER_3WAVE`
  Processed by: `amber_p2vm`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = WAVE,3TEL`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

- **Badpixel Map and Flat Field Map raw files:** These frames are for calculation of a Bad Pixel Map and the Flat Field Map with the AMBER recipe.

  – DO category: `AMBER_DETECTOR_DARK`
    Processed by: `amber_detector`

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

  – DO category: `AMBER_DETECTOR_FFM`
    Processed by: `amber_detector`

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

- **Beam Position Monitoring:** These frames are for calculation of the Quality Control parameters to monitor the beam position of AMBER in the various optical configurations.

– DO category: `AMBER_BEAMPOS`
  Processed by: `amber_BeamPos`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = STD`	`DET NTEL`	Number of Telescopes used
`DPR TECH = IMAGE`		

5.2 Scientific Frames

- **Scientific Observation:** These frames are for scientific observations. For the calculation of visibilities there is at least one science object frame needed. In addition there must be a valid P2VM as well as a Flat-Field and a Bad-Pixel-Map. For improved SNR it is useful to input the optional frames for sky and dark detector illumination.

  – DO category: `AMBER_SCIENCE`
    Processed by: `amber_SciCal`

Classification keywords:	Association keywords:	Note:
`DPR CATG = SCIENCE`		
`DPR TYPE = OBJECT`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

  – DO category: `AMBER_DARK_SCIENCE`
    Processed by: `amber_SciCal`

Classification keywords:	Association keywords:	Note:
`DPR CATG = SCIENCE`		
`DPR TYPE = DARK`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

  – DO category: `AMBER_SKY_SCIENCE`
    Processed by: `amber_SciCal`

Classification keywords:	Association keywords:	Note:
`DPR CATG = SCIENCE`		
`DPR TYPE = SKY`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

  – DO category: `AMBER_CALIB`
    Processed by: `amber_SciCal`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = OBJECT`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

– DO category: `AMBER_DARK_CALIB`
  Processed by: `amber_SciCal`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = DARK`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

– DO category: `AMBER_SKY_CALIB`
  Processed by: `amber_SciCal`

Classification keywords:	Association keywords:	Note:
`DPR CATG = CALIB`		
`DPR TYPE = SKY`	`DET NTEL`	Number of Telescopes used
`DPR TECH = INTERFEROMETRY`		

– DO category: `AMBER_P2VM`
  Processed by: `amber_SciCal`

Classification keywords:	Association keywords:	Note:
`PRO CATG = P2VM_REDUCED`		

6 Data Reduction

In this section the usage of the AMBER pipeline recipes is described. In particular, typical data reduction sessions for each instrument operating mode are presented in Sections 6.2 and 6.6.

6.1 Reduction Cascade

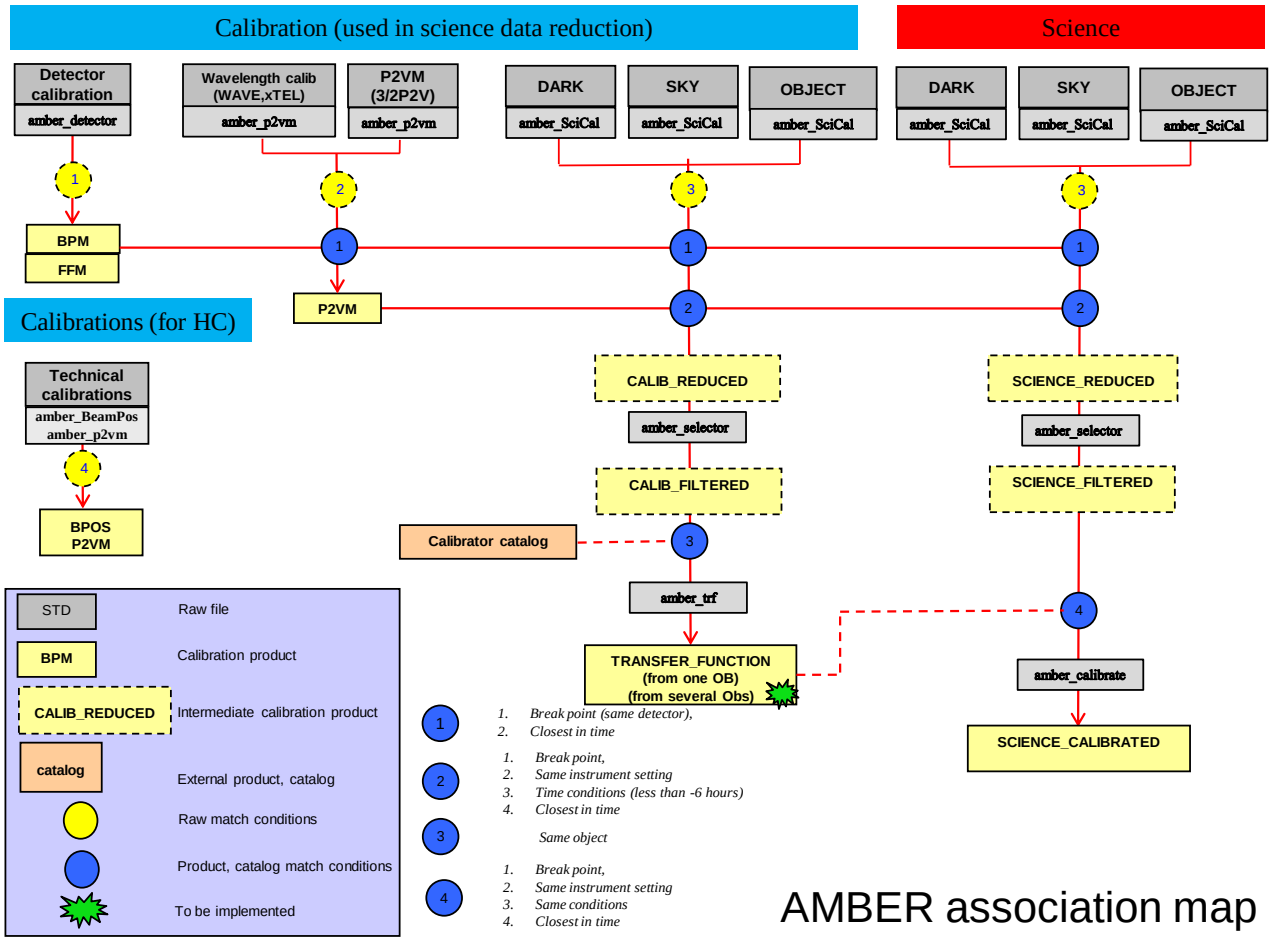

Figure 6.1.1: *AMBER Association Map*

6.2 AMBER pipeline recipes

The current AMBER pipeline is based on a set of stand-alone recipes, assigned to different fundamental operations. For the creation of general calibration data or informational output the following recipes can be used:

- **amber_raw_to_fitsimage:** As the AMBER raw frames are only containing number-tables, one can use this recipe to re-create a fits image out of this tables.
- **amber_detector:** This recipe creates the master flat field and the bad pixel map for the optical train and detector of the instrument.
- **amber_p2vm:** Calculation of the Pixel-To-Visibility-Matrix to be able to disentangle the interferometric channels for 2 and 3 telescopes. Optionally, if the specific raw frames are provided, it will also perform a spectral calibration. Moreover the recipe is also used to monitor the internal visibility of the instrument.
- **amber_SciCal:** This recipe calculates the visibilities along with additional information like the SNR, error bars and the closure phase for the given band.
- **amber_selector:** This recipe can select or filter the visibilities by various algorithm such as Fringe Contrast SNR
- **amber_BeamPos:** This recipe is calculating the QC parameters for alignment monitoring of the optical train
- **amber_ascii_export:** This recipe exports data stored in an AMBER OI product into ASCII files
- **amber_trf:** This recipe calculates the transfer function.
- **amber_calibrate:** This recipe calculates calibrated squared visibilities and closure phases.
- **amber_oimerge:** This recipe merges OI-Fits files

In the following sections a general description on the usage as well as a more detailed information on the individual recipes is given. Moreover, there will also be a description of how to use the recipes with EsoRex and GASGANO.

6.3 The set of frames (SOF)

Each pipeline recipe processes a set of input FITS data files. The filenames are listed together with their DO category in an ASCII file, the *Set of Frames* (SOF; required when launching a recipe). SOF files containing the frames selected by the user are automatically created by *Gasgano* [7].

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

```
FlatFieldMap.fits                    AMBER_FLATFIELD
BadPixelMap.fits                     AMBER_BADPIX
AMBER.2006-02-11T05:13:36.596.fits   AMBER_P2VM
AMBER.2006-02-11T05:53:03.559.fits   AMBER_SCIENCE
AMBER.2006-02-11T05:55:15.715.fits   AMBER_SKY
```

The pipeline recipe will access the listed files when required by the reduction algorithm.

Please note that the AMBER pipeline recipes do not check the classification tags specified in the SOF. However, the recipe will check for every frame if the optical train is the same as it has been during recording the P2VM files. It will also check if all the regions in the files are consistent, since otherwise the usage of a particular P2VM file will result in incorrect data.
The reason for this lack of control is that the AMBER recipes are just the DRS component of the complete pipeline running on Paranal, where the task of data classification and association is carried out by the Data Organizer. On the other hand, using *Gasgano* [7] as an interface to the pipeline recipes will permit to classify the data frames exactly as the DO does.

Please note that a recipe processing an incorrect SOF may stop with an error messages at best. In the worst cases, the recipe would run without any problem, producing results that may look reasonable while they are instead flawed.

6.4 Recipe configuration

At each pipeline recipe a configuration file, containing the default values of the parameters related to that recipe can be assigned. The possibility to create recipe configuration files, and the way information is conventionally stored in such files, depends on the front-end application used for launching the pipeline recipes[1]. In this section the recipe configuration files created and handled by *EsoRex* are described. Please refer to the Gasgano User's Manual [7] for information about saving recipe configuration parameters.

An *EsoRex* recipe configuration file can be generated in the directory `$HOME/.esorex` by *EsoRex* (see Section 6.5 on page 30) at installation time, and has the same name as the related recipe, with the filename extension `.rc`. For instance, the recipe *amber_SciCal* has its *EsoRex* generated configuration file named `amber_SciCal.rc`. Please note that without a recipe configuration file the recipe will use the default values.

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

```
# --Binning
# Binning of frames, default 1
amber.Parameters.binning=10
```

In this example, the parameter `amber.Parameters.binning` is set to the value `10`. In the configuration file generated by *EsoRex*, one or more comment lines are added containing information about the possible values of the parameter, and an alias that could be used as a command line option (see again Section 6.5 on page 30).

The hierarchy introduced in the parameter names has currently no implication. Although it is not used at the moment, it is envisaged that this feature will be used in future releases to avoid potential name clashes. The shorter parameter aliases are made available for use on the command line.

The parameter names belonging to the recipe specific configuration files are described in the corresponding recipe sections.

[1]The plugin concept allows the implementation of different front-end applications with specific functionality. For a complete description on how to create a pipeline recipe launcher, please refer to the CPL User Manual [6].

More than one configuration file may be maintained for the same recipe but, in order to be used, a configuration file not located under `$HOME/.esorex`, or having a name different from the recipe name, must be explicitly specified when launching a recipe (see next section).

6.5 Running a recipe using EsoRex

The AMBER pipeline recipes are implemented as plugins executed by a front-end applications like *EsoRex* or *Gasgano* [7]. The file organizer *Gasgano* [7] provides an intuitive GUI for launching the recipes (and applying the same rules used by the DO for the available data frames classification), whereas *EsoRex* allows to execute the pipeline recipes from a shell prompt.

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

`esorex` [*EsoRex_options*] *recipe_name* [*recipe_options*] *set_of_frames*

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

```
esorex --recipes
```

To get help for an individual recipe (in the subsequent examples, *amber_SciCal* is used), the following is used:

```
esorex --help amber_SciCal
```

In order to see the current parameters setting of a recipe, either the recipe configuration file (if available) located under `$HOME/.esorex` may be viewed, or the following command may be used:

```
esorex --params amber_SciCal
```

If the default recipe configuration file is not found or a particular value is not configured within this file, then the system defaults will be shown and used.

A recipe can be run by specifying its name to *EsoRex*, together with the name of a SOF (see Section 6.3, page 28). For instance, the following command line would be used to run the recipe *amber_SciCal* for processing the files specified in the SOF `amber_SciCal.sof`:

```
esorex amber_SciCal amber_SciCal.sof
```

A recipe configuration file different from the default one (see Section 6.4, page 29) can also be specified on the command line:

```
esorex --recipe-config=my_config.rc amber_SciCal amber_SciCal.sof
```

The recipe parameters can be modified either by editing directly the used configuration file, or by specifying new parameter values on the command line using the command line options. Such command line options should be inserted after the recipe name, and before the SOF name. For instance, to set the *amber_SciCal* recipe `binning` parameter to `10`, the following should be typed:

```
esorex amber_SciCal --binning=10 amber_SciCal.sof
```

There are also parameters used to configure the *EsoRex* launcher, that may be listed in an `esorex.rc` configuration file located under `$HOME/.esorex`. On the command line, the *EsoRex* options must be inserted before, and not after, the specified recipe name. The *EsoRex* options are those that are recipe independent, as for instance the verbosity level, the directory where the recipe products should be written, or the permission to overwrite old products with new ones.

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

6.6 Running a complete data reduction using GASGANO

For AMBER we need a two step data reduction: First a P2VM must be derived and with that P2VM we can then reduce the scientific data taken with the same optical train.

6.7 Creating a primary FITS-Image of the Channels

The recipe $amber_raw_to_fitsimage$ is able to display one AMBER raw frame at a time. Any rawframe listed in Table 6.7.1 can be used by the recipe. One can run the recipe in Gasgano by right clicking on a frame and choosing $amber_raw_to_fitsimage$

The recipe will store the FITS-image at the location $/tmp/amber_display.fits$. It can then be displayed by any suitable FITS viewer, i.e. the FV.

DO category	Type	Explanation	Required
AMBER_2P2V	Raw frame	2 telescopes P2VM	
AMBER_2WAVE	Raw frame	2 telescopes P2VM, spectral calibration	
AMBER_3P2V	Raw frame	3 telescopes P2VM	
AMBER_3WAVE	Raw frame	3 telescopes P2VM, spectral calibration	
AMBER_DARK_SCIENCE	Raw frame	dark frame from scientific observation	
AMBER_SKY_SCIENCE	Raw frame	sky frame from scientific observation	
AMBER_DARK_CALIB	Raw frame	dark frame from calib observation	
AMBER_SKY_CALIB	Raw frame	sky frame from calib observation	
AMBER_SCIENCE	Raw frame	science object frame from scientific observation	
AMBER_CALIB	Raw frame	calibrator object frame from scientific observation	
AMBER_BEAMPOS	Raw frame	beam position in various optical configurations	
AMBER_DETECTOR_DARK	Raw frame	dark frame for the bad pixel mask	
AMBER_DETECTOR_FFM	Raw frame	flatfields	

Table 6.7.1: *Input files for the amber_raw_to_fitsimage recipe.*

6.8 Calculating the Pixel-To-Visibility-Matrix

This recipe will calculate the Pixel-To-Visibility-Matrix for a given optical setup of AMBER, as the latter is needed as an input for all subsequent visibility calculations. Please note that a P2VM is only valid as long as the optical train remains the same. For this reason the P2VM raw files (see Table 6.8.1 for a list) have to be taken shortly before the scientific observation.

DO category	Type	Explanation	Required
AMBER_BADPIX	Calibration frame	Bad Pixel Map of the detector	√
AMBER_FLATFIELD	Calibration frame	Flat Field of the detector and the instrument's optical train	√
AMBER_2P2V	Raw frame	2 telescopes P2VM, 5 frames are needed	√
AMBER_2WAVE	Raw frame	2 telescopes P2VM, spectral calibration	
AMBER_3P2V	Raw frame	3 telescopes P2VM, 10 frames are needed	√
AMBER_3WAVE	Raw frame	3 telescopes P2VM, spectral calibration	

Table 6.8.1: *Input files for the amber_p2vm recipe.*

Please note that the input of spectral calibration files is optional. If there are no spectral calibration files, the product will contain the spectral calibration values as they were during data acquisition.

A screen-shoot of Gasgano launching the amber_p2vm recipe is displayed in Figure 6.8.1.

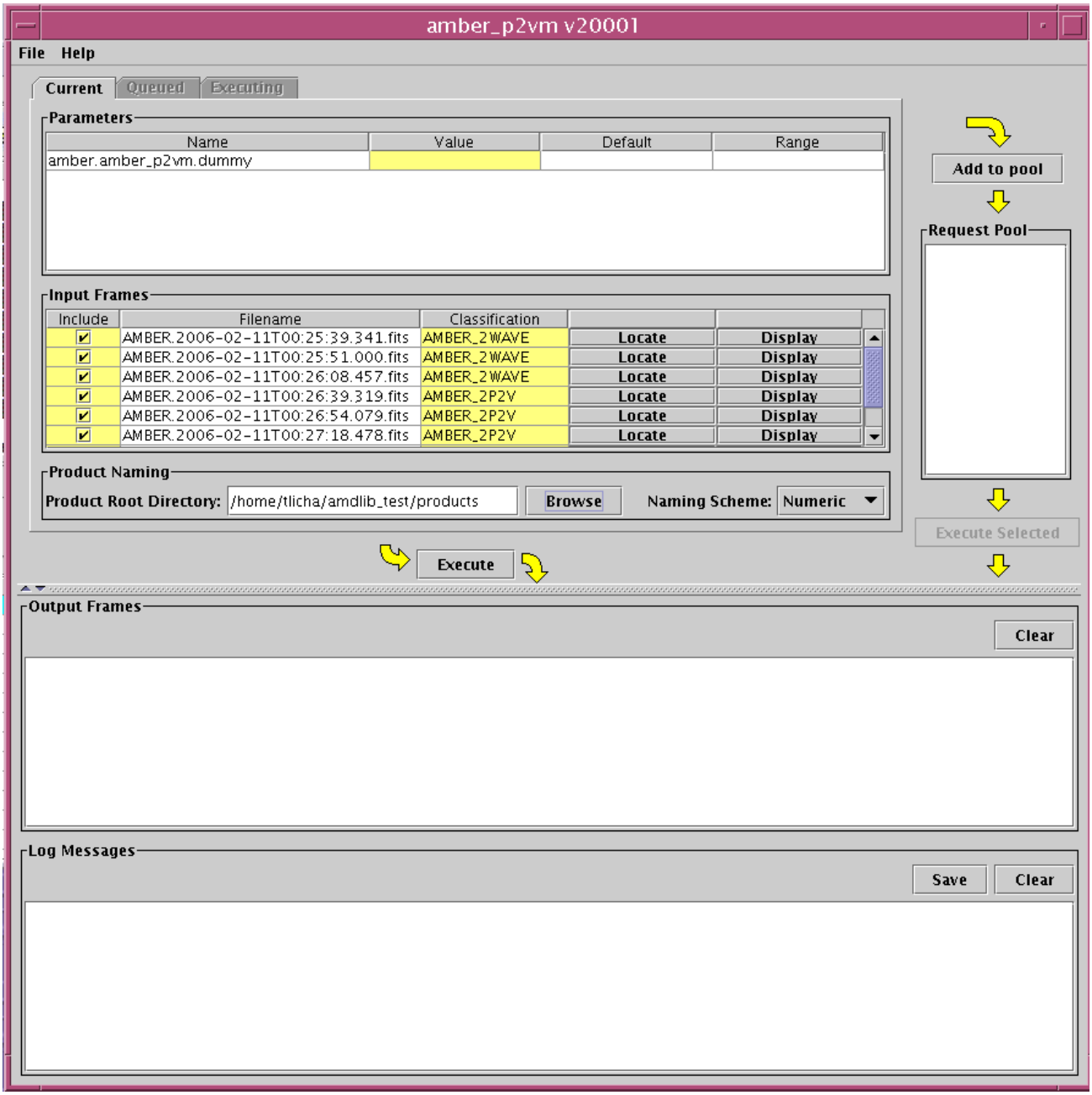

Figure 6.8.1: *The recipe amber_p2vm when launched with GASGANO.*

6.9 Calculating the scientific results

The AMBER pipeline recipe *amber_SciCal* is used to calculate visibilities from interferometric observations.

This recipe is able to handle 2- and 3-telescope observations. The DET NTEL keyword from the first rawfile's header will determine the mode. It can also handle all the different bands (JHK) in any resolution. If there is more than one band in the raw frame the recipe will split the bands and create a separate product file for the J, H, and K band, respectively.

All the files that must/can be included in the input SOF are listed in Table 6.9.1. In addition to the frames marked as required there must be at least one frame of the DO categories AMBER_SCIENCE, AMBER_CALIB or AMBER_SCIENCE_CALIB. For each of them the corresponding product will be created.

DO category	Type	Explanation	Required
AMBER_BADPIX	Calibration frame	Bad Pixel Map of the detector	√
AMBER_FLATFIELD	Calibration frame	Flat Field of the detector and the instrument's optical train	√
AMBER_P2VM	Calibration frame	Pixel-To-Visibility-Matrix of the instrument's optical train	√
AMBER_DARK_SCIENCE	Raw frame	dark frame from scientific observation	
AMBER_SKY_SCIENCE	Raw frame	sky frame from scientific observation	
AMBER_DARK_CALIB	Raw frame	dark frame from calib observation	
AMBER_SKY_CALIB	Raw frame	sky frame from calib observation	
AMBER_SCIENCE	Raw frame	science object frame from scientific observation	√
AMBER_CALIB	Raw frame	calibrator object frame from scientific observation	
AMBER_SCIENCE_CALIB	Raw frame	standard calibrator object frame	

Table 6.9.1: *Input files for the amber_SciCal recipe.*

A screen-shoot of Gasgano launching the *amber_SciCal* recipe is displayed in Figure 6.9.1.

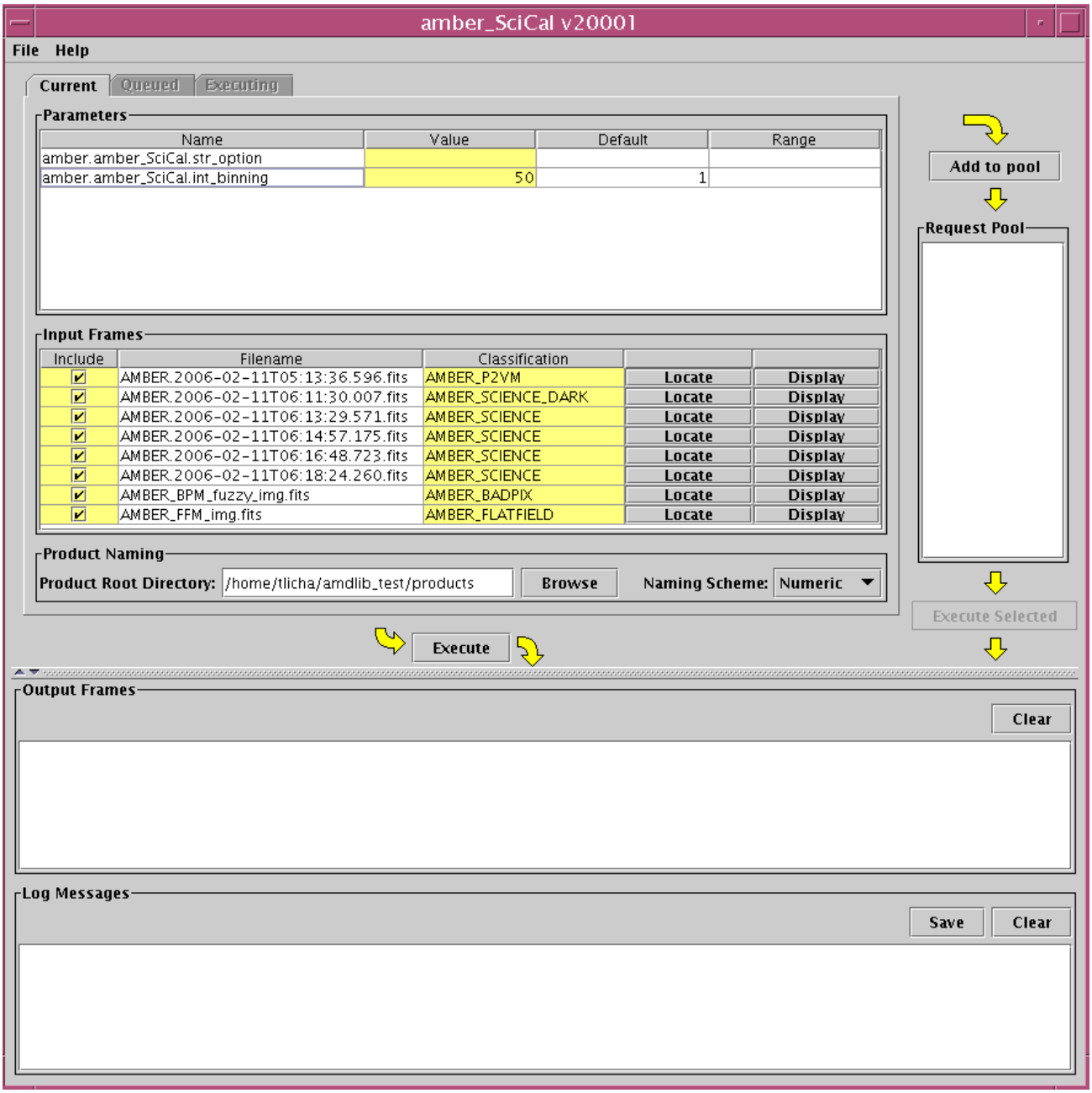

Figure 6.9.1: *The recipe amber_SciCal when launched with GASGANO.*

7 Pipeline Recipes Interfaces

7.1 amber-raw-to-fitsimage

The AMBER pipeline recipe *amber_raw_to_fitsimage* creates two products with the DO.CATG AMBER_RAW_CUBE and AMBER_RAW_SUM. The first transforms and saves the Amber rawfile into a cube structure whereas the second derives the sum of the cube and saves it as an image. The cube/image can be considered as a regeneration of the active CCD area. Any of the AMBER raw frames can be displayed with the recipe (it can handle raw files taken in 2 and 3 telescope mode). The recipe is very useful to check if the interferometric channel is actually showing fringes and if flux is present in all photometric channels. At least one of the files listed in Table 7.1.1 must be included in the SOF. If there are more files only the first one will be processed. As an example we show in Figure 7.1.1 the vertically dispersed channels of the AMBER instrument.

DO category	Type	Description	Required
AMBER_2P2V	Raw frame	2 telescopes P2VM	
AMBER_2WAVE	Raw frame	2 telescopes P2VM, spectral calibration	
AMBER_3P2V	Raw frame	3 telescopes P2VM	
AMBER_3WAVE	Raw frame	3 telescopes P2VM, spectral calibration	
AMBER_DARK_SCIENCE	Raw frame	dark frame from scientific observation	
AMBER_SKY_SCIENCE	Raw frame	sky frame from scientific observation	
AMBER_DARK_CALIB	Raw frame	dark frame from calib observation	
AMBER_SKY_CALIB	Raw frame	sky frame from calib observation	
AMBER_SCIENCE	Raw frame	science object frame from scientific observation	
AMBER_CALIB	Raw frame	calibrator object frame from scientific observation	
AMBER_BEAMPOS	Raw frame	beam position in various optical configurations	
AMBER_DETECTOR_DARK	Raw frame	dark frame for the bad pixel mask	
AMBER_DETECTOR_FFM	Raw frame	flatfields	

Table 7.1.1: *Input files for the amber_raw_to_fitsimage recipe.*

DO category	Type	Description
AMBER_RAW_CUBE	FITS-CUBE	reconstructed cube of the active detector regions
AMBER_RAW_SUM	FITS-IMAGE	summed image of the active detector regions

Table 7.1.2: *Output of the amber_raw_to_fitsimage recipe.*

The recipe creates two FITS files containing the primary image of the reconstructed image of the active detector regions. The header is a copy of the raw frame's header (see Table 7.1.2).

There are no *amber_raw_to_fitsimage* parameters.

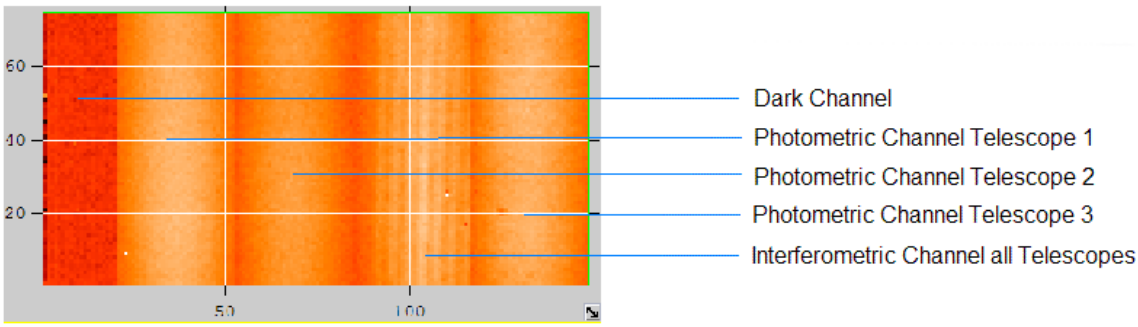

Figure 7.1.1: *Example image for the 3-telescopes mode.*

7.2 amber-detector

The AMBER pipeline recipe *amber_detector* is used to create the bad pixel map and the master flat field for the AMBER instrument. A detailed description of the used algorithms is given in Appendix B on page 58.

The files listed in Table 7.2.1 must be included in the SOF when running the recipe.

DO category	Type	Description	Required
AMBER_DETECTOR_DARK	Raw frame	dark exposure	√
AMBER_DETECTOR_FFM	Raw frame	flat exposure	√

Table 7.2.1: *Input files for the amber_detector recipe.*

The product written by the amber_detector recipe is listed in Table 7.2.2, whereas the QC parameters are listed in Table 7.2.3

File name	DO category	Type	Description
amber_eso_bpm.fits	AMBER_BADPIX	FITS	Bad pixel map
amber_eso_ffm.fits	AMBER_FLATFIELD	FITS	Master flat field

Table 7.2.2: *Product of the amber_detector recipe.*

QC Parameter	Description
BADPIX	the number of bad pixels on the chip
GOODPIX	the number of good pixels on the chip

Table 7.2.3: *QC values written by amber_detector recipe.*

7.3 amber-beampos

The AMBER pipeline recipe *amber_beampos* is used to monitor the beam alignment for the AMBER instrument. A technical template of the AMBER instrument will create the 22 mandatory files to run the recipe (see also Table 7.3.1). These files contain an image of the AMBER instrument beam under various illumination conditions. From these files the recipe will derive an image of the detector as well as QC parameters and write them to a product as well as to the QC log. The product written by the amber_beampos recipe is listed in Table 7.3.2

For each of the frames the QC values listed in Table 7.3.3 are written. Depending on the type of the frame there will be a prefix (see Table 7.3.4) written to identify the specific optical configuration, i.e. a complete QC parameter including the prefix would read:
HIERARCH.ESO.QC.CLDSTP.SIZX=23.45

DO category	Type	Description	Required
AMBER_BEAMPOS	Raw frame	exposures with different optic-settings	√

Table 7.3.1: *Input files for the amber_beampos recipe.*

File name	DO category	Type	Description
amber_beampos_XXX.fits	AMBER_BEAMPOS_REDUCED	FITS	FITS file with primary image of the beam and the QC parameters in the header (XXXX corresponds to the raw file's number)

Table 7.3.2: *Product of the amber_beampos recipe.*

QC Parameter	Description
X	the x position of the beam centroid
Y	the y position of the beam centroid
SIZX	the size in x direction of the beam
SIZY	the size in y direction of the beam
FLUX	the total flux of the beam
PEAKFLUX	the peak flux of the beam

Table 7.3.3: *QC values written by the amber_beampos recipe.*

Please note that this recipe currently relies on the correct order of the frames given in the Set of Frames. This will not be a problem for the online pipeline unless the acquisition template might be changed. In GASGANO the user should send the files in the order given by the entry of the header named TPL.EXPNO.

QC Prefix	Description
CLDSTP	cold stop
A.K	A in K band
B.K	B in K band
C.K	C in K band
A.H	A in H band
B.H	B in H band
C.H	C in H band
A.J	A in J band
B.J	B in J band
C.J	C in J band
BCD.A.K	BCD and A in K band
BCD.B.K	BCD and B in K band
BCD.C.K	BCD and C in K band
BCD.A.H	BCD and A in H band
BCD.B.H	BCD and B in H band
BCD.C.H	BCD and C in H band
BCD.A.J	BCD and A in J band
BCD.B.J	BCD and B in J band
BCD.C.J	BCD and C in J band
OUTF.A	OUTF and A
OUTF.B	OUTF and B
OUTF.C	OUTF and C

Table 7.3.4: *QC value prefixes written by the amber_beampos recipe.*

7.4 amber-p2vm

The AMBER pipeline recipe *amber_p2vm* is used to create a pixel to visibility matrix (P2VM). This P2VM is containing the translation matrix for the interferometric channel used to calculate visibilities from scientific observations. The recipe is able to handle 2- and 3-telescope P2VMs, where the DET NTEL keyword from the first rawfile's header will determine the mode. A more detailed description of the algorithms is given in Appendix B on page 58.

A single recipe parameter is provided by this recipe:

- **activate_NORMALIZE_P2VM:** If the amdlib parameter activate_NORMALIZE_P2VM is set to TRUE, the p2vm is normalized to unity. Please check the amdlib manual[2] (normalizeP2vm) for details.

The raw files are containing all possible variations of the shutter positions of the photometric channels and the response of the interferometric channel for the standard lamps used during that process. Because in 3-telescope mode there is one more photometric channel compared to the 2 telescope mode, more input files are needed (10 instead of 5). The files listed in Table 7.4.1 can/must be used in the SOF when running the recipe.

DO category	Type	Description	Required
AMBER_BADPIX	Calibration frame	Bad Pixel Map of the detector	√
AMBER_FLATFIELD	Calibration frame	Flat Field of the detector and the instrument's optical train	√
AMBER_2P2V	Raw frame	2 telescopes P2VM, 5 frames are needed	√
AMBER_2WAVE	Raw frame	2 telescopes P2VM, spectral calibration	
AMBER_3P2V	Raw frame	3 telescopes P2VM, 10 frames are needed	√
AMBER_3WAVE	Raw frame	3 telescopes P2VM, spectral calibration	

Table 7.4.1: *Input files for the amber_p2vm recipe.*

The only product of the *amber_p2vm* recipe is the P2VM file (see Table 7.4.2).

File name	DO category	Type	Description
amber_p2vm.fits	P2VM_REDUCED	FITS	Pixel to Visibility Matrix

Table 7.4.2: *Product of the amber_p2vm recipe.*

This recipe is writing the QC parameters listed in Table 7.4.3.

[2] Available at http://www.jmmc.fr/amberdrs

QC Parameter	Description
P1 OFFSETY	y offset after spectral calibration in pixels for photometric channel 1
P2 OFFSETY	y offset after spectral calibration in pixels for photometric channel 2
P3 OFFSETY	y offset after spectral calibration in pixels for photometric channel 3
P2VM VISXX	Imternal Visibility for baseline XX
P2VM ERRVISXX	Imternal Visibility Error for baseline XX
P2VM J VISXX	Imternal J-Band Visibility for baseline XX
P2VM J ERRVISXX	Imternal J-Band Visibility Error for baseline XX
P2VM H VISXX	Imternal H-Band Visibility for baseline XX
P2VM H ERRVISXX	Imternal H-Band Visibility Error for baseline XX
P2VM K VISXX	Imternal K-Band Visibility for baseline XX
P2VM K ERRVISXX	Imternal K-Band Visibility Error for baseline XX
ZEROFRAMES	Number of Zeroframes in the raw files

Table 7.4.3: *QC values written by the amber_p2vm recipe.*

7.5 amber-SciCal

The AMBER pipeline recipe *amber_SciCal* is used to calculate raw (uncalibrated) visibilities, transfer functions as well as calibrated visibilities from interferometric observations. This recipe is able to handle 2- as well as 3-telescope observations. The DET NTEL keyword from the first rawfile's header will determine the mode. It can handle all the different bands in any resolution. If there is more than one band in the raw frame the recipe will split the bands and create a separate product for the J, H and K band.

If the pipeline recipe parameter `selectPlusTrf` is set to FALSE, the recipe only calculates uncalibrated visibilities. On the other hand, if set to TRUE, the recipe performs the following additional tasks:

- **CALIB**: If a calibrator is observed, first it derives the uncalibrated visibilities. Then it performs a frame selection based on the Signal-to-noise ratio and another on the piston value. The SNR based selection criteria (see the recipe amber_selector in section 7.6 for the parameter definitions) select the 50 per cent best frames, i.e.:

  ```
  selection-method="Fringe_SNR_percentage_x"
  X1=50
  X2=50
  X3=50
  ANDselection=TRUE
  ```

  The selection values based on the piston are:

  ```
  selection-method="Absolute_piston_value_lt_x"
  X1=
  X2=
  X3=
  ANDselection=TRUE
  ```

  The values of `X1` `X2` and `X3` differ according to the observation mode and are listed in table 7.5.1

  The products of these selection are OI-Fits files with the prefix `snr_filtered` and `pst_filtered` and the DO category `CALIB_REDUCED_FILTERED`
  For convenience, the different filtering criteria are stored in the product header as quality control parameters as follows:

  ```
  USEDVALUE METHOD
  USEDVALUE AND
  USEDVALUE X1
  USEDVALUE X2
  USEDVALUE X3
  ```

  Finally, if the calibrator database (DO category `CALIB_DATABASE_J` or `CALIB_DATABASE_H` or `CALIB_DATABASE_K`) is in the SOF and the observed calibrator can be found in the calibrator database, the recipe calculates the expected visibilities for each wavelength and derives and saves for each selected product a transfer function file (prefix `trf_`) with the following DO category: `AMBER_TRF_J` or

AMBER_TRF_H or AMBER_TRF_K, respectively. The transfer function can then be included in the SOF of a science observation in order to calculate calibrated visibilities.

Please note that the transfer function is derived for each frame in the OI-FITS file. Only when the transfer function is applied to the science observation a mean transfer function is derived for the squared visibilities and the closure phase. This averaged transfer function is then applied to calculate calibrated squared visibilities and a calibrated closure phase.

FINITO	**low resolution** X1, X2, X3	**medium resolution** X1, X2, X3	**high resolution** X1, X2, X3
on	20 μm	200 μm	200 μm
off	40 μm	400 μm	400 μm

Table 7.5.1: *Frame selection criteria of the amber_SciCal recipe: selection is done on the piston value*

- **SCIENCE**: If a science target is observed, first it derives the uncalibrated visibilities. Then it performs a frame selection based on the Signal-to-noise ratio and another on the piston value. The SNR based selection criteria (see the recipe amber_selector in section 7.6 for the parameter definitions) select the 50 per cent best frames, i.e.:

```
selection-method="Fringe_SNR_percentage_x"
X1=50
X2=50
X3=50
ANDselection=TRUE
```

The selection values based on the piston are:

```
selection-method="Absolute_piston_value_lt_x"
X1=
X2=
X3=
ANDselection=TRUE
```

The values of X1 X2 and X3 differ according to the observation mode and are listed in table 7.5.1

The products of these selection are OI-Fits files with the prefix snr_filtered and pst_filtered and the DO category SCIENCE_REDUCED_FILTERED
For convenience, the different filtering criteria are stored in the product header as quality control parameters as follows:

```
USEDVALUE METHOD
USEDVALUE AND
USEDVALUE X1
USEDVALUE X2
USEDVALUE X3
```

Finally, if the transfer function (DO category `AMBER_TRF_J` or `AMBER_TRF_H` or `AMBER_TRF_K`) is in the SOF the recipe derives and saves for each filtered product the calibrated visibilities (prefix `cal_`) with the following DO category: `SCIENCE_CALIBRATED`.

Please note that the transfer function has been previously derived for each frame in the OI-FITS file. Only when the transfer function is applied to the science observation a mean transfer function is derived for the squared visibilities and the closure phase. This averaged transfer function is then applied to calculate calibrated squared visibilities and a calibrated closure phase. Moreover, the errors of the transfer function is propagated and also taken into account when calibrating the visibilities.

All the files that the recipe can handle in the input SOF are listed in Table 7.5.2. In addition to the frames marked as required there must be at least one frame of the DO categories AMBER_SCIENCE or AMBER_CALIB (and AMBER_SCIENCE_CALIB for backwards compatibility). For each of them a corresponding product will be created.

DO category	Type	Description	Required
AMBER_BADPIX	Calibration frame	Bad Pixel Map of the detector	√
AMBER_FLATFIELD	Calibration frame	Flat Field of the detector and the instrument's optical train	√
AMBER_P2VM	Calibration frame	Pixel-To-Visibility-Matrix of the instrument's optical train	√
AMBER_TRF_(J | H | K)	Calibration frame	transfer function	
CALIB_DATABASE_(J | H | K)	Calibration frame	calibrator database	
AMBER_DARK_SCIENCE	Raw frame	dark frame from scientific observation	
AMBER_SKY_SCIENCE	Raw frame	sky frame from scientific observation	
AMBER_DARK_CALIB	Raw frame	dark frame from calib observation	
AMBER_SKY_CALIB	Raw frame	sky frame from calib observation	
AMBER_SCIENCE	Raw frame	science object frame	√
AMBER_CALIB	Raw frame	calibrator object frame	

Table 7.5.2: *Input files for the amber_SciCal recipe.*

The products of the *amber_SciCal* recipe is an OI-Fits file (see Table 7.5.3).

File name	DO category	Type	Description
amber_XXX.fits	SCIENCE_REDUCED	OI-FITS	uncalibrated visibilities
amber_XXX.fits	CALIB_REDUCED	OI-FITS	uncalibrated visibilities
pst_filtered_XXX.fits	SCIENCE_REDUCED_FILTERED	OI-FITS	piston filtered OI FITS file
pst_filtered_XXX.fits	CALIB_REDUCED_FILTERED	OI-FITS	piston filtered OI FITS file
snr_filtered_XXX.fits	SCIENCE_REDUCED_FILTERED	OI-FITS	SNR filtered OI FITS file
snr_filtered_XXX.fits	CALIB_REDUCED_FILTERED	OI-FITS	SNR filtered OI FITS file
trf_XXX.fits	AMBER_TRF_(J | H | K)	OI-FITS	Transfer function
cal_XXX.fits	SCIENCE_CALIBRATED	OI-FITS	Calibrated visibilities

Table 7.5.3: *Products of the amber_SciCal recipe.*

The following recipe parameters are provided by `amber_SciCal`

- **binning:** The number of frames to be averaged for each visibility calculations.

- **selectPlusTrf:** If activated (set to TRUE) the recipe selects good visibilities following the rules described above and tries to calculate the transfer function.
- **activate_BEAUTIFY_PISTON:** If activated (set to TRUE) the piston calculation is improved by using piston closure. Please check the amdlib manual for more details.
- **value_BEAUTIFY_PISTON:** Not yet used
- **activate_MIN_PHOTOMETRY:** If activated (set to TRUE) all frames with photometric values smaller than the value defined in value_MIN_PHOTOMETRY (see below) will be ignored when calculating the total photometry of the object. Please check the amdlib manual (minPhotometry) for more details.
- **value_MIN_PHOTOMETRY:**: Threshold for the recipe parameter activate_MIN_PHOTOMETRY. Please check the amdlib manual (minPhotometry) for more details.
- **activate_DROP:** If activated (set to TRUE) the pipeline will ignore the first <value_DROP> frames in the raw data being calibrated. Please check the amdlib manual (dropFrames) for more details.
- **value_DROP:** Threshold for the recipe parameter activate_DROP. Please check the amdlib manual (dropFrames) for more details.

This recipe is writing the QC parameters listed in Table D.1.0. Note that the average of the squared visibility (V2) and closure phase (CP) is derived including the values at the wavelength boundaries. Moreover, the three bins (BIN1/2/3) are centered at $\frac{1}{6}$, $\frac{3}{6}$ and $\frac{5}{6}$ times the analyzed number of channels, e.g. if 90 channels are present, BIN1/2/3 is centered at channel 15, 45, 75. The quality parameters for CEN are centered in the middle of the analyzed channels (i.e.. at channel 45 for 90 channels). The width is 25 per cent of the analyzed number of channels for BIN1/2/3 and 80 percent for the CEN quality control parameter, i.e. for CEN and 90 channels the averaging is done by using 72 channels (i.e. channel 45 $\pm$ 36).
Please note that all channels and frames are supposed to be valid during the averaging process of the QC calculation and that the QC parameters of baseline 2 and 3 are only derived if amber is operated in three telescope mode.

7.6 amber-selector

The AMBER pipeline recipe *amber_selector* is used to select matching frames from AMBER products created by *amber_SciCal* (see Table 7.5.3 and Table 7.6.1). The output is again an AMBER product (same OI FITS format like the input) but will only contain the matching frames which passed the filter. This means the resulting product can be plotted by the same tools like all other AMBER OI FITS files.

Please note that in three telescopes mode we will of course have three baselines. In the OI fits standard they have to be written in a sequence of triples like {base1, base2, base3}, {base1, base2, base3}, ... To maintain consistency the whole triple has to be written even if only *ONE* visibility matches a certain selection. Therefore the written product can still contain visibilities which actually did not match the selection!!

DO category	Type	Description	Required
SCIENCE_REDUCED	product frame	OI FITS file	√
CALIB_REDUCED	product frame	OI FITS file	√

Table 7.6.1: *Input files for the amber_selector recipe.*

There are various algorithms to select and filter the data. The algorithms can be selected and controlled by the parameters *selection_method*, *selection_x1*, *selection_x2*, *selection_x3* and *selection_ANDselection*. The corresponding aliases for esorex execution are:

```
--selection-method=
--X1=
--X2=
--X3=
--ANDselection=
```

e.g.:

```
esorex  --selection-method="Fringe_SNR_gt_x" --X1=2.3
--X2=4 --X3=5.4 --ANDselection=FALSE SetOfFrame.txt
```

After selecting a method (see below) one can impose a selection threshold for every baseline by defining the variables X1, X2, and X3. Moreover one can also choose the way how the single thresholds should be applied to a frame by setting the Boolean variable ANDselection:

- **ANDselection=FALSE**: If ANDselection is set to FALSE, a frame passes the filter if at least one of the selection criteria are fulfilled (X1 or X2 or X3) for that particular frame.
- **ANDselection=TRUE**: If ANDselection is set to TRUE, a frame passes the filter only if ALL selection criteria (X1 and X2 and X3) are fulfilled for that particular frame.

If only one baseline is available, X2, X3 as well as ANDselection can be neglected.

The different selection methods are:

- **First_x_Frames:** This method will filter the first x frames (given by setting the variable X1) of the observation. The purpose of this method is a more technical one. It can be used to check if the instrument is properly suppressing the very first few frames after switching the detector to ON. These frames are often of low quality due to the design of the readout electronics of the detector. The variables X2, X3 as well as ANDselection can be omitted.
  Units: [X1, X2, X3] == [NONE, NONE, NONE]
  Boundaries for X1, X2, and X3: [1, NumberOfFrames]

- **Fringe_SNR_gt_x:** This method is based on the fringe Signal to Noise Ratio.
  For ANDselection=FALSE: If for a frame the fringe SNR of one baseline is greater than the given threshold, the corresponding frame will be selected.
  For ANDselection=TRUE: If for a frame the fringe SNRs of all baselines are greater than the given threshold, the corresponding frame will be selected.
  Units: [X1, X2, X3] == [NONE, NONE, NONE]
  Boundaries for X1, X2, and X3: [0, FLT_MAX[3]]

- **Fringe_SNR_percentage_x:** This method is based on the fringe Signal to Noise Ratio.
  For ANDselection=FALSE: If for a frame the fringe SNR of one baseline is within the given percentile threshold (of that specific baseline), the corresponding frame will be selected.
  For ANDselection=TRUE: If for a frame the fringe SNRs of all baselines are within the given percentile threshold (of that specific baseline), the corresponding frame will be selected.
  If for example X1=15, X2=20, X3=25 and ANDselection=TRUE, the method selects only frames with SNR among the best 15 percent for baseline 1, *and* among the best 20 percent for baseline 2, *and* among the best 25 percent for baseline 3. If for example X1=15, X2=20, X3=25 and ANDselection=FALSE, the method selects frames with SNR among the best 15 percent for baseline 1, *or* among the best 20 percent for baseline 2, *or* among the best 25 percent for baseline 3.
  Units: [X1, X2, X3] == [Percentage, Percentage, Percentage]
  Boundaries for X1, X2, and X3: [0, 100]

- **Flux_gt_x:** This method is based on a flux criteria: At the moment the flux criteria is defined by: $sqrt(Pi * Pj)/sqrt(fluxPi + fluxPj + npix * RON^2)$ and summed over full spectrum
  Please note that this method is not yet validated!

- **Flux_percentage_x:** This method is based on a flux criteria: At the moment the flux criteria is defined by: $sqrt(Pi * Pj)/sqrt(fluxPi + fluxPj + npix * RON^2)$ and summed over full spectrum
  Please note that this method is not yet validated!

- **Exclude_Frames_by_ASCII_File:** This method is an interface to external tools able to identify frames to be excluded. The files to exclude must be written by frame-index to the ASCII file */tmp/amber_exclude.txt*. Only one frame-index per line is allowed. For example, if frame number 13, 20, 25 and 255 should be excluded, the file should look as shown in Table 7.6.2. The output of the recipe is a OI FITS file which contains all frames but not the four listed before.

- **Include_Frames_by_ASCII_File:** This method is an interface to external tools able to identify frames to be included. The files to include must be written by frame-index to the ASCII file */tmp/amber_include.txt*. Only one frame-index per line is allowed. See also selection method -Exclude Frames by ASCII File-.

[3] FLT_MAX is system specific; e.g. 3e+38

13
20
25
255

Table 7.6.2: *Example of the file /tmp/amber_exclude.txt.*

- **IO-Test_no_filtering:** This method is mainly for testing the integrity of the code. It will read the whole input to internal structures and write it back unchanged. Only some header keywords may be added/erased/updated. The values for X1, X2, X3, and ANDselected can be omitted.

- **Absolute_piston_value_lt_x:** This method is based on the piston value.
  For ANDselection=FALSE: If for a frame the absolute piston value of one baseline is smaller than the given threshold, the corresponding frame will be selected.
  For ANDselection=TRUE: If for a frame the absolute piston value of all baselines are smaller than the given threshold, the corresponding frame will be selected.
  Units: [X1, X2, X3] == [Meter, Meter, Meter]
  Boundaries for X1, X2, and X3: [0, FLT_MAX[3]]

- **Absolute_piston_value_percentage_x:** This method is based on the piston value.
  For ANDselection=FALSE: If for a frame the absolute piston value of one baseline is within the given percentile threshold (of that specific baseline), the corresponding frame will be selected.
  For ANDselection=TRUE: If for a frame the absolute piston value of all baselines are within the given percentile threshold (of that specific baseline), the corresponding frame will be selected.
  If for example X1=15, X2=20, X3=25 and ANDselection=TRUE, the method selects only frames with an absolute piston value among the best 15 percent for baseline 1, *and* among the best 20 percent for baseline 2, *and* among the best 25 percent for baseline 3. If for example X1=15, X2=20, X3=25 and ANDselection=FALSE, the method selects frames with an absolute piston value among the best 15 percent for baseline 1, *or* among the best 20 percent for baseline 2, *or* among the best 25 percent for baseline 3.
  Units: [X1, X2, X3] == [Percentage, Percentage, Percentage]
  Boundaries for X1, X2, and X3: [0, 100]

This recipe is writing the QC parameters listed in Table D.2.0. The QC parameter describing the applied threshold are especially useful for a selection by percentage, as they contain the absolute threshold values.
Moreover, the average of the squared visibility (V2) and closure phase (CP) is derived including the values at the wavelength boundaries. The three bins (BIN1/2/3) are centered at $\frac{1}{6}$, $\frac{3}{6}$ and $\frac{5}{6}$ times the analyzed number of channels, e.g. if 90 channels are present, BIN1/2/3 is centered at channel 15, 45, 75. The quality parameters for CEN are centered in the middle of the analyzed channels (i.e.. at channel 45 for 90 channels). The width is 25 per cent of the analyzed number of channels for BIN1/2/3 and 80 percent for the CEN quality control parameter, i.e. for CEN and 90 channels the averaging is done by using 72 channels (i.e. channel 45 $\pm$ 36).

Please note that only the selected frames are used to derive the QC parameters and all selected channels and frames are supposed to be valid during the averaging process. Obviously, the QC parameters of baseline 2 and 3 are only derived if amber is operated in three telescope mode.

Moreover, one can control the recipe product also with the `AverageFrames` parameter.

- **AverageFrames=FALSE**: If AverageFrames is set to FALSE, the products will include all single frames which passed the filtering.
- **AverageFrames=TRUE**: If AverageFrames is set to TRUE, the frames will be averaged after the filtering and the product will contain only the averaged values.

7.7 amber_trf

The AMBER pipeline recipe *amber_trf* is used to derive the transfer function from AMBER products created by *amber_SciCal* or *amber_selector*. In order to calculate the transfer function, the product must be from a calibrator object and must be found in the calibrator database given in the SOF.

All the files that must be included in the input SOF are listed in Table 7.7.1.

DO category	Type	Description	Required
CALIB_REDUCED	product frame	OI FITS file	√
CALIB_REDUCED_FILTERED	product frame	OI FITS file	
CALIB_DATABASE_J	calib frame	J-Band Calibrator database	√
CALIB_DATABASE_H	calib frame	H-Band Calibrator database	
CALIB_DATABASE_K	calib frame	K-Band Calibrator database	

Table 7.7.1: *Input files for the amber_trf recipe.*

The product of the *amber_trf* recipe is one or several OI-FITS file(s) depending on the number of files in the SOF (see also Table 7.7.2). The name of the product consists of the name of the input filename where the prefix `trf_` is appended.

File name	DO category	Type	Description
trf_XXX.fits	AMBER_TRF_(J | H | K)	FITS	OI FITS file, with xxxx corresponding to the input filename

Table 7.7.2: *Product of the amber_trf recipe.*

This recipe is writing the QC parameters listed in Table D.3.0.

7.8 amber_calibrate

The AMBER pipeline recipe *amber_calibrate* is used to calibrate the squared visibilities as well as the closure phase (if present). In order to derive calibrated squared visibilities and closure phase, the SOF must contain a science object as well as a suitable transfer function. All the files that must be included in the input SOF are listed in Table 7.8.1.

DO category	Type	Description	Required
SCIENCE_REDUCED	product frame	OI FITS file	√
SCIENCE_REDUCED_FILTERED	product frame	OI FITS file	
AMBER_TRF_J	calib frame	J-Band transfer function	√
AMBER_TRF_H	calib frame	H-Band transfer function	
AMBER_TRF_K	calib frame	K-Band transfer function	

Table 7.8.1: *Input files for the amber_calibrate recipe.*

The product of the *amber_calibrate* recipe is one or several OI-FITS file(s) depending on the number of files in the SOF (see also Table 7.8.2). The name of the product consists of the name of the input filename where the prefix `cal_` is appended.

File name	DO category	Type	Description
cal_XXX.fits	SCIENCE_CALIBRATED	FITS	OI FITS file, with xxxx corresponding to the input filename

Table 7.8.2: *Product of the amber_calibrate recipe.*

This recipe is writing the QC parameters listed in Table D.4.0.

7.9 amber_oimerge

The AMBER pipeline recipe *amber_oimerge* is used to merge OI-fits files.

First it tries to group the files in the SOF according to the waveband coverage (J, H, or K) by using the `QC BAND` header parameter. If not present, the recipe assumes that all the files cover the same wavelength.

In a second step the recipe checks, if all the OI-fits files in a group are compatible:

- Extension `OI_WAVELENGTH`: the `EFF_BAND` as well as the `EFF_WAVE` columns must be identical.
- Extension `OI_ARRAY`: the `TEL_NAME` the `STA_NAME` as well as the `STA_INDEX` columns must be identical.

If these checks are successful for a group, the recipe merges the OI-fits files of that group.

All the files that can be included in the input SOF are listed in Table 7.9.1.

DO category	Type	Description	Required
SCIENCE_REDUCED	product frame	OI FITS file	
SCIENCE_REDUCED_FILTERED	product frame	OI FITS file	
CALIB_REDUCED	product frame	OI FITS file	
CALIB_REDUCED_FILTERED	product frame	OI FITS file	
AMBER_TRF_J	product frame	OI FITS file	
AMBER_TRF_H	product frame	OI FITS file	
AMBER_TRF_K	product frame	OI FITS file	
SCIENCE_CALIBRATED	product frame	OI FITS file	

Table 7.9.1: *Input files for the amber_oimerge recipe.*

The product of the *amber_oimerge* recipe is one or several OI-FITS file(s) depending on the number of groups in the SOF. The **DO category** of the merged frame(s) is the same as the first valid **DO category** of the group (see table 7.9.2):

If present in the first valid frame the recipe inherits the QC parameters listed in Table 7.9.3.

7.10 amber-ascii-export

The AMBER pipeline recipe *amber_ascii_export* is part of the distribution but not supported by ESO. It was kindly provided by Klara Shabun. Please contact her directly for any questions on the recipe.

File name	DO category	Type	Description
merged_(J|H|K)-band.fits	SCIENCE_REDUCED	OI-fits	merged OI-fits file
merged_(J|H|K)-band.fits	SCIENCE_REDUCED_FILTERED	OI-fits	merged OI-fits file
merged_(J|H|K)-band.fits	CALIB_REDUCED	OI-fits	merged OI-fits file
merged_(J|H|K)-band.fits	CALIB_REDUCED_FILTERED	OI-fits	merged OI-fits file
merged_(J|H|K)-band.fits	AMBER_TRF_J	OI-fits	merged OI-fits file
merged_(J|H|K)-band.fits	AMBER_TRF_H	OI-fits	merged OI-fits file
merged_(J|H|K)-band.fits	AMBER_TRF_K	OI-fits	merged OI-fits file
merged_(J|H|K)-band.fits	SCIENCE_CALIBRATED	OI-fits	merged OI-fits file

Table 7.9.2: *Product of the amber_oimerge recipe. Please note, that the filename chances to merged_unknown.fits if the waveband can not be identified properly, i.e. if the QC BAND header parameter is missing.*

QC Parameter	Description
LAMBDA CHAN	total number of channels in the file
LAMBDA MIN	lowest wavelength in the file [micron]
LAMBDA MAX	highest wavelength in the file [micron]
BAND	waveband
STA1	index of station 1
STA2	index of station 2
STA3	index of station 3

Table 7.9.3: *QC values inherited by the amber_oimerge recipe.*

7.11 amber_spectral_calibration

The AMBER pipeline recipe *amber_spectral_calibration* is used to calculates the real Y offset between the different photometric and the interferometric channels in the 3-telescope mode. It uses the same algorithm as the *amber_p2vm* recipe.

No recipe parameter is provided by this recipe.

The files listed in Table 7.11.1 must be used in the SOF when running the recipe.

DO category	Type	Description	Required
AMBER_BADPIX	Calibration frame	Bad Pixel Map of the detector	√
AMBER_FLATFIELD	Calibration frame	Flat Field of the detector and the instrument's optical train	√
AMBER_3WAVE	Raw frame	4 3-telescopes spectral calibration files	

Table 7.11.1: *Input files for the amber_spectral_calibration recipe.*

The only product of the *amber_spectral_calibration* recipe is the file listed in table 7.11.2.

File name	DO category	Type	Description
amber_spectral_calibration.fits	AMBER_SPECTRAL_CALIBRATION	FITS	Spectral shifts in the header

Table 7.11.2: *Product of the amber_spectral_calibration recipe.*

This recipe is writing the QC parameters listed in Table 7.11.3.

QC Parameter	Description
P1 OFFSETY	y offset after spectral calibration in pixels for photometric channel 1
P2 OFFSETY	y offset after spectral calibration in pixels for photometric channel 2
P3 OFFSETY	y offset after spectral calibration in pixels for photometric channel 3

Table 7.11.3: *QC values written by the amber_spectral_calibration recipe.*

A Installation

See: Installation instructions

B Algorithms

In this section the data reduction procedures applied by the pipeline recipes currently in use (see Section 6.2) are described in more detail. Common algorithms, as cosmic rays removal or bad pixel cleaning, are described separately.

B.1 General Algorithms

B.1.1 Dark subtraction

Dark frames are taken during the actual observation run. If no dark is available, the dark channel (left most side on the CCD chip) will be used, instead. This will of course reduce the accuracy.

B.1.2 Bad Pixel Map and Flat Field Generation

The following paragraph was adopted from an article (private communication) by M. Heininger, MPI Bonn.

In order to generate all important detector characterization maps, several series of dark and flatfield images are necessary. Experiments have shown that at least 7 series with about 100 images are needed. The illumination level should reach 40 percent of the full well capacity. For example, if saturation is reached after 10 seconds, the recommended exposure times are 600, 1200, 1800, 2400, 3000, 3600, and 4200 ms. For each exposure time, 100 dark images and 100 flatfield images are necessary, adding up to 14 image series, each including 100 images (about 1.5 GBytes). These data requirements are fulfilled by the accompanying recipe.

In Figure B.1.1, a typical cold dark image is shown in the left panel and a typical flatfield image in the right panel. These images are influenced by detector, electronic, and environmental effects, which are highlighted in this figure. On the left side of the cold dark image, a damped oscillation can be seen. The electronic bias shows up as a vertical ramp. One particle event is marked in the dark image (this event is also shown in figures B.1.2 and B.1.3) and in the flatfield image. The nearby cluster of white pixels in the dark image is a cluster of bad pixels, which is found later on by the algorithms described further in this document. Several hot (white) and dark pixels are clearly visible in both images too. The flatfield image on the right shows the same effects as a cold dark image. In addition, the shielded area on the left side of the detector used for calibration purposes and the inhomogeneous illumination caused by the spectrograph are obvious.

All effects visible in Figure B.1.1 must be taken into account when generating a bad pixel map and a flatfield map:

- The damped oscillation and the electronic bias are stable enough to compensate them by using an electronic bias map which contains both effects in one map (see Appendix B.1.4).
- Particle events influence the temporal noise of pixels, and the images which contain such events must be ignored for the affected pixels.
- The shielded area is used to compensate the global detector offset and the 1/f noise (not visible in Figure B.1.1).

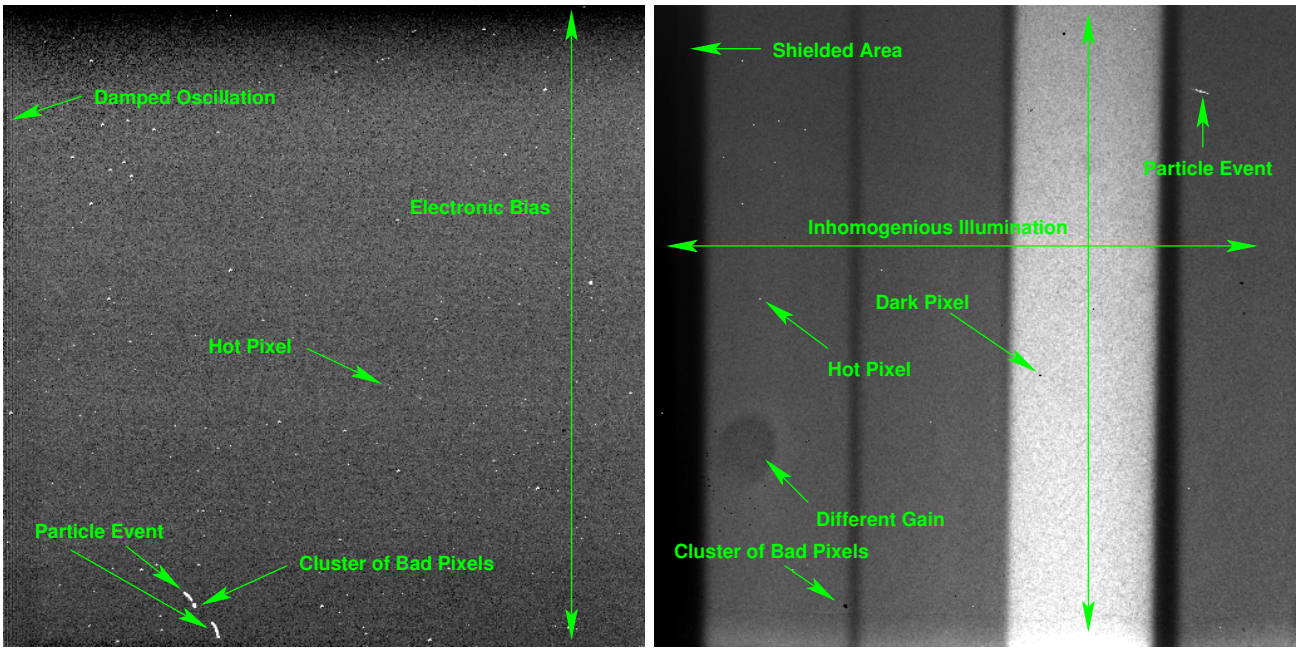

Figure B.1.1: Typical cold dark image on the left and a typical flatfield image on the right.

- The inhomogeneous illumination must be taken into account when calculating the flatfield map because the flatfield map must only contain the different pixel gain.

In order to generate a bad pixel map and a flatfield map the pipeline performs the following tasks:

1. For each series of dark images, a particle event map, an electronic bias map, a pixel bias map, and a pixel statistics map is generated (see Appendix B.1.3, B.1.4, and B.1.5).

2. For each series of flatfield images, a particle event map, a photon noise map, and a pixel statistics map is generated (see Appendix B.1.3 and B.1.5). For this step, the pixel bias maps and the electronic bias maps from the corresponding series of dark images are used.

3. A dark current map from the pixel statistics maps of all series of dark images is generated (see Appendix B.1.6).

4. A bad pixel map is generated using the pixel statistics maps, photon noise maps, and dark current map (see Appendix B.1.7).

5. New pixel statistics maps for the series of flatfield images with compensation for pixel bias, electronic bias, and particle events are generated.

6. A flatfield map from the new pixel statistics maps for the series of flatfield images is generated (see Appendix B.1.8).

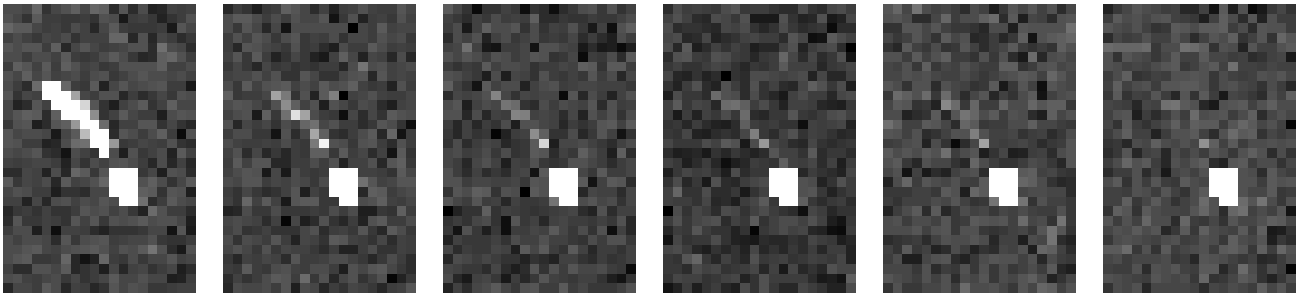

Figure B.1.2: A particle event and afterglow on the detector

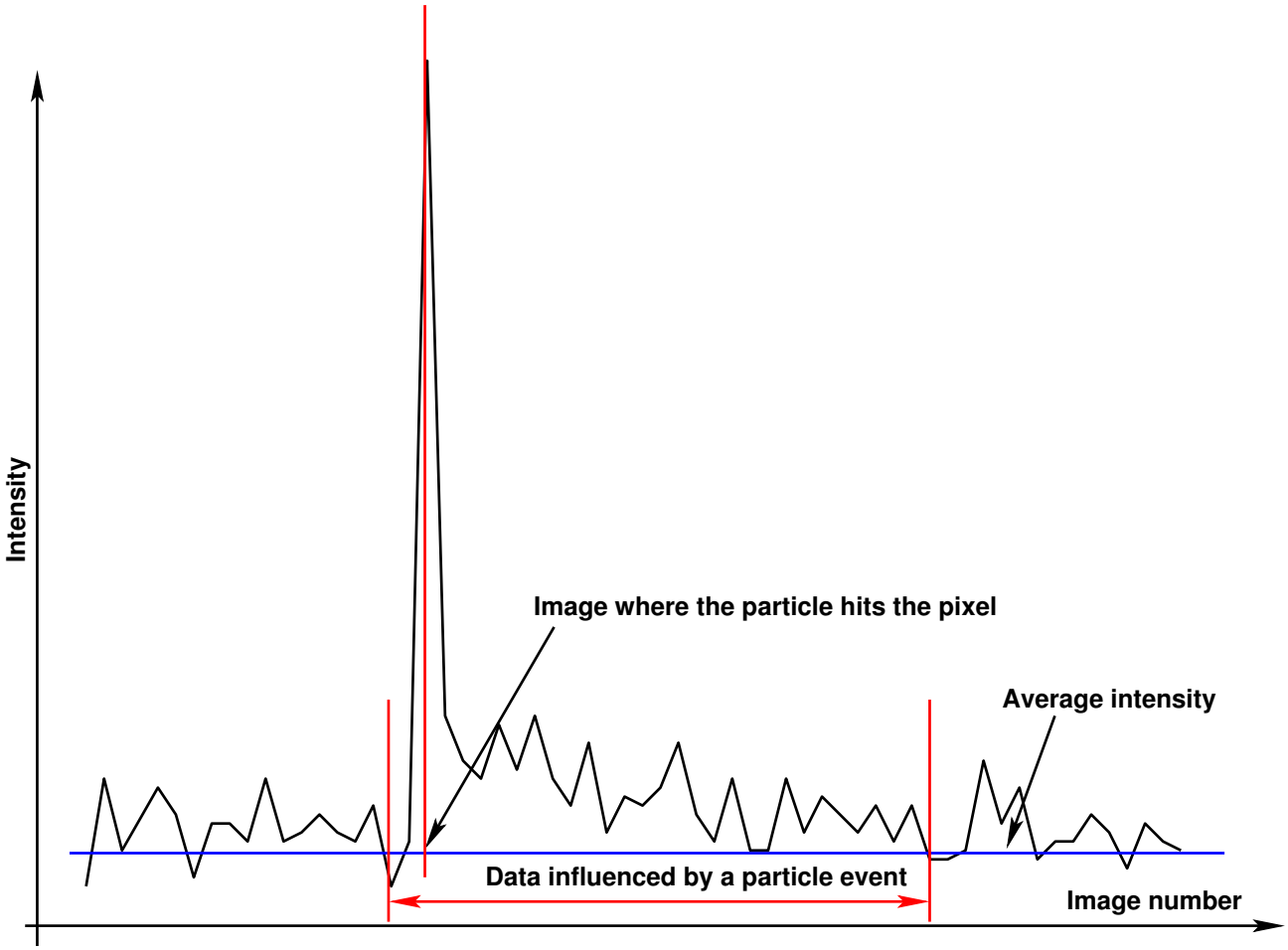

Figure B.1.3: Intensity values of a pixel during a particle event with afterglow

B.1.3 Particle Events

Images taken with the HAWAII-1 detector of the AMBER instrument show random particle events at a rate of about 2 events per quadrant per minute exposure time. This rate seems to be independent of the observing site because images taken in Bonn show the same number of events per minute. These particle events show up as a sudden raise in the intensity of a few pixels. Normally, these pixels build a small cluster of 2-20 pixels. After a few images, the intensity of these pixels drops to the previous value.

In Figure B.1.2 a particle event is shown, where the left panel shows the image when the particle hits the detector. The other panels show the afterglow of the event. The permanently visible cluster is a small cluster of defect pixels on the AMBER detector (chip #159). Figure B.1.3 shows the intensity variations of a pixel during that particle event.

The *amdms* library implements an algorithm which detects particle events under the following conditions:

- More than one pixel is influenced.
- These pixels are connected, which means they are direct neighbors.
- The raise in intensity is larger than a given limit.

If a pixel is influenced by a particle event, the average intensity and the variance has to be recalculated by ignoring the affected images. The algorithm used for the particle event detection is implemented as a filter. This means a pixel has to pass certain tests before a particle event influencing this pixel is accepted and a new mean and variance is calculated:

```
Try to detect particle events, mark the affected images, and recalculate the statistics.
PROC amdmsDetectParticleEvents()
   try to find candidates for particle events (see amdmsFindPEs())
   allocate memory for all candidate pixels and their data values;
   reload the data values for all candidate pixels;
   FOR EACH candidate pixel DO
      analyze the candidate pixel (see amdmsAnalyzePE());
      IF the pixel is still a candidate THEN
         find the affected image interval (see amdmsFindPEInterval());
      ENDIF
   ENDFOR
   FOR EACH candidate pixel DO
      try to verify the particle event (see amdmsVerifyPE());
   ENDFOR
   FOR EACH candidate pixel DO
      recalculate the mean and variance (see amdmsRecalculateMeanVar());
   ENDFOR
ENDPROC
```

Candidate pixels are found by comparing the measured variance with a given limit. For cold dark images, this limit is equal to a noise of $40e^-$; for flatfield images this limit is equal to a noise of $100e^-$. It is also necessary that at least one nearby pixel has a variance which is higher than this limit. This means that isolated pixels with an unusually high variance are treated as bad pixels. It is still possible that a particle event affected only one pixel; in this case that event is not detected, but the pixel is marked as a bad pixel.

```
Try to find candidates for particle events.
PROC amdmsFindPEs()
   FOR EACH pixel DO
      IF the variance of this pixel is less than the limit THEN CONTINUE ENDIF
      IF no nearby pixel shows a variance higher than the limit THEN CONTINUE ENDIF
      mark this pixel as a possible candidate for a particle event;
   ENDFOR
ENDPROC
```

To distinguish between really bad pixels with a high variability and pixels affected by particle events, the following function calculates the mean and variance for the first half and the second half of the pixel values. If there is no significant difference in the variance of both half, the pixel is marked as a bad pixel.

```
Analyze the data of one candidate pixel, which means try to
locate the particle event in the first or second half of the
data values of a pixel.
PROC amdmsAnalyzePE(candidate pixel)
   calculate mean (m) and variance (v) for all data values;
   set a split point at the middle of the data set;
   calculate mean (m_f) and variance (v_f) from 0 up to the split point;
   calculate mean (m_s) and variance (v_s) from the split point up to the end;
   IF v_f < 0.5v AND v_s > 1.5v THEN
      // the particle event is in the second half of the data, mean and variance are
      // recalculated by moving the split point by -20 images (enlarge the second part)
      recalculate mean (m_f) and variance (v_f) from 0 up to the split point;
      recalculate mean (m_s) and variance (v_s) from the split point up to the end;
      IF v_f < 0.5v AND v_s > 1.5v THEN
         // the particle event could be verified in the second part of the data
         store the split point and the end of the data as interval of a particle event;
      ELSE
         remove this candidate pixel from the list;
      ENDIF
   ELSEIF v_s < 0.5v AND v_f > 1.5v THEN
      // the particle event is in the first half of the data, mean and variance are
      // recalculated by moving the split point by +20 images (enlarge the first part)
      recalculate mean (m_f) and variance (v_f) from 0 up to the split point;
      recalculate mean (m_s) and variance (v_s) from the split point up to the end;
      IF v_s < 0.5v AND v_f > 1.5v THEN
         // the particle event could be verified in the first part of the data
         store the split point and the end of the data as interval of a particle event;
      ELSE
         remove this candidate pixel from the list;
      ENDIF
   ELSE
      remove this candidate pixel from the list;
   ENDIF
ENDPROC
```

As shown in Figure B.1.3, the following procedure tries to find the interval of images where the particle hit the pixel and the afterglow of the event affected the measured intensity.

```
Find the images which are affected by the particle event (see Figure B.1.3).
PROC amdmsFindPEInterval(candidate pixel)
   find the peak value inside the interval;
   find the image before the peak where the value is below the mean m;
   find the image after the peak where the value is below the mean m;
   store the first and last image of this interval;
ENDPROC
```

The final test for a candidate pixel is that the interval of the affected images of this pixel and a nearby candidate pixel must overlap. If this is not the case, the pixel is treated as a bad pixel. If there is an overlap, both pixels are treated as pixels affected by the same particle event.

```
Verify the particle event by comparing it with the events found at nearby pixels.
PROC amdmsVerifyPE(candidate pixel)
   IF no nearby pixel is marked as a candidate pixel THEN
      remove this candidate pixel;
      RETURN;
   ENDIF
   IF the interval of a nearby candidate pixel and this pixel has no common subset THEN
      remove this candidate pixel;
      RETURN;
   ENDIF
   define the current candidate pixel and the nearby pixel as a common particle event;
ENDPROC
```

If particle events are found, the mean and variance of the affected pixels are recalculated by ignoring the data showing the particle event and afterglow. A particle event map contains, for each pixel, the first and last image number which defines an interval of images where a pixel is influenced by a particle event. If a pixel shows no such effect, both values are -1. Figure B.1.4 shows an image where all found particle events of one series of dark images are shown.

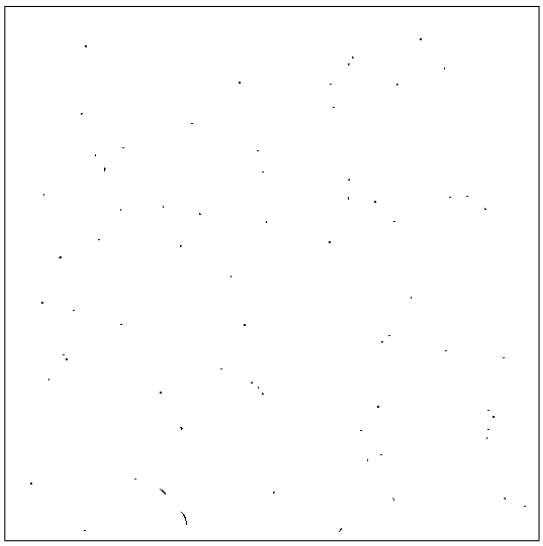

Figure B.1.4: Particle event map for a series of 470 cold dark images with 69 particle events.

B.1.4 Electronic Bias

In the left panel of Figure B.1.1, the effects labeled as *Damped Oscillation* and *Electronic Bias* are effects which can be extracted from a series of cold dark images. The result is an electronic bias map (see Figure B.1.5), which can be used to compensate flatfield images or science images for both effects if the setup (subwindow geometry and exposure time) is equal.

The calculation of the electronic bias uses the pixel statistics (see Appendix B.1.5) calculated from a series of cold dark images. The first step extracts the vertical ramp by calculating the mean intensity of a pixel row, ignoring outliers. This ramp is then subtracted from the pixel statistics. In the second step, the damped oscillation is calculated as the mean intensity of a pixel column, ignoring outliers. These offsets are subtracted from the pixel statistics too. The results are mean intensities where the electronic bias has been removed.

```
Calculate a map containing the damped oscillation and the
electronic bias (vertical ramp) from a series of dark images.
PROC amdmsCalculateElectronicBiasQ1()
   allocate memory for the electronic bias map and initialize it with 0.0;
   // extract the vertical ramp from the pixel statistics
   FOR EACH pixel row DO
      calculate the average intensity for this row using all pixels;
      set the lower and upper limit using the average intensity;
      FOR EACH requested number of iterations DO
         calculate a new average intensity for the current row ignoring pixels
         which have a value below the lower limit or above the upper limit;
         recalculate both limits using the new average intensity;
      ENDFOR
      use the last average intensity as the electronic bias for this pixel row;
      subtract this bias from the pixel statistics of this row;
   ENDFOR
   // extract the damped oscillation from the pixel statistics
   FOR EACH pixel column DO
      calculate the average intensity for this column using all pixels;
      set the lower and upper limit using the average intensity;
      FOR EACH requested number of iterations DO
         calculate a new average intensity for the current column ignoring pixels
         which have a value below the lower limit or above the upper limit;
         recalculate both limits using the new average intensity;
      ENDFOR
      use the last average intensity as the electronic bias for this pixel column;
      subtract this bias from the pixel statistics of this column;
   ENDFOR
ENDPROC
```

The implementation of this algorithm in the *amdms* library ensures that at least half of the pixels in a row or column are used to calculate an average intensity. Figure B.1.5 shows an electronic bias map calculated from a series of cold dark images. In this Figure the damped oscillation on the left side and the electronic bias as a vertical ramp are clearly visible.

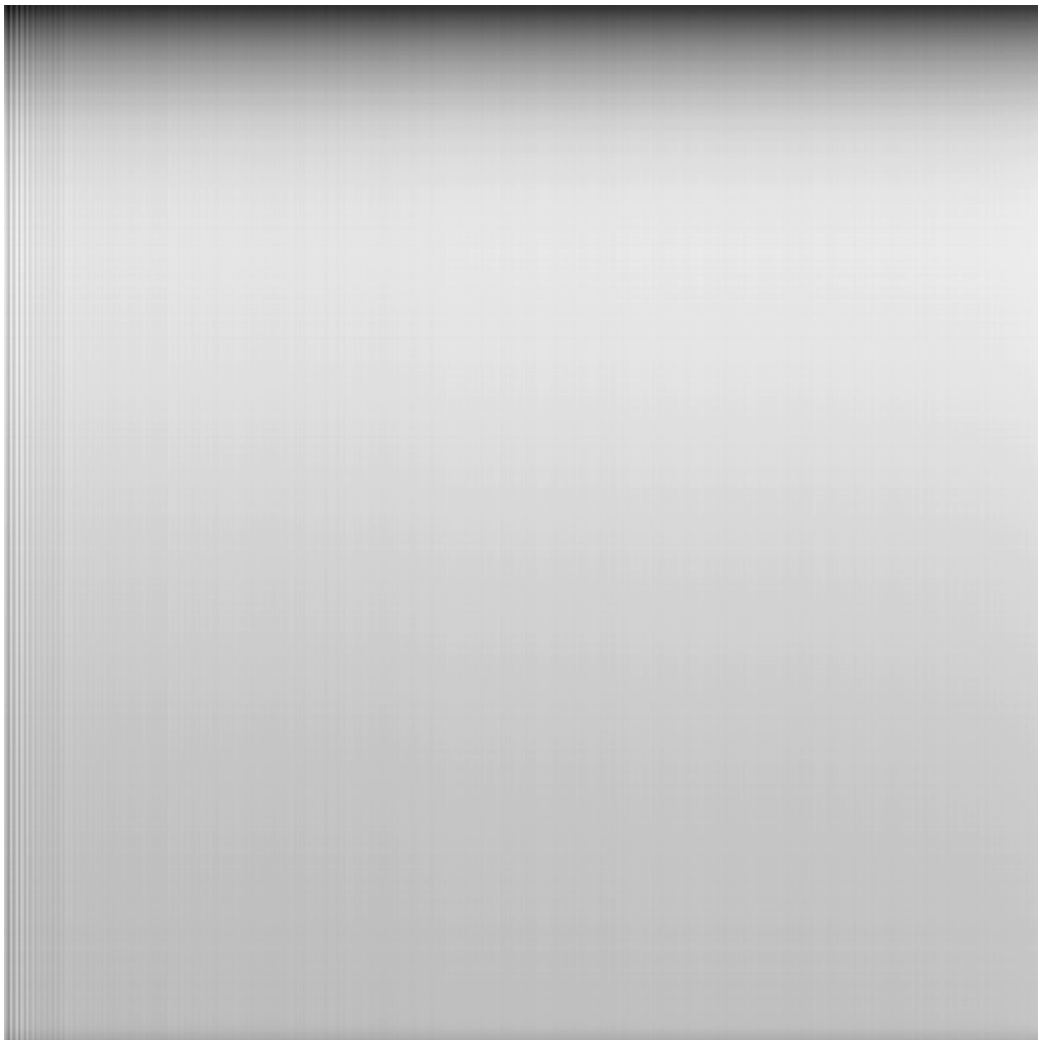

Figure B.1.5: Electronic bias map from a series of dark images with an exposure time of $4.5s$.

B.1.5 Pixel Statistics, Pixel Bias, and Photon Noise

The basis for the calculation of the dark current map (see Appendix B.1.6), the bad pixel map (see Appendix B.1.7), and the flatfield map (see Appendix B.1.8) are the pixel statistics and two derived maps: the pixel bias map and the photon noise map. From each series of cold dark images and flatfield images, the average intensity and temporal variance of each pixel is calculated. These values are calculated by ignoring particle events and subtracting the electronic bias. In Figure B.1.6 the pixel statistics for a series of cold dark images are shown. On the left side, the average intensity shows no electronic bias map, which is obvious if it is compared with a typical cold dark image (see left part of Figure B.1.1) and the corresponding electronic bias map (see Figure B.1.5). The temporal variance on the right side of Figure B.1.6 shows no sign of particle events. This indicates that almost all particle events were found and ignored during the calculation.

The pixel bias map used to compensate for a different "pixel offset" for each pixel is equal to the average mean intensity from the pixel statistics map. The photon noise map contains the relation between the measured

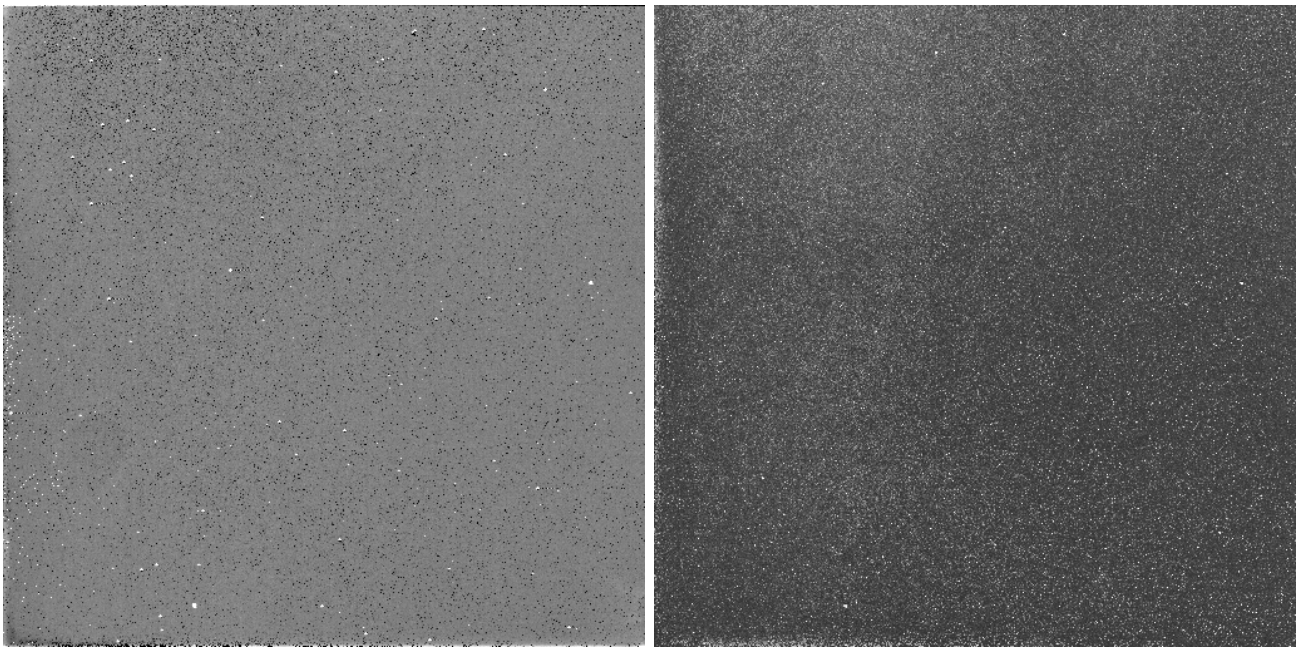

Figure B.1.6: Pixel statistics for a series of cold dark images: **left**: the average intensity for each pixel calculated from a series of 100 cold dark images. The linear scaled image shows values between $-10DU$ and $+10DU$; **right**: the temporal variance for each pixel is shown in a linear scaled image with 0 and $+30DU^2$ as limits.

temporal variance and the expected photon noise for each pixel and each series of flatfield images:

$$PN_i = \sqrt{cf\frac{v_i}{m_i}}$$

where PN_i is the photon noise value of a pixel in the photon noise map, cf is the conversion factor of $4.18\frac{e^-}{DU}$, v_i is the temporal variance, and m_i is the average intensity from the ith pixel. In Figure B.1.8 a photon noise map is shown.

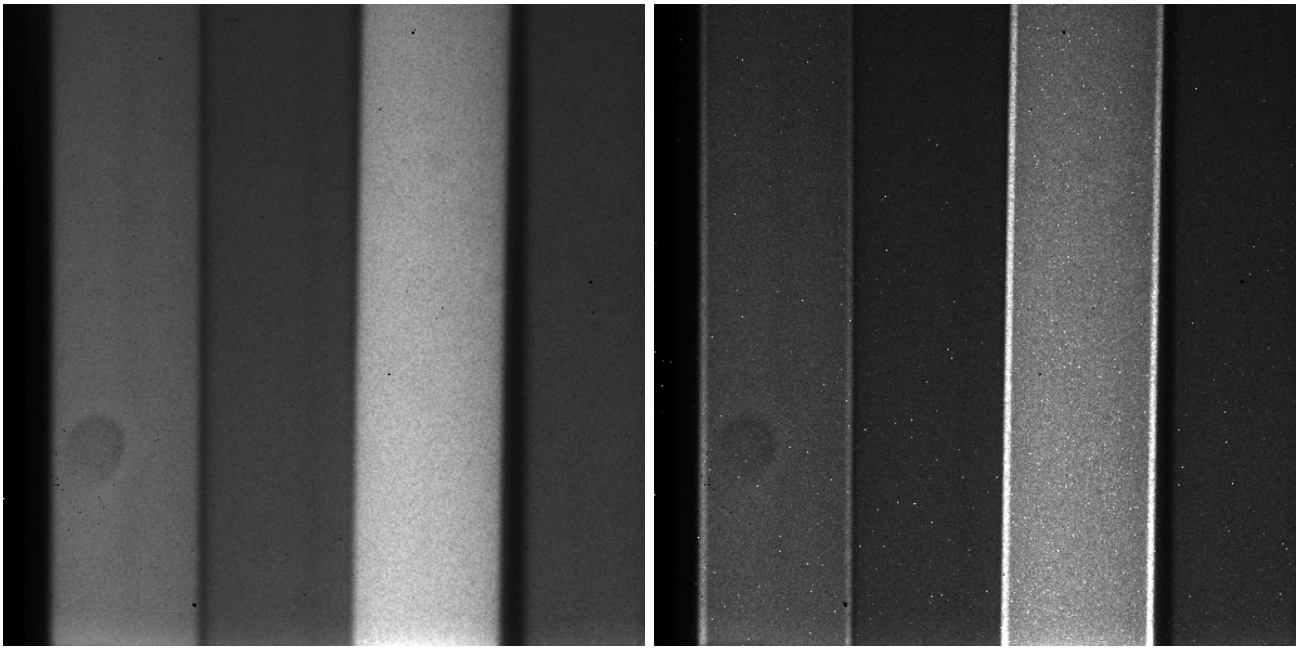

Figure B.1.7: Pixel statistics for a series of flatfield images: **left**: the average intensity for each pixel calculated from a series of 100 flatfield images. The linear scaled image shows values between $0DU$ and $+9000DU$; **right**: the temporal variance for each pixel is shown in a linear scaled image with 0 and $+3000DU^2$ as limits.

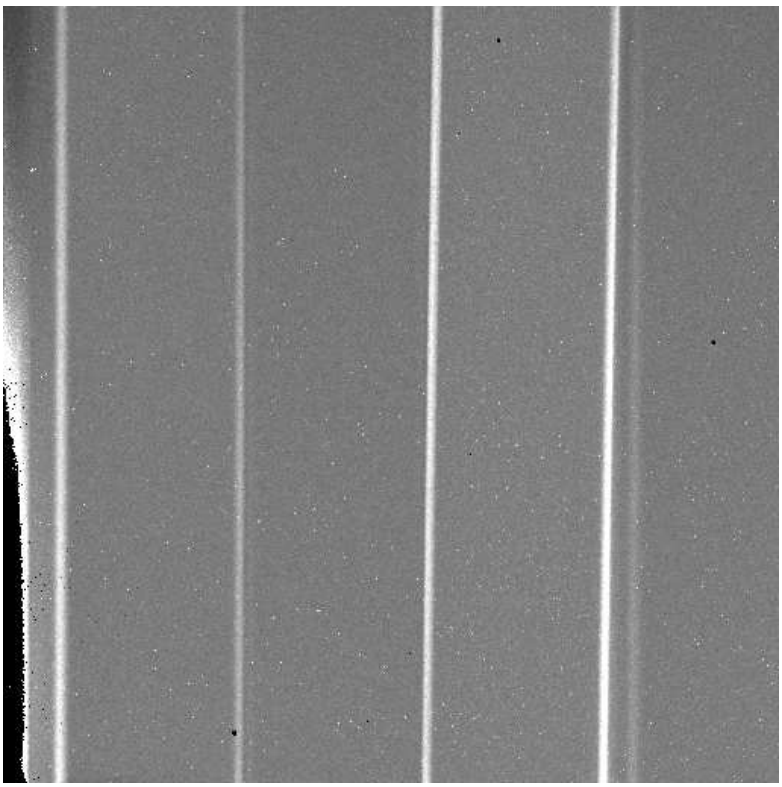

Figure B.1.8: Photon noise map. The image has a linear scale with the limits 0.0 and 2.0.

B.1.6 Dark Current

In the typical cold dark image on the left side of Figure B.1.1, some white (hot) and dark pixels can be seen. These pixels do not have a different offset than the other pixels, but the reason for this effect is the following:

When a pixel is reset, a certain number of electrons (about 100 000 - 120 000) are injected into the capacitor of a pixel. If photons are collected by a pixel, electrons are removed from the pixel capacitor. The measured intensity is equal to the difference between those two states. That means if a pixel shows a large value even without light, the capacitor is not perfect; some electrons are lost during the exposure time. The dark pixels have negative values, which means they collect electrons lost by nearby pixel capacitors. This so-called dark current shows up as white and dark pixels in cold dark images. The dark current map trys to quantify this effect.

The dark current map consists of an offset, slope, and fit quality (χ^2) for each pixel. The offset and slope define a straight line where the units are $[DU]$ and $[DU/s]$. During the generation of the bad pixel map (see Appendix B.1.7), upper and lower limits for the slope and upper limits for χ^2 are used to detect pixels with an unusually high dark current or with a nonlinear dark current.

The input data for the dark current map is a set of pixel statistics maps from the series of cold dark images. Then, a straight line is fitted through the mean intensities for each pixel. The offset, slope, and fit quality are stored in the dark current map. The left panel of Figure B.1.9 shows the offset, the panel in the middle the slope, and the right panel the fit quality.

Figure B.1.9: Dark current map for the AMBER detector: **left**: the dark current map contains the offset of the fitted straight line. The image has a linear scale where the lower and upper limits are $-75e^-$ and $75e^-$; **middle**: the slope of the straight line is shown in this linear scaled image. The lower and upper limits are $-75e^-/s$ and $75e^-/s$, which are the limits used for the bad pixel map generation; **right**: the fit quality (χ^2) is shown in this linear scaled image. The limits are 0.0 and 5.0, which are used for the bad pixel map generation.

B.1.7 Bad Pixel Map

The pipeline step which generates a bad pixel map uses all intermediate data described in the previous sections. The parameters used for the AMBER detector are shown in table B.1.0.

1. The first step uses the variance values from all pixel statistics maps of all series of cold dark images.

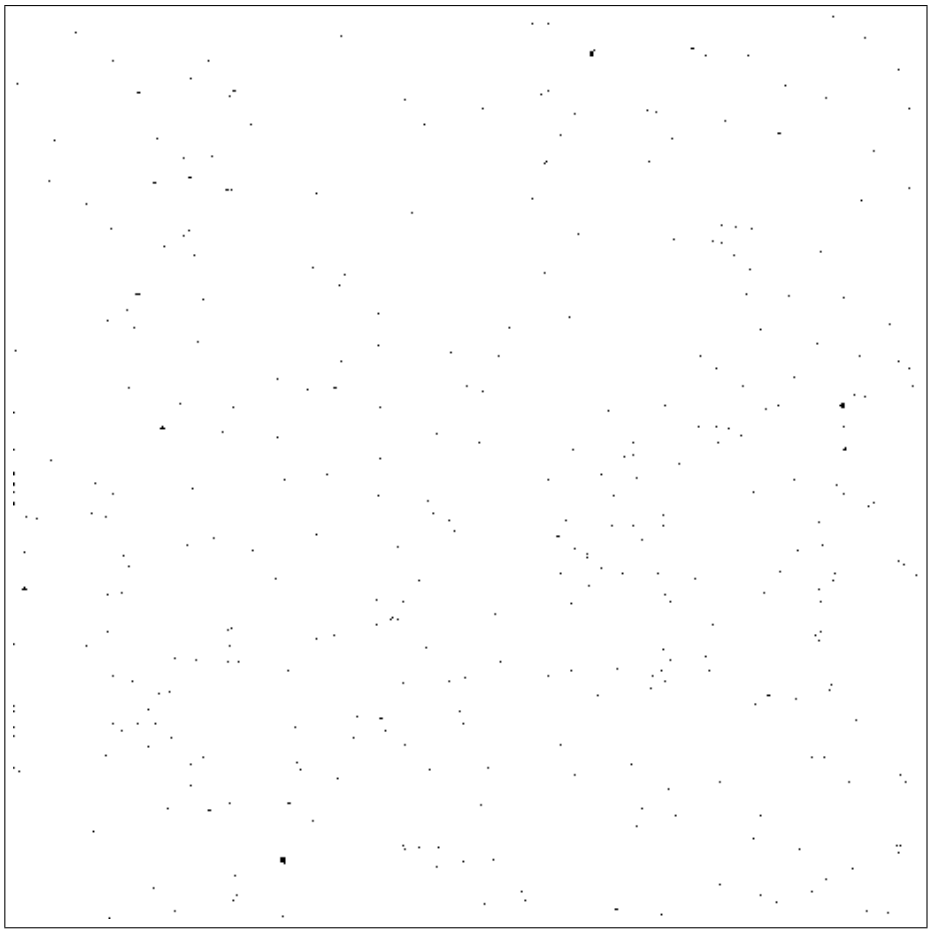

Figure B.1.10: Badpixel map

Pixels which have a variance lower than the minimum value (so-called "dead pixels") are marked as bad pixels. Pixels which have a variance higher than the maximum value are noisier than a normal pixel and also marked as bad pixels.

2. The second step uses the same criteria for the variance, but this time uses values from all pixel statistics of all series of flatfield images. These images have a shielded area where the conditions are similar to cold dark images and, therefore, the same limits are used for pixels in this area. The illuminated part of the flatfield images is ignored in this step.

3. The third step uses the photon noise values in the photon noise maps generated from all series of flatfield images. If the photon noise of a pixel is less than the minimum value or greater than a maximum value, this pixel is marked as a bad pixel.

4. The fourth step uses the slope value in the dark current map. If a pixel has a slope which is less than the minimum or greater than a maximum value, this pixel is marked as a bad pixel.

5. The fifth step uses the χ^2 value in the dark current map. If the fit quality is worse than a certain maximum value, this pixel is marked as a bad pixel.

6. The last step uses the average intensity of a pixel compared to the local average. This local value is computed by using all pixels in a local array of pixels which covers 11 times 11 pixels, but the inner part of 3 times 3 pixels is ignored.

Table B.1.0: Parameters for the bad pixel map generation

Step	Criteria	Min value	Reference value	Maximum value
1	Noise in the cold dark images	$0.23 \equiv 2.0e^-$	$6.93 \equiv 11.0e^-$	$321.94 \equiv 75.0e^-$
2	Noise in the shielded area in the flat-field images	$0.23 \equiv 2.0e^-$	$6.93 \equiv 11.0e^-$	$321.94 \equiv 75.0e^-$
3	Photon noise inside vertical stripes	0.5	1.0	2.25
4	Slope in the dark current map dark	$-17.94 \equiv -75.0e^-/s$	$0.0 \equiv 0.0e^-/s$	$17.94 \equiv 75.0e^-/s$
5	χ^2 in the dark current map		0.0	5.0
6	Local average intensity in the flat-field images	-5.0σ	0.0σ	5.0σ

In Figure B.1.10 a bad pixel map is shown which was calculated using the images taken during commissioning 2, 22.10.2004 at Paranal. Except for a few clusters, all bad pixels are isolated and do not interfere with the science data taken in the four vertical stripes (see right part of Figure B.1.1).

B.1.8 Flatfield Map

A flatfield map for the AMBER detector is calculated by using a set of pixel statistic maps from several series of flatfield images (see left side of Figure B.1.7). For each pixel, a straight line is fitted through the data values, and the normalized slope is used for the flatfield map. A closer look at a typical flatfield image (see right side of figure B.1.1) or a pixel statistics map (see left side of Figure B.1.7) shows that the illumination of the AMBER detector is affected by at least the following effects:

- Only four vertical stripes are illuminated on the detector and the illumination level is different between the four stripes.
- The illumination shows a ramp in vertical direction.
- The illumination shows a horizontal profile, with ramps on both sides of a vertical stripe.

Because of these effects, a normal calculation of the flatfield map is not possible since only a detector flatfield and not an instrument flatfield is required. The effects mentioned above are taken into account by:

1. Treating each vertical stripe separately.
2. Compensating for the vertical ramp and horizontal profile by using a simple model of the light distribution.

The algorithm used to neutralize the ramp and profile can be described as follows:

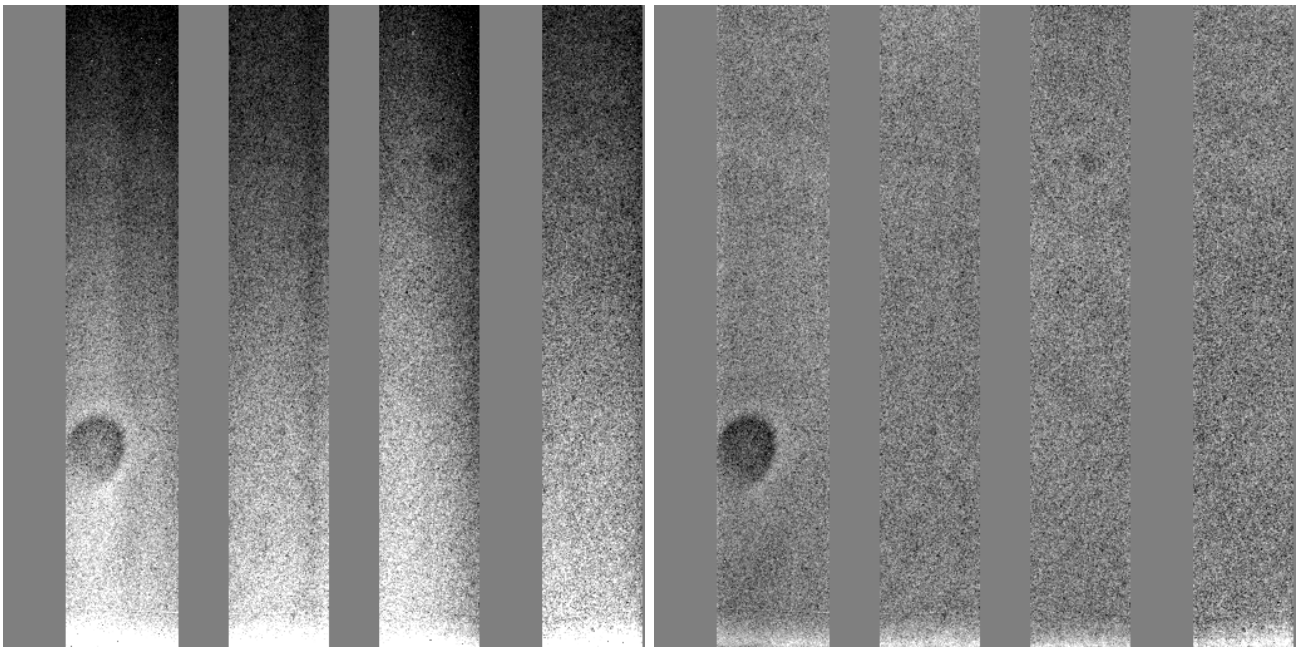

Figure B.1.11: Flatfield map for the AMBER detector: **Left**: the image shows a flatfield map without compensation; **Right**: the image shows a flatfield map with compensation and, therefore, represents only the gain variations of the AMBER detector pixels. Both images have a linear scale with 0.7 and 1.3 as limits.

```
Compensate for the vertical ramp and horizontal profile.
PROC amdmsCleanUpFlatfieldSmooth()
   // calculate a horizontal profile in the lower and upper side of the detector
   FOR EACH pixel column i DO
      calculate the average intensity for 40 pixels at rows 80 - 120
      and use this value as part of the lower profile LP_i;
      calculate the average intensity for 40 pixels at rows 392 - 432
      and use this value as part of the upper profile UP_i;
   ENDFOR
   // calculate smoothened profiles (see amdmsSmoothDataByFiniteDiff1)
   smooth the lower profile by using finite differences;
   smooth the upper profile by using finite differences;
   calculate a center profile as average of the lower and upper profile CP_i;
   // compensate for the vertical ramp
   FOR EACH pixel column i DO
      calculate the slope of a straight line as a = (UP_i - LP_i)/312;
      FOR EACH pixel row j DO
         calculate a compensation factor f = CP_i/(LP_i + a(j - 100));
         calculate a new pixel mean as m = m × f and a new pixel variance as v = v × f × f;
      ENDFOR
   ENDFOR
   // compensate for the profile
   FOR EACH vertical stripe k DO
      calculate the average m of CP_i for all pixel columns inside stripe k;
      FOR EACH pixel column i in the current stripe DO
         calculate a compensation factor f = m/CP_i;
         calculate new mean and variance values for the pixel column i;
      ENDFOR
   ENDFOR
ENDPROC
```

Figure B.1.11 shows two flatfield maps; one where the vertical ramp and horizontal profile are compensated for, and one without compensation. The gray areas in both images are those areas on the detector without enough light to calculate a useful flatfield value.

B.1.9 Flat field correction

The flat field correction merely consists of dividing the frame to be corrected by a given master lamp flat field frame produced by the recipe *amber_detector*

It is currently under discussion if the flat field is implicit with the p2vm correction.

B.2 AMBER-specific Algorithms

For a detailed description of AMBER-specific algorithms please refer to the AMBER data reduction plan, VLT-PLA-AMB-15830-6004.

The following description is a very minute introduction taken from the document before mentioned.

B.2.1 P2VM calculation

The fringe pattern produced on the camera by any two pair of entrance beams, in any of the (up to 3) spectral bands imaged, must be calibrated beforehand, since it depends strongly on the current instrument configuration, especially the position of beams wrt. the pupil masks, and the position of the dispersing elements. Calibrating this fringe pattern is called calibrating the Pixel-to-Visibility Matrix (the P2VM). The AMBER OS will prevent the acquisition of science data when the corresponding P2VM has not been calibrated.

A P2VM calibration is performed by obtaining one frame with all shutters closed to get a sky-type frame, then by opening in turn only one shutter to get the shape of the illumination in the interferometric channel due to one fiber (the so-called vik), then by opening in turn pairs of shutters to retrieve the interference fringe pattern for each pair of telescope, with and without a phase shift of a known value.

0. The visibility of the artificial source (lamp) in the CAU is supposed to be fixed, calibrated and noted V0.
1. CAU Lamp: ON
2. CAU mirror on CAU position.
3. Shutter 1 Close, 2 Close, 3 Close
4. take image File: AMBER 3TSTD CAL 0001.fits
5. Shutter 1 Open, 2 Close, 3 Close
6. take image File: AMBER 3TSTD CAL 0002.fits
7. Shutter 1 Close, 2 Open, 3 Close
8. take image File: AMBER 3TSTD CAL 0003.fits
9. Phase Plate position Off (INS OPTI4 NAME = DEL NO)
10. Shutter 1 Open, 2 Open, 3 Close
11. take image File: AMBER 3TSTD CAL 0004.fits
12. Phase Plate position On (INS OPTI4 NAME = DEL1 2)
13. Shutter 1 Open, 2 Open, 3 Close
14. take image File: AMBER 3TSTD CAL 0005.fits
15. Shutter 1 Close, 2 Close, 3 Open
16. take image File: AMBER 3TSTD CAL 0006.fits
17. Phase Plate position Off (INS OPTI4 NAME = DEL NO)
18. Shutter 1 Close, 2 Open, 3 Open
19. take image File: AMBER 3TSTD CAL 0007.fits
20. Phase Plate position On (INS OPTI4 NAME = DEL12 3)
21. Shutter 1 Close, 2 Open, 3 Open
22. take image File: AMBER 3TSTD CAL 0008.fits
23. Phase Plate position Off (INS OPTI4 NAME = DEL NO)
24. Shutter 1 Open, 2 Close, 3 Open
25. take image File: AMBER 3TSTD CAL 0009.fits
26. Phase Plate position On (INS OPTI4 NAME = DEL12 3)
27. Shutter 1 Open, 2 Close, 3 Open
28. take image File: AMBER 3TSTD CAL 0010.fits

Note that the 5 first steps are needed for a 2 Telescope P2VM. Those calibration frames are processed with amdlib routines to produce a P2V Matrix used for all following visibility extraction, until the Instrument Setup changes and a new P2VM is needed.

Of the previously described calibration frames, the first is a sky-like measurement, without input light. This is done in order to subtract the uncorrelated light sky due to the thermal emission of the fibers, if any. The next three frames are exposures with only one input beam. This allows one to compute the vk coefficients in the equations described before, by dividing the interferometric channel by the measured photometric flux, pixel by pixel and for every input beam. The other 6 frames completely determine the calibration of the instrument. In these frames, two input beams at a time are combined. Per baseline two exposures are made. In one of this couple of frames, an additional phase shift is inserted into one of the beams.

C Changes in the Amdlib

In the following sections we show the differences in the product file of the amber_p2vm and the amber_SciCal recipes due to the change in the Amdlib. For every extension of the product file the old output (left hand side) is compared to the new output (right hand side).

C.1 Amdlib changes in the amber_p2vm recipe

```
Extension 0:
====> file p2vm.fits.0 (main) <====                          | ====> file p2vm.fits.1 (main) <====
DATAMD5 = '6e0783b655188e644dc79a58ab8a8b14' / MD5 checksum   | DATAMD5 = 'e7878b74ed632c31d91ac435e7b88f09' / MD5 checksum
DATE    = '2007-12-21T14:43:12' / Date this file was written  | DATE    = '2008-03-12T16:23:03' / Date this file was written
HIERARCH ESO OCS DRS VERSION = '1.1.1.1 ' / Data Reduction SW | HIERARCH ESO OCS DRS VERSION = '2.0     ' / Data Reduction SW
HIERARCH ESO OCS P2VM ERRVIS12 = 4.500827E-06 / Inst. vis. er <
HIERARCH ESO OCS P2VM ERRVIS23 = 6.578008E-06 / Inst. vis. er <
HIERARCH ESO OCS P2VM ERRVIS31 = 1.671419E-06 / Inst. vis. er <
HIERARCH ESO OCS P2VM VIS12 = 0.7978971 / Inst. visibility fo <
HIERARCH ESO OCS P2VM VIS23 = 0.8500246 / Inst. visibility fo <
HIERARCH ESO OCS P2VM VIS31 = 0.6827367 / Inst. visibility fo <
                                                             > HIERARCH ESO PRO REC1 CAL1 DATAMD5 = '89141505fc7e0573dbc729a
                                                             > HIERARCH ESO PRO REC1 CAL2 DATAMD5 = '8cdd7d20f730354066e0941
HIERARCH ESO PRO REC1 DRS ID = 'cpl-4.1.0a1' / Data Reduction | HIERARCH ESO PRO REC1 DRS ID = 'cpl-4.1.0cvs' / Data Reductio
HIERARCH ESO PRO REC1 PIPE ID = 'amber/2.4.2' / Pipeline (uni | HIERARCH ESO PRO REC1 PIPE ID = 'amber/2.4.3' / Pipeline (uni
                                                             > HIERARCH ESO PRO SCIENCE =   F / Scientific product if T
                                                             > HIERARCH ESO PRO TECH = 'INTERFEROMETRY' / Observation techni
                                                             > HIERARCH ESO QC P2VM ERRVIS12 = 4.533544E-06 / Inst. vis. err
                                                             > HIERARCH ESO QC P2VM ERRVIS23 = 6.637083E-06 / Inst. vis. err
                                                             > HIERARCH ESO QC P2VM ERRVIS31 = 1.687855E-06 / Inst. vis. err
                                                             > HIERARCH ESO QC P2VM H ERRVIS12 = 0. / Inst. vis. error for 1
                                                             > HIERARCH ESO QC P2VM H ERRVIS23 = 0. / Inst. vis. error for 2
                                                             > HIERARCH ESO QC P2VM H ERRVIS31 = 0. / Inst. vis. error for 3
                                                             > HIERARCH ESO QC P2VM H VIS12 = 0. / Inst. vis. for 12 in H
                                                             > HIERARCH ESO QC P2VM H VIS23 = 0. / Inst. vis. for 23 in H
                                                             > HIERARCH ESO QC P2VM H VIS31 = 0. / Inst. vis. for 31 in H
                                                             > HIERARCH ESO QC P2VM J ERRVIS12 = 0. / Inst. vis. error for 1
                                                             > HIERARCH ESO QC P2VM J ERRVIS23 = 0. / Inst. vis. error for 2
                                                             > HIERARCH ESO QC P2VM J ERRVIS31 = 0. / Inst. vis. error for 3
                                                             > HIERARCH ESO QC P2VM J VIS12 = 0. / Inst. vis. for 12 in J
                                                             > HIERARCH ESO QC P2VM J VIS23 = 0. / Inst. vis. for 23 in J
                                                             > HIERARCH ESO QC P2VM J VIS31 = 0. / Inst. vis. for 31 in J
                                                             > HIERARCH ESO QC P2VM K ERRVIS12 = 4.533544E-06 / Inst. vis. e
                                                             > HIERARCH ESO QC P2VM K ERRVIS23 = 6.637083E-06 / Inst. vis. e
                                                             > HIERARCH ESO QC P2VM K ERRVIS31 = 1.687855E-06 / Inst. vis. e
                                                             > HIERARCH ESO QC P2VM K VIS12 = 0.8035409 / Inst. vis. for 12
                                                             > HIERARCH ESO QC P2VM K VIS23 = 0.8512553 / Inst. vis. for 23
                                                             > HIERARCH ESO QC P2VM K VIS31 = 0.683566 / Inst. vis. for 31 i
                                                             > HIERARCH ESO QC P2VM VIS12 = 0.8035409 / Inst. visibility for
                                                             > HIERARCH ESO QC P2VM VIS23 = 0.8512553 / Inst. visibility for
                                                             > HIERARCH ESO QC P2VM VIS31 = 0.683566 / Inst. visibility for

Extension 1:
====> file p2vm.fits.0 (main) <====                          | ====> file p2vm.fits.1 (main) <====
EXTNAME = 'P2VM_MATRIX'        / name of this binary table ex | EXTNAME = 'P2VM    '           / name of this binary table ex
TTYPE1  = 'Wavelength'         / label for field   1          | TTYPE1  = 'EFF_WAVE'           / label for field   1
```

```
TTYPE2  = 'Matrix  '           / label for field   2          | TTYPE2  = 'MATRIX  '           / label for field   2
TTYPE3  = 'Vk      '           / label for field   3          | TTYPE3  = 'VK      '           / label for field   3
TTYPE4  = 'Sum_of_Vk'          / label for field   4          | TTYPE4  = 'SUM_VK  '           / label for field   4
TTYPE5  = 'Bad_pixels'         / label for field   5          | TTYPE5  = 'BAD_PIXELS'         / label for field   5
TTYPE6  = 'Photometry'         / label for field   6          | TTYPE6  = 'PHOTOMETRY'         / label for field   6
TTYPE7  = 'Validity_flags'     / label for field   7          | TTYPE7  = 'FLAG    '           / label for field   7
TTYPE8  = 'Phase   '           / label for field   8          | TTYPE8  = 'CALIB_PHASE'        / label for field   8
TTYPE9  = 'FlatField'          / label for field   9          | TTYPE9  = 'FLAT_FIELD'         / label for field   9
TUNIT1  = 'nm      '           / physical unit of field       | TUNIT1  = 'm       '           / physical unit of field
TUNIT8  = 'rad     '           / physical unit of field       | TUNIT8  = 'deg     '           / physical unit of field
```

C.2 Amdlib changes in the amber_SciCal recipe

```
Extension 0:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
DATAMD5 = '44e6e50d318223b1c676a138439a0e1a' / MD5 checksum   | DATAMD5 = '3e7dc82521e5932776a482428bbdc157' / MD5 checksum
DATE    = '2007-12-21T14:36:43' / file creation date (YYYY-MM | DATE    = '2008-03-12T15:53:29' / file creation date (YYYY-MM
HIERARCH ESO OCS DRS VERSION = '1.1.1.1 ' / Data Reduction SW | HIERARCH ESO OCS DRS VERSION = '2.0     ' / Data Reduction SW
HIERARCH ESO PRO REC1 DRS ID = 'cpl-4.1.0a1' / Data Reduction | HIERARCH ESO PRO REC1 DRS ID = 'cpl-4.1.0cvs' / Data Reductio
HIERARCH ESO PRO REC1 PIPE ID = 'amber/2.4.2' / Pipeline (uni | HIERARCH ESO PRO REC1 PIPE ID = 'amber/2.4.3' / Pipeline (uni
                                                             > HIERARCH ESO PRO SCIENCE =   F / Scientific product if T
                                                             > HIERARCH ESO PRO TECH = 'INTERFEROMETRY' / Observation techni
HIERARCH ESO QC FRAMES SELECTED PERCENT = 0. / Frames selecte | HIERARCH ESO QC FRAMES SELECTED PERCENT = 11.4 / Frames selec
HIERARCH ESO QC FRAMES SELECTED SNRGT2 = 0 / Frames selected  | HIERARCH ESO QC FRAMES SELECTED SNRGT2 = 114 / Frames selecte

Extension 1:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
ARRAYX  =            -5483173. / [m] Array centre x coordinat | ARRAYX  =             1951952. / [m] Array centre x coordinat
ARRAYY  =             1951952. / [m] Array centre y coordinat | ARRAYY  =            -5483173. / [m] Array centre y coordinat

Extension 2:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====

Extension 3:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
INSNAME = 'AMBER   '           / Spectral Setup unique Id     | INSNAME = 'AMBER   '           / Instrument name

Extension 4:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
INSNAME = 'AMBER   '           / Detector Name                | INSNAME = 'AMBER   '           / Instrument name
NAXIS1  =                  862 / width of table in bytes      | NAXIS1  =                  879 / width of table in bytes
TFORM14 = '0L      '           / data format of field: 1-byte | TFORM14 = '17L     '           / data format of field: 1-byte

Extension 5:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
INSNAME = 'AMBER   '           / Detector Name                | INSNAME = 'AMBER   '           / Instrument name
NAXIS1  =                  318 / width of table in bytes      | NAXIS1  =                  335 / width of table in bytes
```

```
TFORM10 = '0L      '           / data format of field: 1-byte | TFORM10 = '17L     '           / data format of field: 1-byte

Extension 6:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
INSNAME = 'AMBER   '           / Detector Name                | INSNAME = 'AMBER   '           / Instrument name
NAXIS1  =                  608 / width of table in bytes      | NAXIS1  =                  625 / width of table in bytes
TFORM14 = '0L      '           / data format of field: 1-byte | TFORM14 = '17L     '           / data format of field: 1-byte

Extension 7:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
HIERARCH ESO OCS BAND1 LOWBOUND = 1. / Lower bound of spectra | HIERARCH ESO QC BAND1 LOWBOUND = 1. / Lower bound of spectral
HIERARCH ESO OCS BAND1 NAME = 'J       ' / Name of spectral b | HIERARCH ESO QC BAND1 NAME = 'J       ' / Name of spectral ba
HIERARCH ESO OCS BAND1 UPBOUND = 1.469 / Upper bound of spect | HIERARCH ESO QC BAND1 UPBOUND = 1.469 / Upper bound of spectr
HIERARCH ESO OCS NBBANDS =   1 / Number of spectral bands     | HIERARCH ESO QC NBBANDS =    1 / Number of spectral bands
INSNAME = 'AMBER   '           / Detector name                | INSNAME = 'AMBER   '           / Instrument name
NAXIS1  =                  590 / width of table in bytes      | NAXIS1  =                  726 / width of table in bytes
TFORM10 = '17E     '           / data format of field: 4-byte | TFORM10 = '1E      '           / data format of field: 4-byte
TFORM11 = '17E     '           / data format of field: 4-byte | TFORM11 = '1E      '           / data format of field: 4-byte
TFORM12 = '17D     '           / data format of field: 8-byte | TFORM12 = '1D      '           / data format of field: 8-byte
TFORM7  = '1E      '           / data format of field: 4-byte | TFORM7  = '17D     '           / data format of field: 8-byte
TFORM8  = '1E      '           / data format of field: 4-byte | TFORM8  = '17D     '           / data format of field: 8-byte
TFORM9  = '1D      '           / data format of field: 8-byte | TFORM9  = '17D     '           / data format of field: 8-byte
TTYPE10 = 'PISTON_OPD_DISPERSED' / label for field  10        | TTYPE10 = 'OPD     '           / label for field  10
TTYPE11 = 'PISTON_ERR_DISPERSED' / label for field  11        | TTYPE11 = 'OPD_ERR '           / label for field  11
TTYPE12 = 'FRINGE_CONTRAST_SNR_DISPERSED' / label for field   | TTYPE12 = 'FRINGE_SNR'         / label for field  12
TTYPE5  = 'FLUX_PI_MORE_PJ'    / label for field   5          | TTYPE5  = 'FLUX_SUM'           / label for field   5
TTYPE6  = 'FLUX_PI_MORE_PJ_ERR' / label for field   6         | TTYPE6  = 'FLUX_SUM_CORRECTION' / label for field   6
TTYPE7  = 'PISTON_OPD'         / label for field   7          | TTYPE7  = 'FLUX_RATIO'         / label for field   7
TTYPE8  = 'PISTON_ERR'         / label for field   8          | TTYPE8  = 'FLUX_RATIO_CORRECTION' / label for field   8
TTYPE9  = 'FRINGE_CONTRAST_SNR' / label for field   9         | TTYPE9  = 'FLUX_PRODUCT'       / label for field   9
TUNIT7  = 'm       '           / physical unit of field       | TUNIT9  = 'e-^2    '           / physical unit of field
TUNIT8  = 'm       '           / physical unit of field      <

Extension 8:
====> file amber_0000.fits.0 (main) <====                     | ====> file amber_0000.fits.1 (main) <====
                                                              > ===> xtension 8
                                                              > AMB_REVN=                    1 / Revision number of the table
                                                              > BITPIX  =                    8 / 8-bit bytes
                                                              > END
                                                              > EXTNAME = 'AMBER_SPECTRUM'     / name of this binary table ex
                                                              > GCOUNT  =                    1 / one data group (required key
                                                              > INSNAME = 'AMBER   '           / Instrument name
                                                              > NAXIS   =                    2 / 2-dimensional binary table
                                                              > NAXIS1  =                   56 / width of table in bytes
                                                              > NAXIS2  =                   17 / number of rows in table
                                                              > PCOUNT  =                    0 / size of special data area
                                                              > TFIELDS =                    4 / number of fields in each row
                                                              > TFORM1  = 'E       '           / data format of field: 4-byte
                                                              > TFORM2  = 'E       '           / data format of field: 4-byte
                                                              > TFORM3  = '3D      '           / data format of field: 8-byte
                                                              > TFORM4  = '3D      '           / data format of field: 8-byte
```

```
> TTYPE1  = 'EFF_WAVE'           / label for field   1
> TTYPE2  = 'EFF_BAND'           / label for field   2
> TTYPE3  = 'SPECTRUM'           / label for field   3
> TTYPE4  = 'SPECTRUM_ERROR'     / label for field   4
> TUNIT1  = 'm       '           / physical unit of field
> TUNIT2  = 'm       '           / physical unit of field
> XTENSION= 'BINTABLE'           / binary table extension
```

D QC1 Dictionary

The following tables give a list of the Quality Control parameters of the various recipes.

D.1 QC parameter of the amber_SciCal recipe

QC Parameter	Description
FRAMES SELECTED SNRGT2	the number of frames with signal to noise greater than 2
FRAMES SELECTED PERCENT	the percentage of frames with signal to noise greater than 2
FRAMES BAND	the band (J, H, K)
STA1	telescope station 1
STA2	telescope station 2
STA3	telescope station 3 (in 3 telescopes mode only)
LAMBDA CHAN	total number of channels in the file
LAMBDA MIN	lowest waveband in the file [micron]
LAMBDA MAX	highest waveband in the file [micron]
LAMBDA CEN MIN	lowest analyzed waveband in the central bin
LAMBDA CEN MAX	highest analyzed waveband in the central bin
LAMBDA CEN CHAN	total number of analyzed channels in the central bin
LAMBDA BIN1 MIN	lowest analyzed waveband in bin 1 [micron]
LAMBDA BIN1 MAX	highest analyzed waveband in bin 1 [micron]
LAMBDA BIN1 CHAN	total number of analyzed channels in bin 1
LAMBDA BIN2 MIN	lowest analyzed waveband in bin 2 [micron]
LAMBDA BIN2 MAX	highest analyzed waveband in bin 2 [micron]
LAMBDA BIN2 CHAN	total number of analyzed channels in bin 2
LAMBDA BIN3 MIN	lowest analyzed waveband in bin 3 [micron]
LAMBDA BIN3 MAX	highest analyzed waveband in bin 3 [micron]
LAMBDA BIN3 CHAN	total number of analyzed channels in bin 3
—V2 CEN BAS1	averaged uncalibrated squared visibility in the central bin for baseline 1
—V2 CEN BAS2	averaged uncalibrated squared visibility in the central bin for baseline 2
—V2 CEN BAS3	averaged uncalibrated squared visibility in the central bin for baseline 3
—CP CEN	averaged uncalibrated closure phase in the central bin
—V2 BIN1 BAS1	averaged uncalibrated squared visibility in bin 1 for baseline 1
—V2 BIN1 BAS2	averaged uncalibrated squared visibility in bin 1 for baseline 2
—V2 BIN1 BAS3	averaged uncalibrated squared visibility in bin 1 for baseline 3
—CP BIN1	averaged uncalibrated closure phase in bin 1
—V2 BIN2 BAS1	averaged uncalibrated squared visibility in bin 2 for baseline 1
—V2 BIN2 BAS2	averaged uncalibrated squared visibility in bin 2 for baseline 2
—V2 BIN2 BAS3	averaged uncalibrated squared visibility in bin 2 for baseline 3
—CP BIN2	averaged uncalibrated closure phase in bin 2
—V2 BIN3 BAS1	averaged uncalibrated squared visibility in bin 3 for baseline 1
—V2 BIN3 BAS2	averaged uncalibrated squared visibility in bin 3 for baseline 2
—V2 BIN3 BAS3	averaged uncalibrated squared visibility in bin 3 for baseline 3

—CP BIN3	averaged uncalibrated closure phase in bin 3
—V2 CEN BAS1 STDEV	corresponding standard deviation with respect to the frames
—V2 CEN BAS2 STDEV	corresponding standard deviation with respect to the frames
—V2 CEN BAS3 STDEV	corresponding standard deviation with respect to the frames
—CP CEN STDEV	corresponding standard deviation with respect to the frames
—V2 BIN1 BAS1 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN1 BAS2 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN1 BAS3 STDEV	corresponding standard deviation with respect to the frames
—CP BIN1 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN2 BAS1 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN2 BAS2 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN2 BAS3 STDEV	corresponding standard deviation with respect to the frames
—CP BIN2 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN3 BAS1 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN3 BAS2 STDEV	corresponding standard deviation with respect to the frames
—V2 BIN3 BAS3 STDEV	corresponding standard deviation with respect to the frames
—CP BIN3 STDEV	corresponding standard deviation with respect to the frames
—SNR BAS1	averaged signal to noise ratio for baseline 1
—SNR BAS2	averaged signal to noise ratio for baseline 2
—SNR BAS3	averaged signal to noise ratio for baseline 3

Table D.1.0: *QC values written by the amber_SciCal recipe. In this table the — has to be replaced by UNCAL, TRF, or CAL if uncalibrated visibilities, transfer functions, or calibrated visibilities are derived.*

D.2 QC parameter of the amber_selector recipe

QC Parameter	Description
FRAMES SELECTED	The number of frames passing the filter
FRAMES PERCENTAGE SELECTED	The percentage of frames passing the filter
X1 THRESHOLD	The threshold applied to baseline 1
X2 THRESHOLD	The threshold applied to baseline 2
X3 THRESHOLD	The threshold applied to baseline 3
LAMBDA CHAN	total number of channels in the file
LAMBDA MIN	lowest waveband in the file [micron]
LAMBDA MAX	highest waveband in the file [micron]
LAMBDA CEN MIN	lowest analyzed waveband in the central bin
LAMBDA CEN MAX	highest analyzed waveband in the central bin
LAMBDA CEN CHAN	total number of analyzed channels in the central bin
LAMBDA BIN1 MIN	lowest analyzed waveband in bin 1 [micron]
LAMBDA BIN1 MAX	highest analyzed waveband in bin 1 [micron]
LAMBDA BIN1 CHAN	total number of analyzed channels in bin 1
LAMBDA BIN2 MIN	lowest analyzed waveband in bin 2 [micron]
LAMBDA BIN2 MAX	highest analyzed waveband in bin 2 [micron]
LAMBDA BIN2 CHAN	total number of analyzed channels in bin 2
LAMBDA BIN3 MIN	lowest analyzed waveband in bin 3 [micron]
LAMBDA BIN3 MAX	highest analyzed waveband in bin 3 [micron]
LAMBDA BIN3 CHAN	total number of analyzed channels in bin 3
UNCALV2 CEN BAS1	averaged uncalibrated squared visibility in the central bin for baseline 1
UNCALV2 CEN BAS2	averaged uncalibrated squared visibility in the central bin for baseline 2
UNCALV2 CEN BAS3	averaged uncalibrated squared visibility in the central bin for baseline 3
UNCALCP CEN	averaged uncalibrated closure phase in the central bin
UNCALV2 BIN1 BAS1	averaged uncalibrated squared visibility in bin 1 for baseline 1
UNCALV2 BIN1 BAS2	averaged uncalibrated squared visibility in bin 1 for baseline 2
UNCALV2 BIN1 BAS3	averaged uncalibrated squared visibility in bin 1 for baseline 3
UNCALCP BIN1	averaged uncalibrated closure phase in bin 1
UNCALV2 BIN2 BAS1	averaged uncalibrated squared visibility in bin 2 for baseline 1
UNCALV2 BIN2 BAS2	averaged uncalibrated squared visibility in bin 2 for baseline 2
UNCALV2 BIN2 BAS3	averaged uncalibrated squared visibility in bin 2 for baseline 3
UNCALCP BIN2	averaged uncalibrated closure phase in bin 2
UNCALV2 BIN3 BAS1	averaged uncalibrated squared visibility in bin 3 for baseline 1
UNCALV2 BIN3 BAS2	averaged uncalibrated squared visibility in bin 3 for baseline 2
UNCALV2 BIN3 BAS3	averaged uncalibrated squared visibility in bin 3 for baseline 3
UNCALCP BIN3	averaged uncalibrated closure phase in bin 3
UNCALV2 CEN BAS1 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 CEN BAS2 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 CEN BAS3 STDEV	corresponding standard deviation with respect to the frames
UNCALCP CEN STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN1 BAS1 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN1 BAS2 STDEV	corresponding standard deviation with respect to the frames

UNCALV2 BIN1 BAS3 STDEV	corresponding standard deviation with respect to the frames
UNCALCP BIN1 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN2 BAS1 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN2 BAS2 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN2 BAS3 STDEV	corresponding standard deviation with respect to the frames
UNCALCP BIN2 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN3 BAS1 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN3 BAS2 STDEV	corresponding standard deviation with respect to the frames
UNCALV2 BIN3 BAS3 STDEV	corresponding standard deviation with respect to the frames
UNCALCP BIN3 STDEV	corresponding standard deviation with respect to the frames
UNCALSNR BAS1	averaged signal to noise ratio for baseline 1
UNCALSNR BAS2	averaged signal to noise ratio for baseline 2
UNCALSNR BAS3	averaged signal to noise ratio for baseline 3

Table D.2.0: *QC values written by the amber_selector recipe.*

D.3 QC parameter of the amber_trf recipe

QC Parameter	Description
LAMBDA CHAN	total number of channels in the file
LAMBDA MIN	lowest wavelength in the file [micron]
LAMBDA MAX	highest wavelength in the file [micron]
LAMBDA CEN MIN	lowest analyzed wavelength in the central bin
LAMBDA CEN MAX	highest analyzed wavelength in the central bin
LAMBDA CEN CHAN	total number of analyzed channels in the central bin
LAMBDA BIN1 MIN	lowest analyzed wavelength in bin 1 [micron]
LAMBDA BIN1 MAX	highest analyzed wavelength in bin 1 [micron]
LAMBDA BIN1 CHAN	total number of analyzed channels in bin 1
LAMBDA BIN2 MIN	lowest analyzed wavelength in bin 2 [micron]
LAMBDA BIN2 MAX	highest analyzed wavelength in bin 2 [micron]
LAMBDA BIN2 CHAN	total number of analyzed channels in bin 2
LAMBDA BIN3 MIN	lowest analyzed wavelength in bin 3 [micron]
LAMBDA BIN3 MAX	highest analyzed wavelength in bin 3 [micron]
LAMBDA BIN3 CHAN	total number of analyzed channels in bin 3
TRFV2 CEN BAS1	averaged transfer function in the central bin for baseline 1
TRFV2 CEN BAS2	averaged transfer function in the central bin for baseline 2
TRFV2 CEN BAS3	averaged transfer function in the central bin for baseline 3
TRFCP CEN	averaged transfer function closure phase in the central bin
TRFV2 BIN1 BAS1	averaged transfer function in bin 1 for baseline 1
TRFV2 BIN1 BAS2	averaged transfer function in bin 1 for baseline 2
TRFV2 BIN1 BAS3	averaged transfer function in bin 1 for baseline 3
TRFCP BIN1	averaged transfer function closure phase in bin 1
TRFV2 BIN2 BAS1	averaged transfer function in bin 2 for baseline 1
TRFV2 BIN2 BAS2	averaged transfer function in bin 2 for baseline 2
TRFV2 BIN2 BAS3	averaged transfer function in bin 2 for baseline 3
TRFCP BIN2	averaged transfer function closure phase in bin 2
TRFV2 BIN3 BAS1	averaged transfer function in bin 3 for baseline 1
TRFV2 BIN3 BAS2	averaged transfer function in bin 3 for baseline 2
TRFV2 BIN3 BAS3	averaged transfer function in bin 3 for baseline 3
TRFCP BIN3	averaged transfer function closure phase in bin 3
TRFV2 CEN BAS1 STDEV	corresponding standard deviation with respect to the frames
TRFV2 CEN BAS2 STDEV	corresponding standard deviation with respect to the frames
TRFV2 CEN BAS3 STDEV	corresponding standard deviation with respect to the frames
TRFCP CEN STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN1 BAS1 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN1 BAS2 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN1 BAS3 STDEV	corresponding standard deviation with respect to the frames
TRFCP BIN1 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN2 BAS1 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN2 BAS2 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN2 BAS3 STDEV	corresponding standard deviation with respect to the frames

TRFCP BIN2 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN3 BAS1 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN3 BAS2 STDEV	corresponding standard deviation with respect to the frames
TRFV2 BIN3 BAS3 STDEV	corresponding standard deviation with respect to the frames
TRFCP BIN3 STDEV	corresponding standard deviation with respect to the frames
TRFSNR BAS1	averaged signal to noise ratio for baseline 1
TRFSNR BAS2	averaged signal to noise ratio for baseline 2
TRFSNR BAS3	averaged signal to noise ratio for baseline 3
OBS CAL RA	Observed Calibrator RA in radians
OBS CAL DEC	Observed Calibrator DEC in ra
OBS NAME	Observed Calibrator name
OBS AVR PARANG	Observed Calibrator Average PARANG in degrees
OBS ESOTRUSTED	Observed Calibrator Trusted status
OBS AVR PBL12	Observed Calibrator Average PBL12 in m
OBS AVR PBL13	Observed Calibrator Average PBL13 in m
OBS AVR PBL23	Observed Calibrator Average PBL23 in m
DB NAME	Closest Calibrator in the database
DB DIAM	Calibrator Diameter in marcsec
DB DIAM ERR	Calibrator Diameter error in marcsec
DB DIST	Computed Distance in arcsec
DB MAG JBAND	Calibrator Magnitude in the J-Band
DB MAG HBAND	Calibrator Magnitude in the H-Band
DB MAG KBAND	Calibrator Magnitude in the K-Band
DB FLAG	Calibrator Quality Flag

Table D.3.0: *QC values written by the amber_trf recipe.*

D.4 QC parameter of the amber_calibrate recipe

QC Parameter	Description
LAMBDA CHAN	total number of channels in the file
LAMBDA MIN	lowest wavelength in the file [micron]
LAMBDA MAX	highest wavelength in the file [micron]
LAMBDA CEN MIN	lowest analyzed wavelength in the central bin
LAMBDA CEN MAX	highest analyzed wavelength in the central bin
LAMBDA CEN CHAN	total number of analyzed channels in the central bin
LAMBDA BIN1 MIN	lowest analyzed wavelength in bin 1 [micron]
LAMBDA BIN1 MAX	highest analyzed wavelength in bin 1 [micron]
LAMBDA BIN1 CHAN	total number of analyzed channels in bin 1
LAMBDA BIN2 MIN	lowest analyzed wavelength in bin 2 [micron]
LAMBDA BIN2 MAX	highest analyzed wavelength in bin 2 [micron]
LAMBDA BIN2 CHAN	total number of analyzed channels in bin 2
LAMBDA BIN3 MIN	lowest analyzed wavelength in bin 3 [micron]
LAMBDA BIN3 MAX	highest analyzed wavelength in bin 3 [micron]
LAMBDA BIN3 CHAN	total number of analyzed channels in bin 3
CALV2 CEN BAS1	averaged calibrated visibility squared in the central bin for baseline 1
CALV2 CEN BAS2	averaged calibrated visibility squared in the central bin for baseline 2
CALV2 CEN BAS3	averaged calibrated visibility squared in the central bin for baseline 3
CALCP CEN	averaged calibrated closure phase in the central bin
CALV2 BIN1 BAS1	averaged calibrated visibility squared in bin 1 for baseline 1
CALV2 BIN1 BAS2	averaged calibrated visibility squared in bin 1 for baseline 2
CALV2 BIN1 BAS3	averaged calibrated visibility squared in bin 1 for baseline 3
CALCP BIN1	averaged calibrated closure phase in bin 1
CALV2 BIN2 BAS1	averaged calibrated visibility squared in bin 2 for baseline 1
CALV2 BIN2 BAS2	averaged calibrated visibility squared in bin 2 for baseline 2
CALV2 BIN2 BAS3	averaged calibrated visibility squared in bin 2 for baseline 3
CALCP BIN2	averaged calibrated closure phase in bin 2
CALV2 BIN3 BAS1	averaged calibrated visibility squared in bin 3 for baseline 1
CALV2 BIN3 BAS2	averaged calibrated visibility squared in bin 3 for baseline 2
CALV2 BIN3 BAS3	averaged calibrated visibility squared in bin 3 for baseline 3
CALCP BIN3	averaged calibrated visibility squared closure phase in bin 3
CALV2 CEN BAS1 STDEV	corresponding standard deviation with respect to the frames
CALV2 CEN BAS2 STDEV	corresponding standard deviation with respect to the frames
CALV2 CEN BAS3 STDEV	corresponding standard deviation with respect to the frames
CALCP CEN STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN1 BAS1 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN1 BAS2 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN1 BAS3 STDEV	corresponding standard deviation with respect to the frames
CALCP BIN1 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN2 BAS1 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN2 BAS2 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN2 BAS3 STDEV	corresponding standard deviation with respect to the frames

CALCP BIN2 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN3 BAS1 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN3 BAS2 STDEV	corresponding standard deviation with respect to the frames
CALV2 BIN3 BAS3 STDEV	corresponding standard deviation with respect to the frames
CALCP BIN3 STDEV	corresponding standard deviation with respect to the frames
CALSNR BAS1	averaged signal to noise ratio for baseline 1
CALSNR BAS2	averaged signal to noise ratio for baseline 2
CALSNR BAS3	averaged signal to noise ratio for baseline 3

Table D.4.0: *QC values written by the amber_calibrate recipe.*

E Troubleshooting Guide