esa SPACE TELESCOPE
European Coordinating Facility

Algorithms for Extracting Spectra from NICMOS Grism Images.

User Manual
Revision 3.1

Revision history:

Revision 3.1 : Wolfram Freudling, May 99
Revision 3.0 : Wolfram Freudling, April 99
Revision 2.1 : Norbert Pirzkal, Wolfram Freudling February 98
Revision 2.0 : Norbert Pirzkal, Wolfram Freudling February 98
Revision 1.1 : Wolfram Freudling, Norbert Pirzkal August 97
Revision 1.0 : Wolfram Freudling, Robert Thomas, Lin Yan April 97
Revision 0.0 : Wolfram Freudling March 96

An online version of this manual is available at
http://ecf.hq.eso.org/nicmos/calnicc/manual/index.html

Contents

What is Calnic C?

- Calnic C is the non-interactive, stand-alone counterpart to the NICMOSlook grism extraction program. It automatically extracts spectra from NICMOS slitless spectroscopy images in a pipeline manner.
- Calnic C uses the SeXtractor program to automatically locate and determine the size of objects in a direct image and extract the corresponding spectra from a grism/spectrum image of the same part of the sky.

Why use Calnic C?

- Calnic C can be used to extract spectra from all kind of slitless spectroscopy images. However, it is particular useful for NICMOS grism images since it takes advantage of NICMOS specific FITS header keywords, uses the Data Quality and Error arrays that are present in NICMOS images processed with the STScI NICMOS pipeline, and uses the ST-ECF NICMOS grism calibration data base to apply proper weights and flatfields and calibrate the wavelength and flux scale of the spectra.
- Calnic C stores the extracted spectra in a Calnic C/NICMOSlook FITS table, which can be read, analysed, and manipulated with the NICMOS specific tool NICMOSlook.
- Calnic C can conveniently be used when the number of spectra to be extracted is too large to extract them manually using NICMOSlook. Calnic C was designed to automatically extract a large quantity of spectra without human interaction.

How to use this Manual

- Review the descriptions of the algorithms used by Calnic C **and NICMOSlook** which are *fully* described in the Algorithms chapter (p. 15).
- Read and follow the Installation chapter first (p. 9).
- Follow sections 2.6.1 (p. 12) and 2.6.2 (p. 12) to setup and run Calnic C.
- Read sections 2.6.4 (p. 13) to learn about the various outputs of Calnic C.

Chapter 1

Introduction

A unique capability of NICMOS is the grism mode, which permits slitless spectra at low resolution. Typically, a direct image is taken in conjunction with grism images for the wavelength calibration. Two IDL programs to extract spectra from such data have been developed at the Space Telescope – European Coordinating Facility[1]. One of it, Calnic C, produces spectra from all objects its detects in an non-interactive way. The other one, NICMOSlook[2], is a fully interactive widget-driven program. Both programs share the algorithms for the basic spectrum extraction. In addition, Calnic C has as an additional feature the built-in capability to use a sophisticated object detection program. On the other hand, NICMOSlook has additional capabilities to correct the spectra for effects which require interactive intervention. This manual describes the algorithms used by both programs. In addition, it gives the details on how to run Calnic C. A separate manual[3] is available to describe the non-interactive version of the program.

The capabilities of Calnic C are the following. Objects are identified on a direct image and classified as stars or galaxies using a neural network approach implemented as the SExtractor program (Bertin & Arnouts, 1996, A & A Suppl. 117, 393). The position of the object are used to extract spectra from a grism image of the same region. The wavelength calibration of the extracted spectra are performed using the position of the objects as determined by the SExtractor program as the zero point and parameterized dispersion relations. After extraction of the spectra, they they are corrected for the wavelength dependence of the quantum efficiency of the detector. The flux scale is then computed using the standard NICMOS flux calibration data. The extracted spectra are corrected for contamination from nearby objects. Subsequently, the extracted spectra are automatically searched for emission and absorption lines. In addition, the continuum emission is automatically determined. The final data products are binary FITS tables with the spectra, error estimates, object parameters derived from the direct imaging and details of the spectrum extraction process as well as postscript files with plots of the individual spectra.

[1] http://ecf.hq.eso.org
[2] http://ecf.hq.eso.org/nicmos/nicmoslook/
[3] http://ecf.hq.eso.org/nicmos/nicmoslook/manual/nicmoslook.html

Chapter 2

Installing Calnic C

Calnic C can be run by users either after the imaging data have been obtained, or at the time when grism images are requested and recalibrated from the archive. The procedure for requesting spectral extraction from the HST archive is described in chapter 6. For the stand-alone version of Calnic C a running version of IDL able to call external functions is necessary.

2.1 Downloading the software

The source code for Calnic C is available on the World Wide Web as a GNU ZIP compressed tar file from http://ecf.hq.eso.org/nicmos/calnicc . After bringing up your favorite Web browser, log on to http://ecf.hq.eso.org/nicmos/calnicc, click on the links to download the latest version of Calnic C or the documentation.

2.2 Installation Procedure

The following instructions assume that a compressed tar file (`calnicc_`*version*`.tar.gz` where *version* is the version number) has been downloaded to a computer with a UNIX operating system running IDL. Specific UNIX commands are given for the C-shell.

Once the compressed tar file is downloaded to a local disk, the installation is as follows:

1. **Create a new directory in which to install the program.**
   Calnic C can be installed in the directory of your choice. You need only to make sure that you have at least 10.0 MB free space available. For example, if you want to install Calnic C in `/usr/local/calnicc`, create this directory with:

   ```
   mkdir /usr/local/calnicc
   ```

2. **Create a new environment variable called** `CALNICC_BASE`.

This variable should point to the newly created directory in the step above. If you chose `/usr/local/calnicc` as the location for the installation, `CALNICC_BASE` should be set to that ENTIRE path name using:

```
setenv CALNICC_BASE /usr/local/calnicc
```

Since this variable is required to be able to run and recompile Calnic C, it is a good idea to add this statement to your shell's startup script (e.g. ~/.cshrc). WARNING: Always make sure that the CALNICC_BASE variable points to the correct directory, especially if you have several versions of Calnic C installed.

3. **Change to the newly created Calnic C directory and extract the distribution**

   ```
   cd $CALNICC_BASE
   ```

   Remember that the full path name of the downloaded distribution must be supplied when decompressing the distribution file. In our example, and if you have downloaded the Calnic C distribution tar file to the directory `/tmp`, you would extract it with the following command:

   ```
   zcat /tmp/calnicc_version.tar.gz | tar xf -
   ```

   or

   ```
   gunzip -c /tmp/calnicc_version.tar.gz | tar xvf -
   ```

4. **[Optional] Add the directory `CALNICC_BASE/bin` to the `PATH` environment variable**
   This step is highly recommended since it will allow you to start Calnic C from any directory. This step can be done by adding the line to the shell startup script (e.g. ~/.cshrc).

   ```
   setenv PATH $PATH:$CALNICC_BASE/bin
   ```

   or

   ```
   set path = ( $path $CALNICC_BASE/bin )
   ```

IDL Version Note:
Calnic C currently comes pre-compiled to run under IDL 5.0. If you are planning to run Calnic C under IDL 4.0 then you will need to recompile Calnic C. See 2.5 for more information.

Operating System Note:
The Calnic C distribution includes precompiled code for the Solaris operating system. For all other operating systems, the SExtractor Library has to be recompiled. The details are given in section 2.4.

2.3 Environment Variables and Calnic C

Before running Calnic C it is necessary to set up a few environment variables (section 5.3). Additionally, you may also wish to modify the default behavior of Calnic C; refer to section 2.6.1 for more details.

2.4 Rebuilding the shared SExtractor Library

It is possible to remake the shared SExtractor library as long as the 'make', 'ld' and 'cc' (preferably gcc) commands exist on the system. This is not necessary for Solaris users as compiled libraries are supplied with the distribution files.
To remake the library:

1. `cd $CALNICC_BASE/src/sextractor` At the Unix prompt; Make sure `$CALNICC_BASE` is set!
2. `make -f makefile`

This will recompile the necessary source files and place the library in the `$CALNICC_BASE/lib/`*system* subdirectory.

2.5 Rebuilding the IDL Calnic C binary Library

Any change to the IDL code of Calnic C requires a "recompilation" of Calnic C IDL programs in order to remake `calnicc_binary`. This must be done while running IDL. To remake this library:

1. `cd $CALNICC_BASE/src` (Make sure that $CALNICC_BASE is set!)
2. `idl` (at the Unix prompt)
3. `@cnc.pro` (at the idl prompt)

This executes an IDL `.run` command on all the IDL program files and will `save` the results in `$CALNICC_BASE/bin/calnicc_binary`.

2.6 Using Calnic C

2.6.1 Customizing Calnic C

Calnic C is a completely non-interactive program intended for use in a "pipeline". The program is controlled by a number of setup files and environment variables.
The setup files reside in the directory `$CALNICC_BASE/calib` and can be modified with a text editor.

`calnicc.setup` Controls behavior of Calnic C.
Refer to section 5.1.2.

`grismspec.dat` Defines grism characteristics.
Read this file carefully before attempting modifications.

`default.xxxx` Controls behavior of `SExtractor`.
See section 3.3.2.

2.6.2 Running the Program

Before attempting to run Calnic C make sure that all environment variables (section 5.3) and system config files (section 2.6.1) are set to desired values. To run, simply type the following at the UNIX prompt:

`calnicc direct.fits grism.fits`
(where direct.fits and grism.fits are the fits files of the direct and grism images respectively)
or

`calnicc -f im_list` (where `im_list` is an ascii file containing a direct and grism FITS file pair, plus the name of an optional object list, per line. This list should be in the same format as what is produced by the companion program NICMOSlook (See section E)).

The first method demonstrates how to call Calnic C with a single direct/grism FITS file pair. The second method shows how to process a series of direct/grism FITS file pairs that are stored in an input file.
Refer to the following section for a more detailed description of the input Calnic C requires.

2.6.3 Input to Calnic C

Calnic C expects a pair of direct and grism FITS images as input. The location of these files must either be specified by full pathnames (if they don't

reside in the working directory from where Calnic C was called) or in the directory specified by the environment variable, `CNC_DATA`. Alternatively, it is possible to run Calnic C

on a series of direct/grism FITS pairs that are listed in a plain ASCII file, one pair per line.
In the `$CALNICC_BASE/data` directory there are two pairs of sample data files. The file called `niclist` contains the file name pairs (one for each line) of these sample fits files. This is an example of how to construct an input file for running Calnic C with the `-f` option. For example, to extract spectra from all of the provided sample images, type
`calnicc -f $CALNICC_BASE/data/niclist`
Appendix G contains example output of one of these image pairs.

2.6.4 Output of Calnic C

Spectra output files from Calnic C are written into either the current working directory (`default`) or into the directory defined by the environment variable, `CNC_SPECOUT`. This output consists of postscript files containing a graphical representation of each object's spectrum and a single FITS file with a FITS table extension containing the information on the spectra of each object detected on the
images.
In the following list, `IMAGE` is derived from the direct input file name.

`IMAGE_SPC.fits` :
Contains one binary table extension for the spectrum information for each object detected on `IMAGE`.

`IMAGE_`*n*`.ps`
Postscript file for each spectrum on `IMAGE`, where *'n'* is the object number (starting with 0) as found on `IMAGE`.

`IMAGE_direct.ps` :
Contains an image of the direct image in which the detected objects are fully marked and labeled.

`IMAGE_grism.ps` :
Contains an image of the grism image in which the detected objects are fully marked and labeled, and the spectra are outlined with boxes.

`DEBUG` :
When a debugging level greater than 0 is set (see section 5.5) the debug information is written to this file

Catalog and log output files are written into either the current working directory (`default`) or into the directory defined by the environment variable, `CNC_CAT`. The following is a list of the output files:

`calnicc.cat` : The calnicc Catalog (Human readable) `calnicc_archive.cat` : The calnicc Catalog (For archive database)

`calnicc.log` : The calnicc Log file

See Section 3.7 for details on these file.

2.6.5 Viewing the Postscript Output Spectra

Use your favorite Postscript viewer to examine the output spectra files. For example, if your direct image is called `ov3_dir.fits`, the spectrum of the first object found will be plotted in file *ROOT*`_0.ps` which will be located in directory `$CNC_SPECOUT` (where *ROOT* is taken from the *ROOTNAME* keyword from the FITS file header, or whatever keyword isa actually declared to point to the target name in the `$CALNIC_BASE/calib/fitskeywds.dat` file).
To view this spectrum e.g. with `ghostview`, type:
`ghostview $CNC_SPECOUT/ov2_0.ps`

2.6.6 Reading a Spectrum's FITS Table Extension into IDL

The spectra contained in the FITS tables produced by Calnic C can be read, displayed, and analysed using the NICMOSlook program. Please refer to the NICMOSlook documentation.

Chapter 3

Algorithms

3.1 Overview

The purpose of Calnic C is to extract and calibrate spectra of all detected objects in a NICMOS grism image with minimal human intervention. For each object detected on a direct image, a spectrum is extracted from the grism image and a FITS file is created along with appropriate catalogue and log file entries. These output files and logs may then be used for follow-up investigations of objects of interest.

Calnic C was developed primarily using IDL (Interactive Data Language). However the C-program SExtractor (Bertin & Arnouts, 1996) is used for source object detection.

Figure 3.1 shows a list of the processing steps.

3.2 Reading Image files

NICMOS data are stored in the FITS format. Each FITS file contains a main header followed by 5 extensions that contain the actual image data along with error, pixel quality, sample, and integration time information. These files are read into internal IDL data structures using IDL programs designed specifically for FITS files of this format. If there are any problems with reading these files, the program immediately reports a problem and stops.

3.3 Object Detection & Classification on Direct Images

3.3.1 NICMOSlook

The user of NICMOSlook is responsible for selecting, classifying and determining the size of objects suitable for extraction of spectra. For that purpose, there are a number of

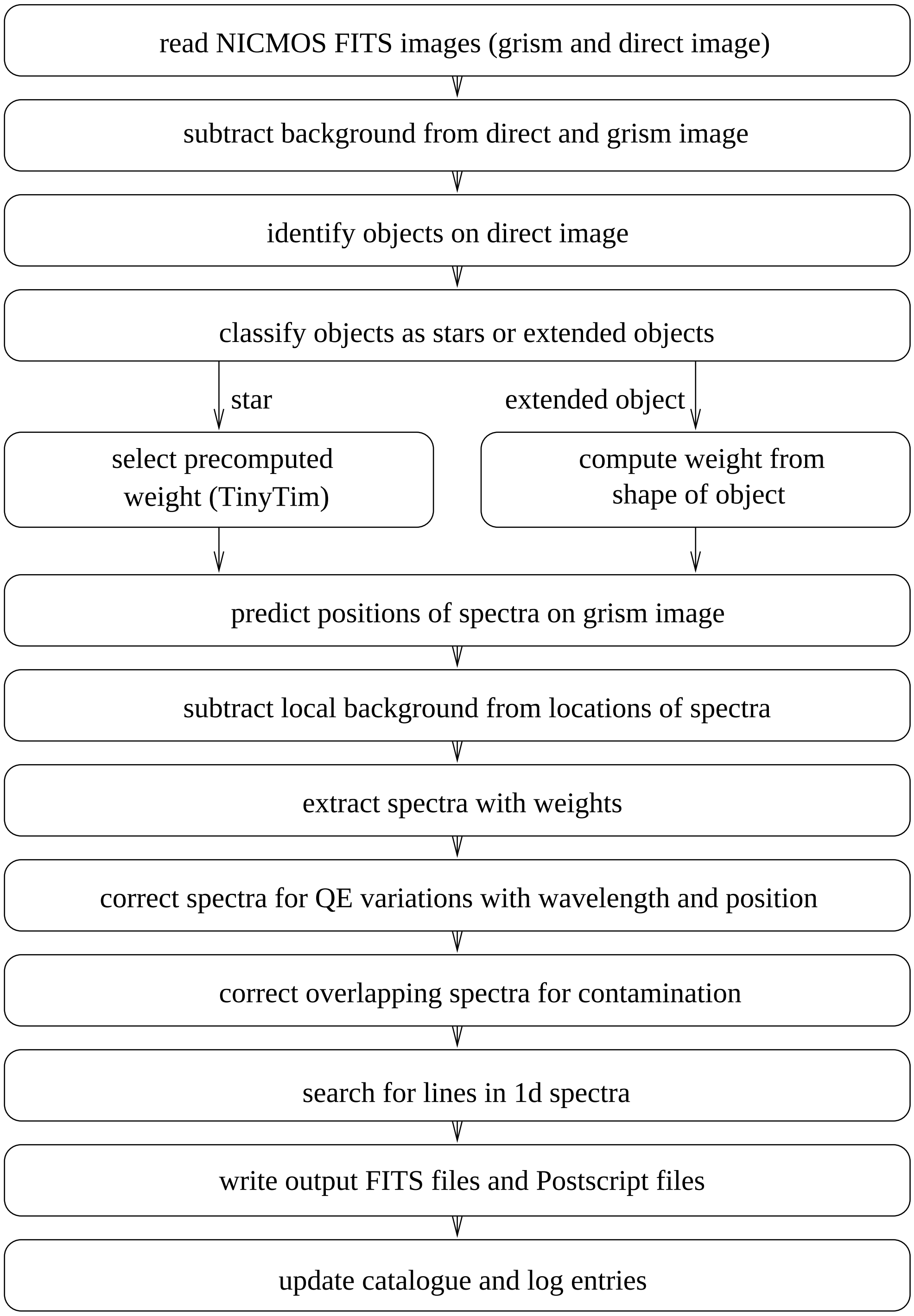


Figure 3.1: Processing steps in Calnic C.

interactive options which are discribed in detail in the NICMOSlook manual[1].

3.3.2 Calnic C

SExtractor

For a non-interactive extraction of spectra, it is essential that objects are found and classified reliably. Therefore, a sophisticated object finding program was included in Calnic C. The third-party program, SExtractor is used to detect object on the direct image. This program is thoroughly documented in the "SExtractor 1.0 User's Guide". See section 5.1.5 for more information about SExtractor and its specific setup files.

Verify Object Significance

The object detection with SExtractor is performed without using the information provided by the data quality flags and the error arrays. This might in some cases lead to the detection of spurious objects. In order to identify such spurious detections, the pixels where the object lies are compared to the quality flag array. If the ratio of bad pixels to the total number of pixels used by the object is above a user-defined threshold, the object is considered as spurious and not considered any further. In addition, the total flux within any object is compared to the quadratic sum of the error estimates for the individual pixels within the object. Again, a user supplied threshold for the significance is used to remove spurious or weak objects.

Grism Image Object Identification

There are several reasons why an object might be easier to detect on the grism image itself than on the direct image. One reason is that the direct image of the object is located outside of the detector, but part of the spectrum is still visible on the grism image. Another, more interesting case is if the most of the flux of an object is contained in a single spectral line. In that case, the usually shorter exposure time for the direct image might not be enough to detect the object, which can be detected on the grism image because of its longer exposure time. To maximize the chances of detecting every object on an image, the grism image is searched for additional object. The grism image is prepared by replacing pixels used by previously detected objects' spectra set to the background level. Subsequently, DAOFIND is used to find objects more than `DAO_THRESH`$\times\sigma$ above the background, where σ is the rms of the image after removing the spectra. Since no zero point for the wavelength scale is known for those spectra, the objects' location is simply noted in the log file, but no attempts are made to extract their spectra.

[1] http://ecf.hq.eso.org/nicmos/nicmoslook/manual/nicmoslook.html

3.4 Background Subtraction

3.4.1 Variations in Background

QE variations with position on NICMOS detectors are strong. Because the grism images cannot be flatfielded, this leads to significant structure in the images. Since the QE variations also depend significantly on the wavelength, the expected background in both the grism image and the direct image will depend on the spectrum of the background in space. The best way to properly account for the background at the time the data were taken is to use a chopping mode so that the target occupies a different part of the detector each time it is observed. Background estimates can then be constructed for any part of the detector and subtracted from the grism/spectrum image of interest. In cases where only one single pointing is available (e.g. for parallel observations), two methods methods of background subtraction are available to the user.

The first one is to use only one grism image at the time, and to interpolate from the regions of blank sky surrounding a spectrum of interest. The second option is to use a pre-defined background and to scale it appropriately to the local background level. The former method, which is described in section 3.4.2, has the advantage that it is more robust and is free of assumptions. The later method has the advantage that it might provide a better estimate of the variations of the background at the exact position of the spectrum (see section 3.4.3).

3.4.2 Interpolated Background Subtraction

The safest option is to remove the background by interpolation over the position of the spectrum. This option can be switched on or off by setting the option NOINTERPOL in the `calnicc.setup` file. When enabled (and only if the Sky Subtraction option is also enabled), the sky background will be estimated by interpolating the background regions surrounding the spectrum. This option should be used whenever the pre-determined background estimates mentioned above do not properly account for the nature of the background observed in your grism/spectrum image. It is recommended that this option be used by default.

The algorithm used during the interpolation process is as follow: First, a region of blank sky `BG_PIXELS` wide is identified around the spectrum, using the location, size and orientation of the object on the direct image, and from the known dispersion relation of the grism (In this computation, the shape of the object is approximated by an ellipse with minor and major axes that are `SZ_FACTOR` times the 2^{nd} moments of the object). Then, this region is searched for the presence of other spectra. If a pixel in that region contains information from another spectrum, it is replaced by the closest unoccupied pixel. In the end, the region that is used to estimate the background is made up of only the unoccupied pixels that surround the spectrum of interest. Figure 3.2 shows an example of how Calnic C chose this region in a complex situation. Finally, a straight line is fit to the flux values of each of the columns of this region of blank sky. This fit is then used to compute and subtract the background at each of the pixel positions of the spectrum.

3.4.3 Pre-Defined Background Subtraction

While the pre-launch version of Calnic C used a pre-computed model of the background in space, it has since been replaced by empirical estimates for each grism. These new estimates were computed by medianing several sparse images together. Calnic C and NICMOSlook scale these pre-defined backgrounds using a scaling factor that is calculated by taking the mean flux values of a region surrounding each spectrum, and dividing it by the mean of the background image in the same region. The scaled background estimate is then subtracted from the grism image prior to extracting each spectrum.

The region used to scale the background is the same one that is used by the Interpolated Background Subtraction method described in section 3.4.2. Figure 3.2 shows an example of how Calnic C chose the region in a complex situation.

The uncertainty in the background estimate is computed from the error plane of the grism image. The uncertainty is the square root of the sum of the errors divided by the square root of the number of pixels.

Pre-launch model of Background

Three different thermal sources contribute to the NICMOS near infrared background. One is the thermal radiation from the Hubble Space Telescope (HST) itself, another is from the NICMOS optics, and the last one is the near infrared zodiacal background. The thermal radiation from both HST and the NICMOS optics can be simply characterized by a blackbody with temperatures of 293 Kelvin and 275 Kelvin respectively. The coefficients of the emissivity of HST and the NICMOS can be taken as 0.1 and 0.15 respectively. The efficiencies of thermal transmission are set as 72% for HST and 85% for NICMOS.

There are two different sources contributing to the near infrared zodiacal background. One is from the thermal radiation of the interplanetary dust (ZE), which can be characterized by a blackbody with a temperature of 265 Kelvin and an emissivity of 6.0×10^{-8}. The second background source is the scattered light from the dust (ZS). The quantitative description for this part is uncertain. Thompson (1996, private communication) suggests $\frac{10^{-9}}{\lambda^{1.69}}$ photon/s/cm^2/μ/sr. For the pre-launch version of Calnic C, Thompson's equation was used. At $\lambda > 3\mu$, the zodiacal background from the dust thermal emission (ZE) is much higher than the scattered light (ZS), however, at the shorter wavelength, such as J band, the zodiacal scattered light dominates.

The thermal radiation from HST dominates the near infrared background in K-band. In H-band, all three background sources are important, and in J, the zodiacal light contributes the most to the background calculation.

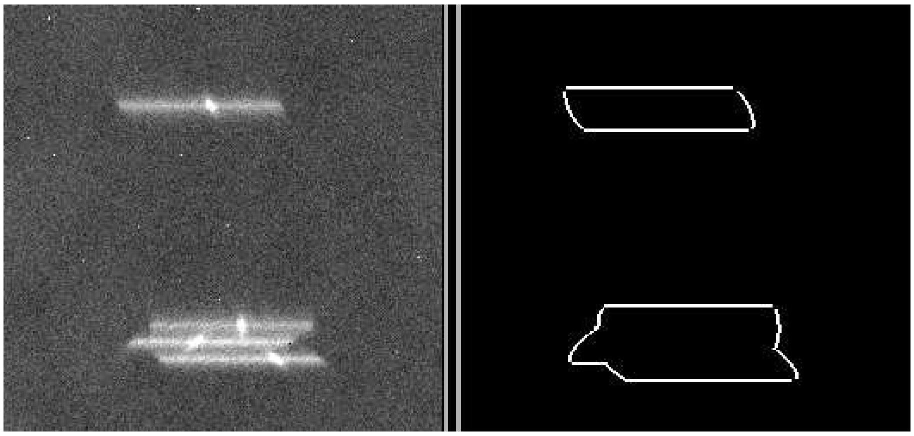


Figure 3.2: Example of grism image and an image showing the pixels used to calculate a background estimate. The highlighted areas on the right panel are the pixels actually used to estimate the background.

Current Background Estimates

The background estimates that are currently included with Calnic C and NICMOSlook were obtained empirically by medianing several sparse images together. These new estimates of the G096 and G141 grisms backgrounds, when scaled appropriately, were observed to be significantly better background estimates than the pre-launch models. However, in the case of the G206 grism, the Interpolated background method better accounts for the relatively large variations of the G206 background.

3.5 Spectra Extraction

3.5.1 Wavelength Calibration and distortions

Description of grisms

Both the dispersion relation and the deviation of the spectra from a straight line are parameterized as third order polynomials. The coefficients of the polynomials and the orientation of the spectra relative to the direction of the rows in the grism images are specified in file `$CALNICC_BASE`/calib/grismspec.dat.

Dispersion Relation

The dispersion relation of each grism is parameterized as

$$\lambda = a0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{3.1}$$

where x is the deflection in pixels relative to the position of the direct image (x_{obj}, y_{obj}) and λ is the corresponding wavelength.

Distortion Model

The deviations of the spectra from a straight line is parameterized as

$$\Delta y = b_0 + b_1 r + b_2 r^2 + b_3 r^3 \tag{3.2}$$

where

$$r = \sqrt{(x - x_{obj})^2 + (y - x_{obj})^2} \tag{3.3}$$

is the distance of a pixel (x,y) from the object coordinates (x_{obj}, y_{obj}) in pixels and Δy is the deviation in pixels of the spectrum from a horizontal line.

In addition, the rotation of the spectrum relative to an exact alignment along a row is specified by a rotation angle. The angle is measured clockwise in units of degrees.

The alignment of the spectrum is taken into account by rotating the grism image around the object position (x_{obj}, y_{obj}) prior to the extraction. The distortions in the spectra are taken into account by introducing a corresponding distortion in the weights used for the extraction.

3.5.2 Flux and Error Bars

Once an objects has been identified on the direct image and the location of its spectrum in the grism/spectrum image has been determined, the spectral extraction can proceed. The spectrum is constructed using

$$F_l = \sum w_{i,l} \cdot g_{i,l} \tag{3.4}$$

where the sum is over pixels with flux $g_{i,l}$ and weight $w_{i,l}$ at a given wavelength l.

Individual error estimates for each pixel are taken from the array 'err' of the input grism image, which should include all sources of uncertainty for the charge in each individual pixel. This errors are propagated to compute the error at each wavelength in the extracted spectrum as

$$\epsilon_{\mathrm{STAT},l} = \sqrt{\sum (w_{i,l} \cdot \epsilon_{i,l})^2}. \tag{3.5}$$

This statitical error is one component of the final error estimates (see section 3.5.7).

3.5.3 Extraction Weights

Spectral extraction is accomplished with the help of a weight matrix to calculate the flux vector for each wavelength. The extraction weights used to extract spectra and compute their associated errors need to be related to the size of the object of interest. We distinguish between two types of objects: point source objects (e.g. stars) and extended objects (e.g. galaxies).

Weights for Point Sources

Calnic C uses pre-computed weights for the extraction of spectra of point sources. These weights were computed from artificial PSFs generated by the PSF generation software TinyTim[2] For the weighted extraction, the weight for a given wavelength l and position perpendicular to the dispersion direction i is given by:

$$w_{i,l} = P_{i,l} / \sum_i P_{i,l}^2 \tag{3.6}$$

where $P_{i,l}$ is the grism image of a point source with a flat spectrum. This is the optimum extraction for point sources when the noise is independent of the flux, i.e. is background dominated.

A FITS file containing the weight matrix for each grism is used for extracting spectra for point sources.

Weights for Extended Objects

Extended objects are treated differently from point sources since the proper weighting of the spectra is not known a priori. The proper weights are determined with the help of the direct image. The basic assumption behind the weighting used by calnic C is that the shape of the object is independent of wavelength.

The weights of extended objects are computed as follows. First, the size and orientation of the objects are computed by multiplying the moments of the direct image with a user defined factor `SZ_FACTOR`. The weights are then computed by summing the pixel values in a given row of the object on the direct image. The extend of the object on the direct image is determined by the ellipse which is defined by the size and orientation of the object (see figure 3.3). Since the orientation of the object is not necessarily along the columns of the image, lines of constant wavelength are not necessarily perpendicular to the dispersion direction. This is later taken into account for the extraction of spectra of extended objects.

3.5.4 Flatfielding of Spectra

When the spectra are extracted, the fluxes have to be corrected for the variations of the quantum efficiency of the detector. These variations depend both on the wavelength and on the position of the object on the detector. Since this is a wavelength dependent effect, the flatfielding can only be done *after* the spectra are extracted (i.e. are wavelength calibrated). In previous releases of \$CALNICC_BASE ,the correction factors were derived through interpolation from a monochromatic flatfield image stored as a single FITS file called nicmosFF.fits (in the `$CALNICC_BASE`/calib directory).

The flat fields can now be specified in two different ways. One is a three-dimensional FITS files, the other one is a list of flatfield files. The file type specified in the grism setup

[2]http://ecf.hq.eso.org/ rhook/nicmos-tinytim.html

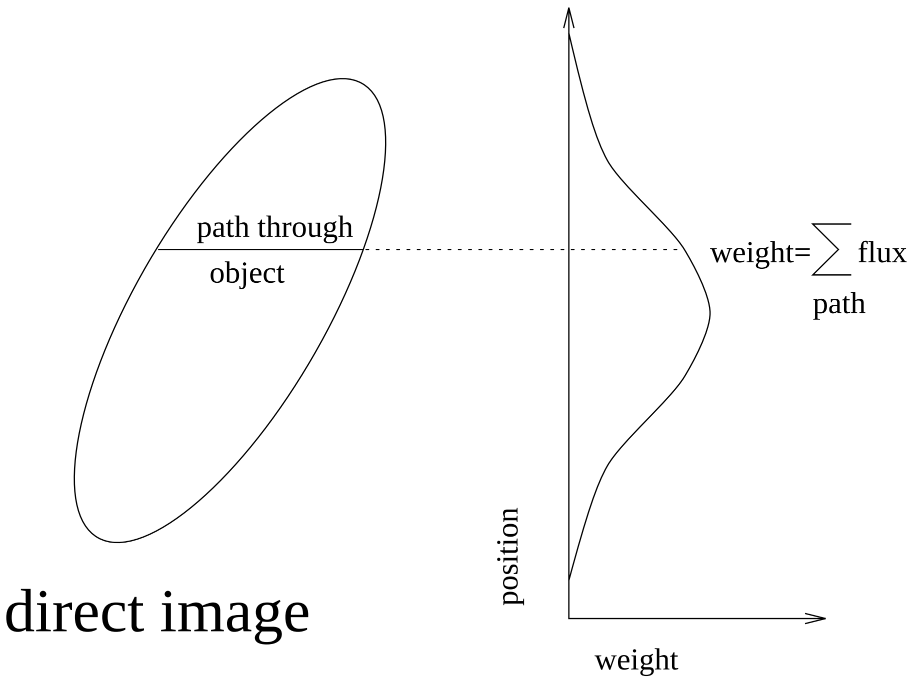


Figure 3.3: How weights are determined for extended objects.

is recognized by the name of the file. Any file name which contains the string "fits" will be interpreted as FITS file. All other filenames will be interpreted as ASCII list.

The three dimensional FITS file has been used in the pre-flight calibration data base and the option remains enabled for backwards compatibility only. *It is not recommended to use this feature.*

The ASCII input file list is assumed to be located in the `$CALNICC_BASE/calib` directory. It should contain at least 2 rows with 2 columns each separated by a single space. The first column should contain the file name of a NICMOS flatfield. The second column should contain the effective wavelength of this flatfield. The NICMOS flatfield are assumed to contain the data in the FITS extension 1, as those in the STScI generated nref directory[3] do. The flatfield names are assumed to be relative to the `$CALNICC_BASE/calib` directory.

$CALNICC_BASE is distributed with one full set of NICMOS flatfields, which are located in `$CALNICC_BASE/calib/nref` . However, the user should find the most appropriate flatfields for a given data set and generate an appropriate list.

[3]http://ecf.hq.eso.org/nicmos/nref

3.5.5 Flux Calibration

Method

Once a spectrum has been extracted, it is flux calibrated by dividing it by the grism response function pointed to in the `$CALNICC_BASE/calib/grismspec.dat` file. The calibrated spectrum is computed as

$$F_{\mathrm{CAL},l} = F_l / R_l \tag{3.7}$$

where R_l is the response for the lth wavelength bin.

This flux calibration introduces additional uncertainties in the extracted spectra. An error $\epsilon_{\mathrm{Rc},l}$ is attached to each flux calibration value R_l. In addition, the uncertainty in the zero point of the wavelength calibration due to errors in the measurements of the object x coordinate leads to an error in the applied response at a particular wavelength. This uncertainty is estimated as

$$\epsilon_{\mathrm{Rx},l} = \frac{\partial R_l}{\partial x} \cdot \mathtt{POS\_ERR} \tag{3.8}$$

where `POS_ERR` is the uncertainty of the zero point of wavelength calibration in pixels, i.e. the uncertainty of the x position of the object on the direct image. `POS_ERR` is a user supplied parameter. Realistic values are on the order of 0.5 pixels. The total error in the flux calibration is

$$\epsilon_{\mathrm{R},l} = \sqrt{\epsilon^2_{\mathrm{Rx},l} + \epsilon^2_{\mathrm{Rc},l}} \tag{3.9}$$

This error has to be taken into account in the estimate of the total uncertainty of an extracted spectrum (see section 3.5.7).

Response Files

The response function is specified in grism response files, which are simple three columns ASCII files containing the wavelength (in μ), the inverse sensitivity (in DN/s per mJy), and the error estimate for the inverse sensitivity (again in DN/s per mJy). These files are derived from calibration data and are regularly updated on the ST-ECF NICMOS homepage[4]. The user can change which response file is used to calibrate extracted spectra by simply editing the `grismspec.dat` file in `$CALNICC_BASE/calib` directory. Please note that the response files are computed using a particular set of flatfields and should therefore be applied to spectra extracted using the same set of flatfields.

Calnic C and NICMOSlook can additionally apply the appropriate units conversion to convert from mJy to $10^{-15} erg/s/cm^2/Angstrom$. Which units are used in the actually output is determined by the `PUNITSW` parameter in the `$CALNICC_BASE/calib/calnicc.setup` file.

[4] http://ecf.hq.eso.org/nicmos/nic3_calib

3.5.6 Deblending Overlapping Objects

On a grism image where many objects are detected, it is likely that the spectrum of one object may overlap with the spectra of others. In these cases, attempts can be made to "deblend" the spectra; i.e. remove or minimize contamination of one spectrum from another.

First, all spectra that overlap are identified and categorized based on the closeness and size of the objects according to three different (user definable) thresholds. The results of this categorization determines whether it is possible (or worth while) to attempt to deblend the spectra.

When it is impossible to remove spectral contamination, the wavelengths affected are noted in the catalogue entry and on the hardcopy output. If there is only a slight contamination, a warning is issued. These two features will be implemented in future development.

In cases where it is possible to deblend a particular spectrum it is necessary to determine which other spectra are affecting it and to calculate the contamination factors at each wavelength. The contamination factors are computed by summing the weights corresponding to the portion of overlap. For example, if object J is contaminating object I's spectrum, the contamination of I is merely the sum of object J's weight in the zone of overlap. This sum is in the y-direction at each wavelength affected. This process of summing the weights is performed for each and every spectrum affecting another object's spectrum.

Once all contamination factors are calculated they are assembled into a square matrix where the i-th column in row j contains the contamination of spectrum i from spectrum j. One matrix is created for each wavelength in the spectrum. In order to correct the spectra, we simply perform a matrix inversion on each matrix. The elements of these inverted matrices are used to calculate a correction coefficient for each flux value of the contaminated spectra.

A possible scenario is that two or more spectra are almost aligned and separating them is quite impossible. This situation shows itself in the form of a singular matrix. In future releases of Calnic C, such cases will be noted in the catalogue and log file and attempts will be made to find an invertible submatrix in order to deblend other objects' spectra.

Identification of overlapping spectra

On the direct image, let the location of an object i be x_i and y_i. On the grism image, the location of corresponding spectrum is from $x_{min,i}$ to $x_{max,i}$ and $y_{min,i}$ to $y_{max,i}$, where x is the direction of the wavelength dispersion.

$x_{min,i}$ and $x_{max,i}$ are computed from the position of the object x_i, and the dispersion length as specified in grismspec.dat.

$y_{min,i}$ and $y_{max,i}$ are computed from the position of the object y_i, and, for point sources from the "width" of the weight and, for extended objects, from the size of the object as determined by SExtractor.

Two objects i and j overlap if they overlap in wavelength in the sense that

$$x_{min,j} < x_{max,i} \quad OR \quad x_{max,j} > x_{min,i} \tag{3.10}$$

and they are close in y

$$|y_i - y_j| < t_i \cdot (y_{max,i} - y_{min,i} + y_{max,j} - y_{min,j}) \tag{3.11}$$

where t_i is one of several user-defined thresholds.

The user can define three thresholds called `o_thresh1` (default: 0.05), `o_thresh2` (default 2) and `o_thresh3` (default: 3).

Three levels of overlap are used:

1. overlap with `o_thresh1`, i.e. spectra almost aligned AND objects have same size (e.g. both point sources). In this case, we cannot correct for it and we only identify the wavelength range of the contamination. This wavelength range will be marked in the output plots in future releases of Calnic C.
2. overlap with `o_thresh2`, but not case 1. This is the most interesting case: here we can correct for it.
3. overlap with `o_thresh3` but not with `o_thresh2`. In this case, there is no (or only very small) contamination, and therefore only a warning "potential contamination through nearby spectrum" is issued.

Below, only case 2 is discussed.

The first step is to identify overlap between any 2 spectra. Note that a spectrum can overlap with several other spectra.

Not every wavelength in the contaminated spectrum is affected. Say, the k-th wavelength point in spectrum for object j is called $F_{j,k}$, and the l-th wavelength point in spectrum for object i is called $F_{i,l}$. Let $x_{max,j}$ and $x_{min,i}$ be the minimum and maximum of the x-coordinates of a spectrum i, and similarly $y_{max,j}$ and $y_{min,i}$ be the minimum and maximum of the y-coordinates of a spectrum i (see figure 3.4).

For the case $x_{min,j} < x_{max,i}$, the following points in spectrum j are contaminated: $F_{j,k}$, where $k = 1...(x_{max,i} - x_{min,j} - x_i)$ AND the following points in spectrum i are contaminated: $F_{i,l}$, where $l = (x_{max,i} - x_{min,j} - x_j)...x_{max,j} - x_j$

The case $x_{max,j} > x_{min,i}$ can be written in an equivalent way.

Computation of Contamination

For the regions of overlap, the spectral point $F_{j,k}$ contains the correct flux from object j, but in addition it contains flux from spectral point $F_{i,l}$, where $l = k - x_{max,i} - x_{min,j}$ and vice versa. Let's call $F_{i,l}$ the spectrum as normally extracted, and $F_{i,l}(true)$ the true spectrum of object i at wavelength l, and $c_{ij,l}$ the contamination factor, then:

$$F_{i,l} = F_{i,l}(true) + c_{ij,l} F_{j,k}(true) \tag{3.12}$$

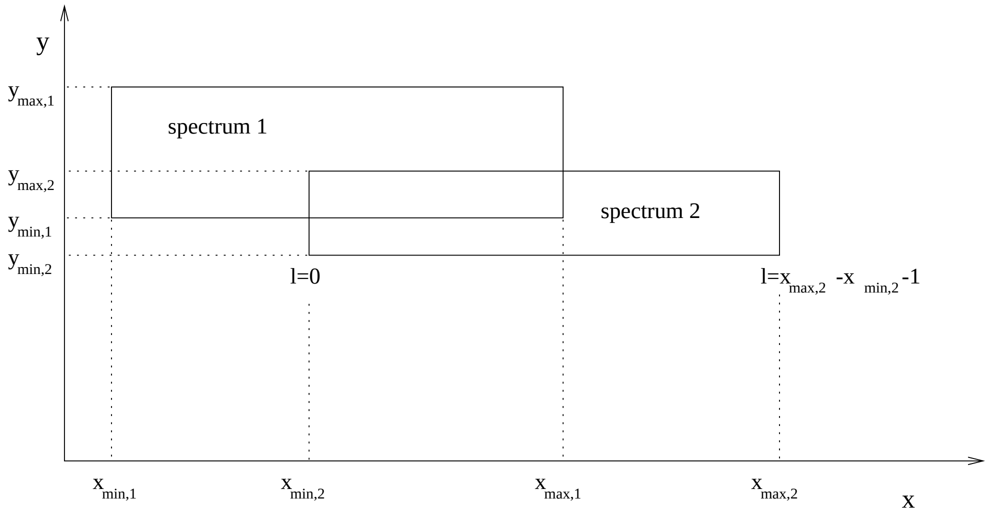


Figure 3.4: Coordinates used to decide overlap of spectra.

were $k = l + x_{max,i} - x_{min,j}$ and

$$F_{j,k} = F_{j,k}(true) + c_{ji,k}F_{i,l}(true) \tag{3.13}$$

were $l = k - x_{max,i} - x_{min,j}$. Now we want to estimate the 2 contamination factors $c_{ji,k}$ and $c_{ij,l}$: Normally the spectra are computed

$$F_{i,l} = \sum_{\text{region of object i}} g_i(y)w_i(y) \tag{3.14}$$

where g is the flux on the grism image and w is the weight. The sum goes perpendicular to the dispersion direction in the region of object i.

Now in order to estimate the contamination from spectrum j, we just add the weights for object j at the position of object i:

$$c_{ij} = \sum w_j(y) \tag{3.15}$$

The sum is along y in the region of object i. Note that the sum of the weights of object j is in region of object i!

and similarly:

$$c_{ji} = \sum_{\text{along y in region of object j}} w_i(y). \tag{3.16}$$

Correction for Contamination

From the contamination factors, the spectra can be corrected. The crucial simplifying assumption is that the spectrum of every object is independent of the position within the

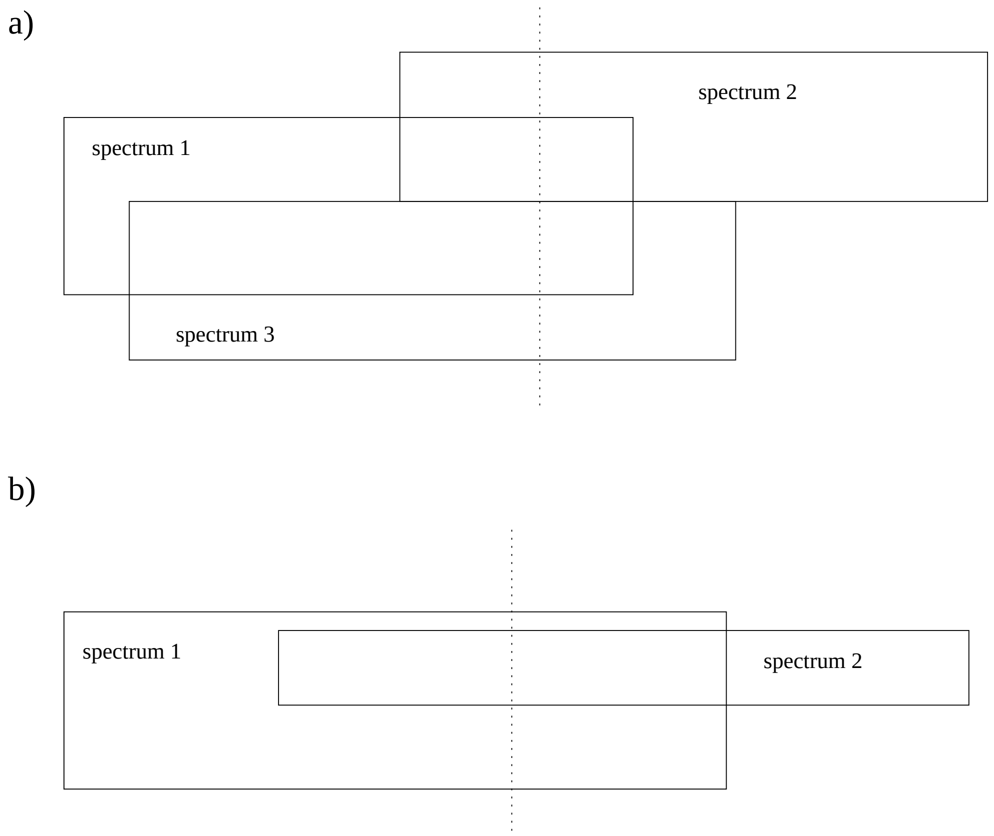


Figure 3.5: Two examples of situations where the spectra can be deblended at the wavelength marked with the dashed lines. a) even though there is no region of spectrum 1 which is not contaminated at the dashed line, both the spectrum 2 and 3 are partially uncontaminated and can therefore be estimated. This estimate in turn can be used to correct spectrum 1. b) Again, spectrum 1 can be estimated from the uncontaminated part, which in turn allows to extract spectrum 2.

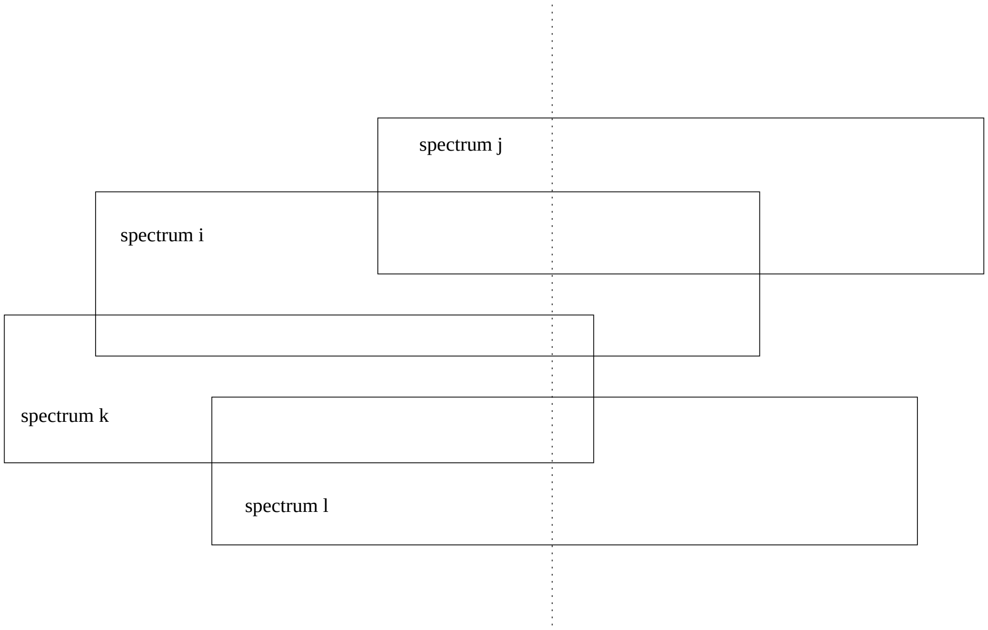


Figure 3.6: Example of a complex situation with the overlap of several spectra.

object. In that case, for a given wavelength in one spectrum, the contamination can simply be written as a set of linear equations,

$$\vec{F}_{meas} = M \cdot \vec{F}_{true} \tag{3.17}$$

where $\vec{F}_{meas}$ is a vector of fluxes for a set of contaminating objects as extracted from the grism image. The ith element of the vector contains the flux from object i at a wavelength which contaminates the wavelength of object j. $\vec{F}_{meas}$ is the equivalent vector as it would be measured if all the spectra were isolated from each other.

The matrix M contains in the i-th column of row j the contamination of spectrum i from spectrum j. If there is no contamination, there would be a zero. This matrix has a "1" in the diagonal (i.e. for i=j), which means just that we assume that the spectrum i contains all the flux from object i, and, in addition, the contamination from other spectra.

Figure 3.5 shows two simple examples which can be treated by this algorithm. In both examples, at least part of one spectrum is not contaminated. Since the underlying assumption is that the contaminated parts of the spectra have the same features as the uncontaminated parts, this information can be used to deduce each spectrum. This is accomplished by solving equation 3.19 by simply inverting the matrix. After computing the inverted matrix I with matrix elements $I_{i,j}$, we can compute the corrected ("true") spectra for each wavelength:

$$\vec{F}_{true} = I \cdot \vec{F}_{meas} \tag{3.18}$$

or

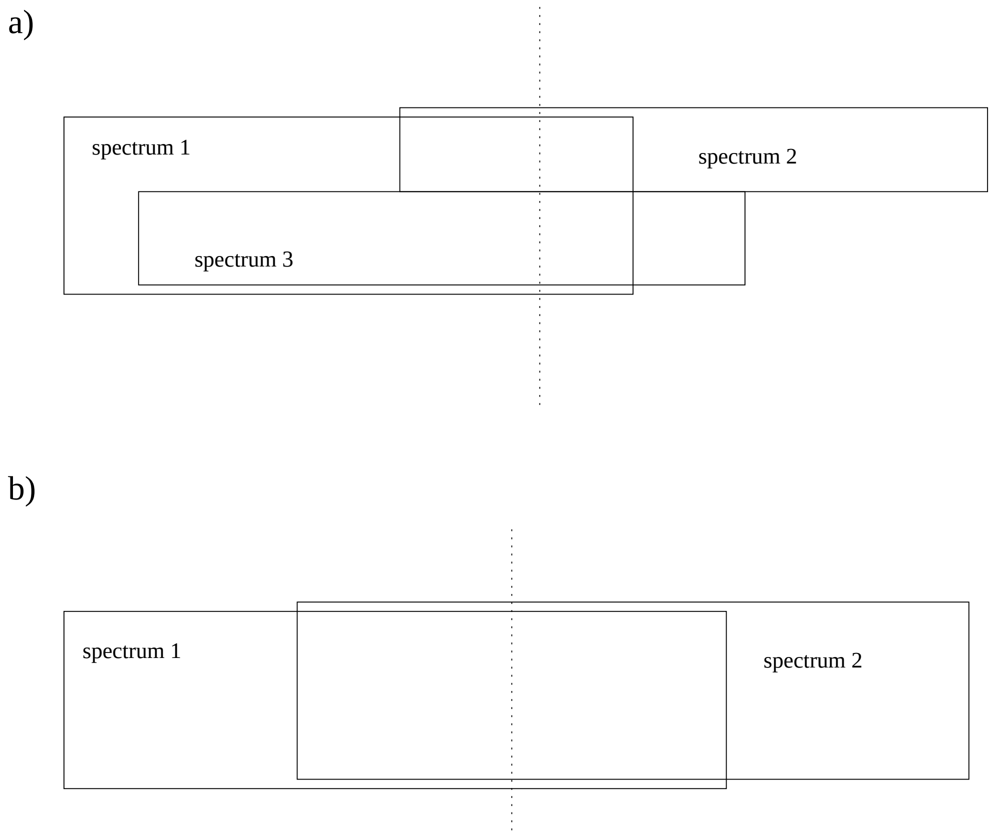


Figure 3.7: Two examples of situations where the spectra cannot be deblended at the wavelength marked with the dashed lines. In both situations, there is no region of spectrum 1 which is not contaminated by other spectra. Therefore, there is no information in that image which could be used to decide to which spectrum a particular feature belongs.

$$F_{true,i} = \sum_{j=1,n_{ov}} (I_{i,j} F_{meas,j}) \tag{3.19}$$

where n_{ov} is the number of overlapping spectra at this wavelength.

Note that in general the situation could be more complex than those shown in figure 3.5. At a given wavelength, spectrum i could be contaminated by spectrum j and k, and spectrum k itself could be contaminated by another spectrum l (in addition to spectrum i, of course). This situation is illustrated in figure 3.6. In this case, n_{ov}=4 and the set of linear equations would be:

$$\begin{pmatrix} F_{meas,i} \\ F_{meas,j} \\ F_{meas,k} \\ F_{meas,l} \end{pmatrix} = \begin{pmatrix} 1 & c_{ij} & c_{ik} & 0 \\ c_{ji} & 1 & 0 & 0 \\ c_{ki} & 0 & 1 & c_{kl} \\ 0 & 0 & c_{lk} & 1 \end{pmatrix} \begin{pmatrix} F_{true,i} \\ F_{true,j} \\ F_{true,k} \\ F_{true,l} \end{pmatrix} \tag{3.20}$$

Note that c_{ji} is not necessarily the same as c_{ij}. The contamination of spectrum i from spectrum j is not necessarily the same as the contamination of spectrum j from i.

Error estimates

The main source of errors in the deblending procedure are uncertainties in the estimated contamination c_{ij} from each contaminating spectrum due to the simplifying assumptions spelled out above. These errors depend on the structure in the contaminating object as a function of wavelength and are hard to quantify. The errors can be big and depend on the flux in the contaminating spectrum. For example, in the case of a faint object being contaminated by a bright object, the subtracted flux might be considerably larger than the remaining flux in the spectrum. As a simple rough estimate of the errors introduced by the deblending procedure, a user supplied fraction `DEBLEND_ERR_MIN` is used, i.e. the error is estimated as

$$\epsilon_{\mathrm{DEB},i} = \sqrt{\sum_{j=1,n_{ov}} (\delta_{i,j} \times \texttt{DEBLEND\_ERR\_MIN} \times I_{i,j} \times F_{meas,j},)^2} \tag{3.21}$$

where the $\delta_{i,j}$ assures that only the flux from the contaminating spectra is considered.

Realistic values of `DEBLEND_ERR_MIN` are on the order of 5 to 20%.

Unsolvable cases

The most common unsolvable situation where 2 objects of same size are aligned (same y) is already treated separately (see above). There could be other cases (such as for two or several extended objects which are almost aligned) in which weights along y-axis happen to add up in such a way that the algorithm to separate the spectra fails. Two examples are given in Figure 3.7. Note that in this situation, there is no information in the image which could be used to decide to which spectrum a particular feature belongs. Therefore, no algorithm can solve this problem.

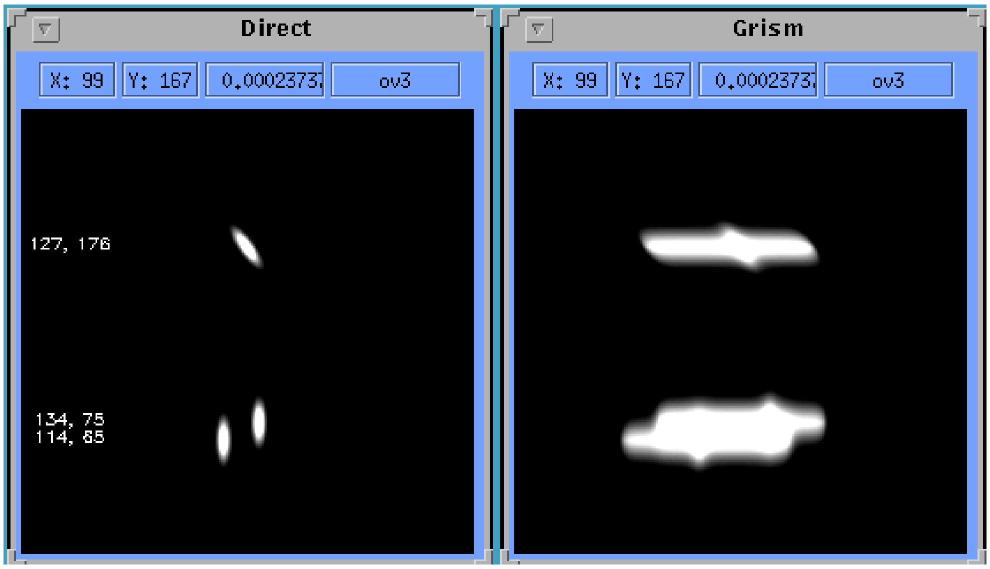


Figure 3.8: Direct and grism image containing overlapping objects.

In the particular algorithm used by the software, this will show up as a non-invertible matrix. Such a case has to be noted. The fact that there is one such situation on a grism image should not prevent other pairs of spectra to be decontaminated. Therefore, one has to search for sub-matrices which can be inverted in such a case.

Example

In figure 3.8, an example of a grism image with overlapping objects is given. The two figures that follow are the extracted spectra of the object "0" (the object in between two objects). Figure 3.9 shows the spectra as it is extracted without first removing the contamination from the nearby objects. Notice how the spectra contains not only the contamination from the nearby objects, but a continuum that is much too high. The second figure (3.10) shows the same object's spectrum after it was extracted using the deblending algorithm. One can plainly see the improvement as the continuum is lower and the peaks from the contaminating objects are removed.

Using Deblending

The deblending algorithms as described here leads to satisfactory results only if the size and orientation of the objects is well determined, which is only the case if the s/n on the direct image is sufficient. If the number of objects detected by Calnic C on an image is large, the CPU time for deblending becomes prohibitive. In most cases the deblending will therefore require some human intervention for good results.

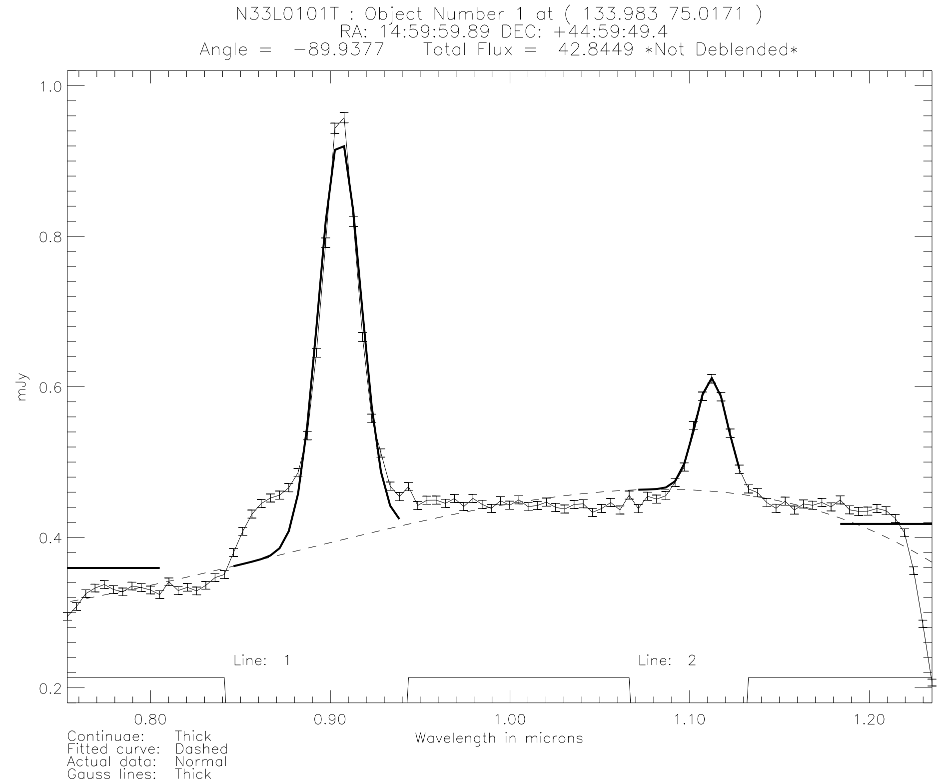


Figure 3.9: Example of an object that has not been deblended.

- Therefore, it is recommended to turn off the deblending option in Calnic C for routine extraction of spectra. It should only be turned on in situations where deblending is clearly needed.
- The sextractor object finding might work well enough for deblending purposes if the objects on the direct image are high s/n and well separated. However, in most cases the user will want to mark the size and orientation of the objects interactively using NICMOSlook, and use the option for importing coordinates files into Calnic C (see 2.6.2).
- For more advice on how to deblend, see the section on Deblending in the Using NICMOSlook chapter of the NICMOSlook manual.

3.5.7 Total Error Estimate

Two kinds of errors estimates are provided by the program. One is the error from the extraction of a single spectrum, and the other one is the uncertainties introduced by the deblending procedure. The plots, FITS and ASCII output files list the deblending error ϵ_{DEB} (see 3.5.6) separately and also compute a total error ϵ_{tot} for each wavelength. The

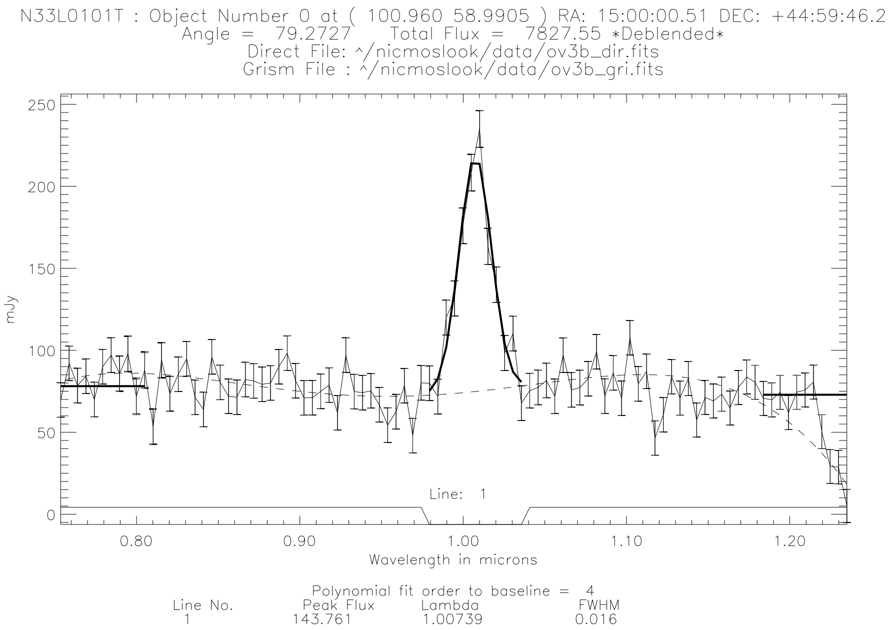


Figure 3.10: Example of the same object in figure 3.9 that has been deblended. Note the additional error bars attached to each point which reflect the uncertainty in the deblending procedure. Only a few points at the very long wavelength side of the spectrum were not contaminated by the neighboring spectrum, and only one error bar is give for those points.

later is computed from the statistical error $\epsilon_{\rm STAT}$ (see section 3.5.2), the calibration error $\epsilon_{\rm R}$ (see section 3.5.5) and the deblending error $\epsilon_{\rm DEB}$ as

$$\epsilon_{\rm tot} = \sqrt{[(\epsilon_{\rm R}/R)^2 + (\epsilon_{\rm STAT}/F)^2] \cdot F_{\rm CAL}^2 + \epsilon_{\rm DEB}^2} \tag{3.22}$$

where as before F is the spectrum before calibration, R is the response function, and $F_{\rm CAL}$is the fully calibrated spectrum. This formulae is the result of the assumption that all three errors are statistically independent.

3.5.8 Correction of Spectra for the Pixel Response Function

Accurate flux calibration of extracted spectra has to take into account the effect of undersampling of the Point Spread Function on the NIC3 detector. Intra-pixel sensitivity variations have a significant effect on the extracted spectra of compact objects. A correction for the effect of the Pixel Response Function (PRF) needs interactive intervention and is therefore only implemented in NICMOSlook.

The Location of Spectra and the Response Function

Spectra on NICMOS grism images are not exactly aligned with the row of the images. Therefore, different parts of a spectrum fall at different y-locations (see figure 3.11). If distortions in the spectra are neglected, the y-position of a spectrum as a function of its x position for a given inclination θ can be written as $y = y_{\rm obj} - y_{\rm offset} + (x - x_{\rm obj}) \sin\theta$, where $x_{\rm obj}$ and $y_{\rm obj}$ are the coordinates of the spectrum on the direct image, and $y_{\rm offset}$ is the offset between the grism and direct image. The x position of the spectrum corresponds to the wavelength of the extracted spectrum. Therefore, a non-zero θ implies that at some wavelength, the spectrum is located between two rows on the detector, while at other wavelengths, the spectrum is centered on a particular row.

The quantum efficiency (QE) of NICMOS depends on the exact position within each pixel where the photon hits the detector. The effect of this intra-pixel response function (PRF)on the photometry of point sources is shown in figure 3.12. A similar impact is expected on the flux calibration of extracted spectra of point sources. The flux in each wavelength bin of a spectrum is similar the aperture magnitude measured in point source photometry. The inclination of the spectrum relative to the rows implies that the quantum efficiency and therefore the flux calibration of spectra is a function of the intra pixel position of the object. This is shown in Figure 3.13, which shows a grism image and extracted spectra of a calibrator source. The spectrum is modulated by a wave-like pattern. The phase of this pattern is determined by the fact that the QE is at its minimum where the spectrum falls between two rows.

The effect on the flux calibration is shown in figures 3.14 and 3.15. The figures show extracted spectra of a calibration source extracted from dithered exposures which lead to different sub-pixel y-coordinates of the star. It can be seen that the amplitude of the wave-like pattern imprinted on the spectra is about 10%, the period is determined by the inclination of the spectrum relative to the rows of the detector, and the phase is determined by the intra-pixel position of the spectrum.

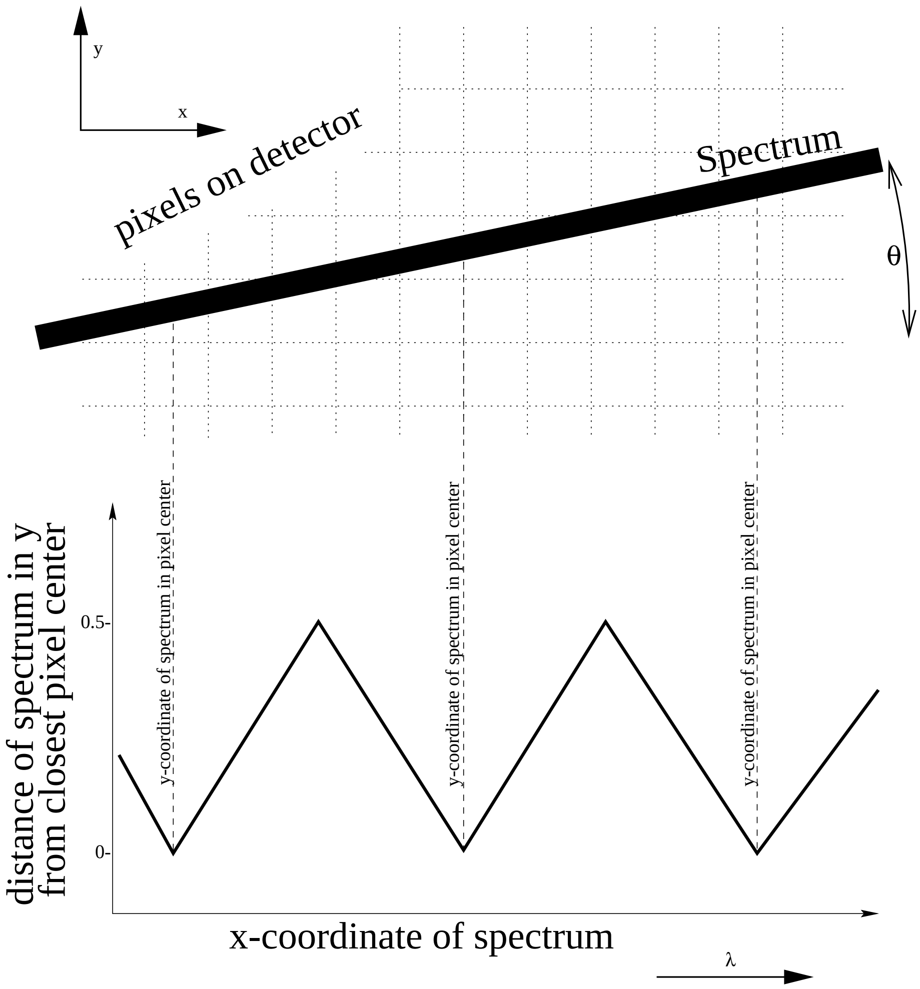


Figure 3.11: *Schematic of the spectrum location as a function of x-coordinate. The spectrum is inclined relative to the rows of the detector by an angle θ. This inclination puts the center of the spectrum at different y-locations as a function of wavelength. The spectrum is at the center of a row at the indicated positions, which correspond to different wavelength wavelengths. A change of θ or and overall shift of the spectrum along the y-axis by a fraction of a pixel will change the wavelength at which the centers of rows are crossed.*

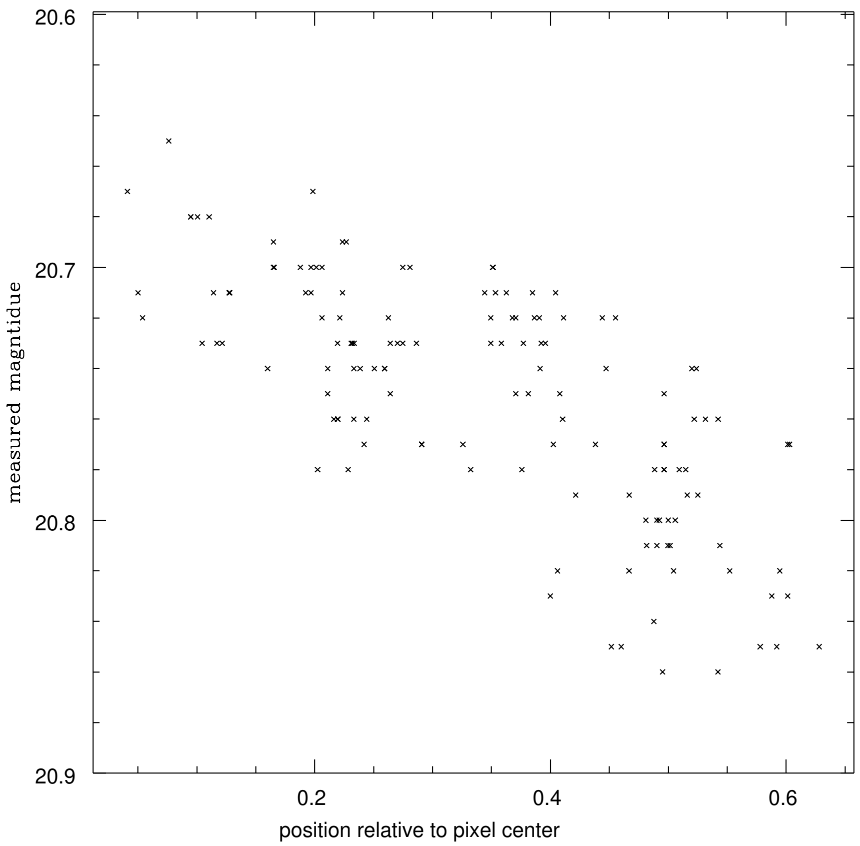


Figure 3.12: *Measured raw magnitudes of a star in the HDF-S field. The magnitude of the same star has been measured on individual exposures, which were taken at different dither positions. The magnitudes are plotted as a function of the measured position of the star relative to the closest pixel center. The data were kindly provided by Richard Hook.*

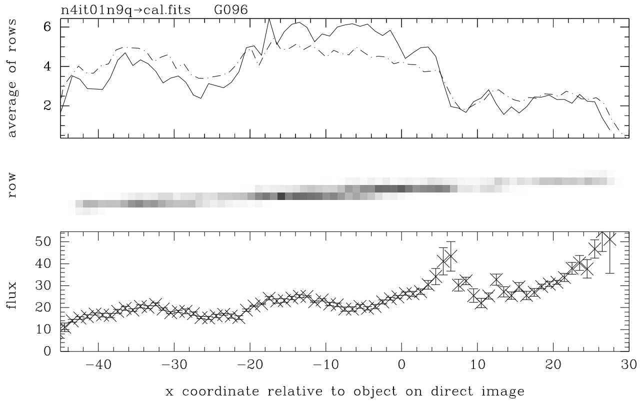
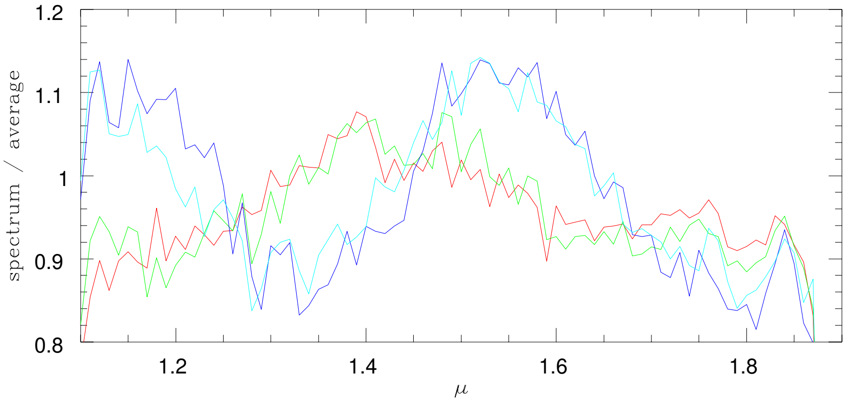


Figure 3.13: *A NICMOS Grism spectrum of the flux calibrator P330e. The middle panel shows a cutout from the NICMOS grism images. Wavelength decreases to the right. Note that at any wavelength, most of the flux is contained in one or two pixels. The regions where the spectrum lies between two rows can easily be identified. At these places, the flux in the spectrum seem to be below the flux in adjacent regions in the spectrum. This visual impression is strengthened by the plot in the upper panel, which shows the average of each column in the cut out as a function of x coordinates relative to the position of the object on the grism image. Finally, the lower panel shows a spectrum extracted by NICMOSlook with the PRF correction option turned off. The x-axes of all three panels are aligned. Comparison of the upper and lower panels show that the flux calibration of NICMOSlook correctly converts the raw fluxes into the expected overall shape of the spectrum. However, the flux is underestimated where the spectrum crosses the boundary between rows.*



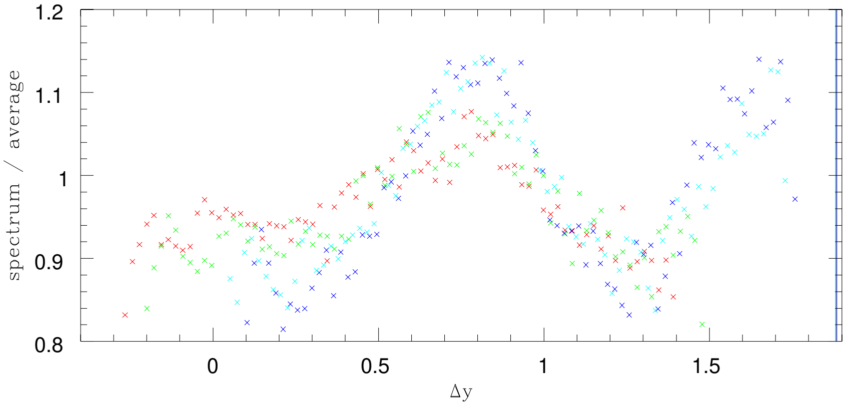


Figure 3.14: *Four spectra of calibrator P330e taken with grism G141. The four spectra were placed at different dither positions, which resulted in different sub-pixel locations of the object. The upper panel shows the spectra as extracted by NICMOSlook, divided by a model of the spectrum of the star. It can be seen that the different extracted spectra deviate in a wave-like pattern with an amplitude of about 10% from the average spectrum. The lower panel shows the same data plotted as a function of* $\Delta y = y_{\text{spectrum}} - y_i$*, where* y *is the row of the spectrum location, and* y_i *is an arbitrary integer number chosen separately for each spectrum. It can be seen that the imprinted pattern of the individual spectra are now shifted and are in phase. This demonstrates that the phase of the pattern is determined by the intra-pixel position of the spectrum.*

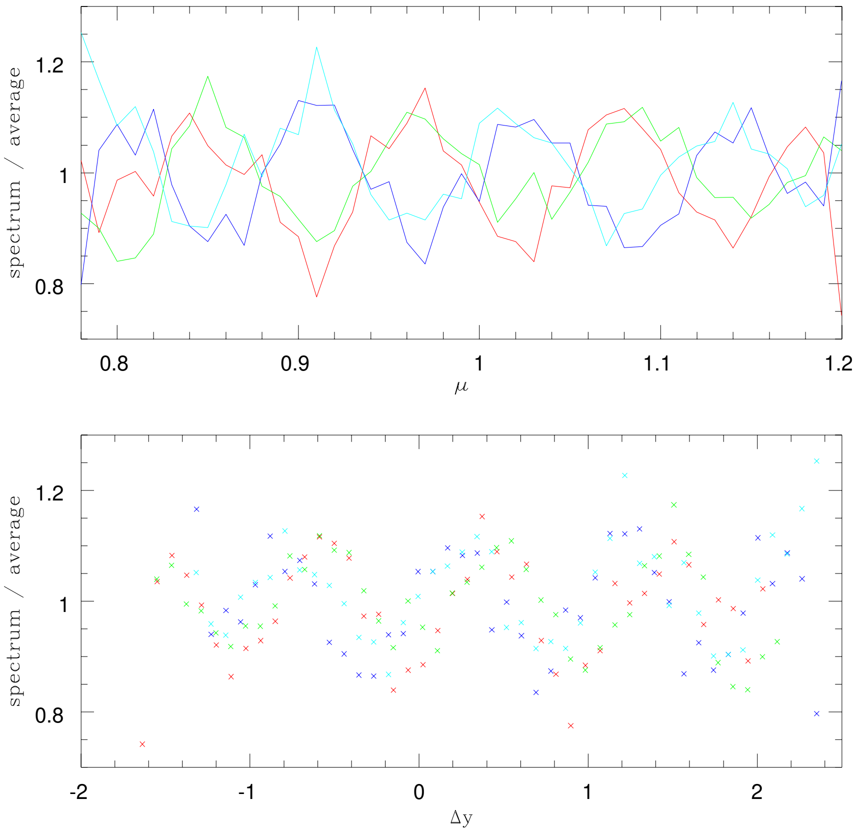


Figure 3.15: *Same plot as figure 3.14, but for grism G096. The inclination θ of G096 is larger than the one of G141. This is the reason for the larger range of Δy covered. In the spectrum, this will show as a shorter period of the PRF pattern in the spectrum.*

Correction of Extracted Spectra

The functional form of the relation between the intra-pixel position and quantum efficiency depends on detector properties, the point spread function and the size of the imaged object. For point sources, the function can be determined empirically. It is easy to formulate the problem in terms of a correction factor f_l to the extracted spectrum, where the true spectrum $F_{l,\mathrm{prf}}$ relates to the raw extracted spectrum F_l as

$$F_{l,\mathrm{prf}} = f_l \cdot F_l \tag{3.23}$$

where l labels the different wavelength bins. For simplicity, the dispersion relation here is assumed to be linear,

$$\lambda_l = a_0 + a_1 x_l \tag{3.24}$$

where x_l is the deflection of the lth wavelength bin in pixels relative to the position of the object on the direct image. A simple model for the correction factor was fitted to the calibration data of the first NICMOS campaign,

$$f_l(\lambda) = 1 + A\cos[p(\lambda - a_0) + \psi] \tag{3.25}$$

The period p is given by

$$p = 2\pi\tan(\theta)/a_1 \tag{3.26}$$

where a_1 is the dispersion in μ/pixel, and the phase ψ by

$$\psi = 2\pi(y_{\mathrm{obj}} + y_{\mathrm{offset}}) \tag{3.27}$$

where y_{obj} is the position of the object on the direct image, and $y_{\mathrm{obj}} + y_{\mathrm{offset}}$ is the y coordinate of the spectrum at the reference wavelength a_0. For high signal-to-noise spectra, θ and $y_{\mathrm{obj}} + y_{\mathrm{offset}}$ can easily be determined. The only free parameter to be directly determined from the extracted calibration spectra is the amplitude A. Figure 3.16 shows the fit to the calibration data taken during the first NICMOS campaign. The amplitudes determined from NICMOS campaigns (when NIC3 was in focus) is about 0.1, whereas outside the campaigns the amplitudes are about 0.05. Significant deviations from these average values have been detected, and it is currently not known whether these changes in amplitude correspond to changes in the PSF or are due to the calibration data taken at different positions on the detectors. Because of these variations in amplitude, we estimate that correction for PRF will on average reduce the amplitude of the wave like pattern in spectra by a factor of 2 to 5.

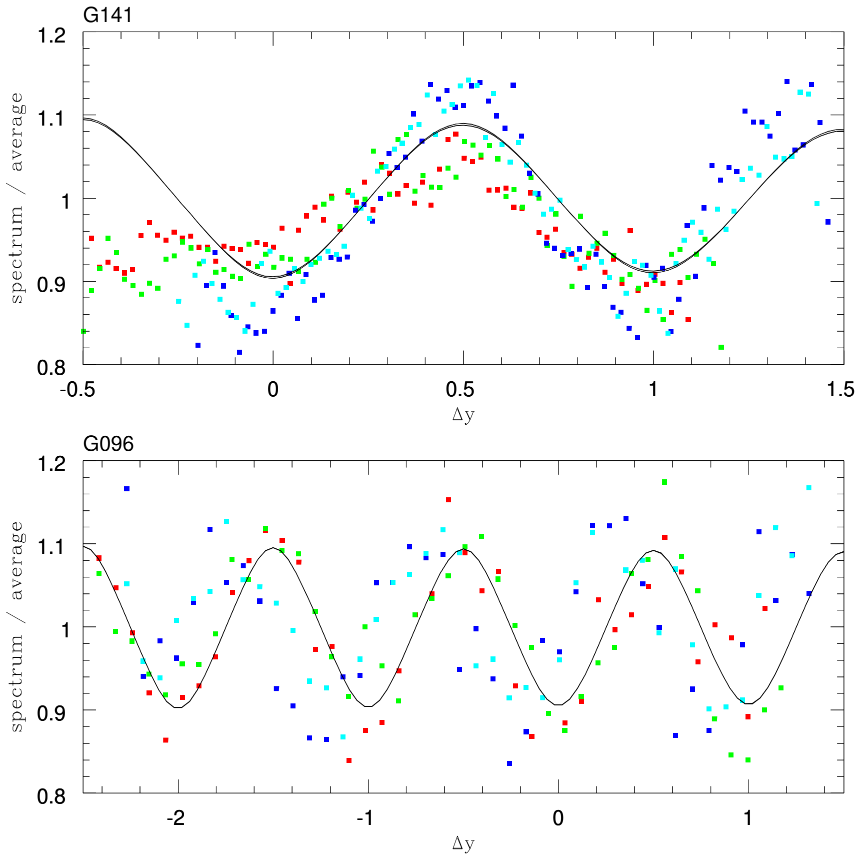


Figure 3.16: *Fit of the PRF correction factor. The data are the same as in figure 3.14 and 3.15. With the amplitude A as the only free parameter, the behavior of both grisms can be fitted with the model. Note in particular that the amplitude seems to be almost independent of wavelength.*

3.6 Characterization of extracted Spectra

3.6.1 Introduction

The purpose of Calnic C to provide the user with a robust extraction of spectra from the grism images. Further analysis of the spectra is up to the user. However, in order to select appropriate spectra for further investigation, it is useful to characterize the spectra with a few simple parameters. For that purpose, Calnic C attempts to identify the continuum emission, find emission and absorption lines and broad spectral features for all extracted spectra. Gaussian fit parameters of up to five lines within the spectrum, the average of the continuum at wavelengths longer and shorter than the wavelength of the lines, and the significance of broad absorption line features are provided in the output catalogue.

3.6.2 Line Searches

Lines in the spectra are identified by fitting a polynomial to the spectrum, and then searching for deviations from the fit. First, a polynomial of order `fit_order1` (default 1) is fit to the whole spectrum. Subsequently, all points from the original spectrum which deviate by more than `DET_THRESH`$\times\sigma$ from the fit are removed from the fit region, where $sigma$ is the rms of the deviations of the spectrum from the fitted polynomial. For each deviant point removed from the fit region, the 2 neighboring points are also removed. A new polynomial is then fitted to the reduced fitting region, and the whole process is iterated until the fitting region is constant or the maximum number of `n_iteration` iterations has been reached. After `n_iterlow` iterations, the order of the polynomial is increased to `fit_order2`.

After this iteration, contiguous regions are identified among the rejected points, i.e. neighboring points which all fit the rejection criterion. Contiguous regions which contain at least `MIN_NPOINTS` points are considered to be spectral.

After subtracting the polynomial from the spectrum, a Gaussian is fitted to each of the line regions separately. The fit is weighted by the error estimate of the spectrum. The parameters σ_i, peak flux P_i and wavelengths λ_i are recorded, where i labels the lines.

Subsequently, a search for overlapping lines is carried out. If any two lines are overlapping in the sense

$$(\lambda_1 - \lambda_2) < \texttt{blend\_factor} \cdot (\sigma_1 - \sigma_2), \tag{3.28}$$

then those 2 lines are merged, i.e. a single Gaussian is fit to the 2 contiguous regions identified before.

The five lines with the highest signal/noise ratio are finally recorded both in the catalogue and in the spectrum header.

In figure 3.17, an example of an output spectrum is shown. Lines are numbered in order of significance (Line 0 being the most significant and Line 4 being the least). Note that the broken horizontal line along the bottom of the graph shows the continuum through which the polynomial was fit. The solid darker lines are the Gaussian fits through the line.

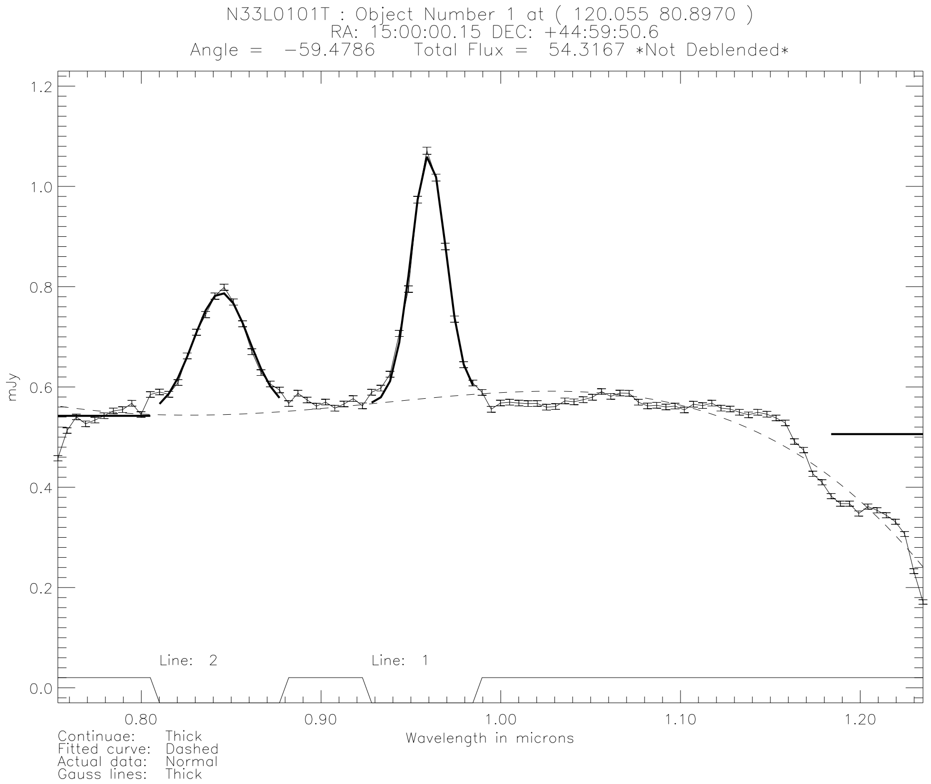


Figure 3.17: Example of the paper product delivered by Calnic C. The extracted spectrum is shown with error bars. Superimposed is the fit of a polynomial to the continuum. The bold lines are the Gaussian fits to the identified lines. The regions of the polynomial fit as well as the lines are indicated at the bottom of the plot. The lines are labeled starting with the most significant line.

3.6.3 Continuum

The average of the continuum emission is computed at wavelengths shorter than the center wavelength of the line with the shortest wavelength λ_s, i.e. cont_short is computed as $\sum F_l$, where the sum is over all wavelength with $\lambda < \lambda_s - \texttt{cont\_factor} \times \sigma_s$.

Similarly, cont_long longwards of the line l with the longest wavelength is computed as $\sum F_l$, where the sum is over all wavelengths with $\lambda > \lambda_l + \texttt{cont\_factor} \times \sigma_l$.

3.6.4 Broad Absorption Features

This functionality is to be implemented with future development.

3.7 Spectral FITS files Format

Once the results of the processing are assembled, they are written to a FITS file. The extensive primary header of this file contains relevant information regarding the observation run (see Appendix B). Following this header are a series of FITS binary table extension headers and associated tables. One table extension for each detected object is written. These tables contain 5 columns which contain the wavelength and flux vectors of the extracted spectra along with vectors for the statistical, deblending, and total errors from the extraction process.

Chapter 4

Catalogue Output

The following shows an example of a human readable catalogue entry.
Object Number 0 from observation file ../data/direct.fits
RA = 0.00000 DEC = 0.00365056
Output FITS file: N33L0101T_0.fits
Object Type Probability (1.0 = star; 0.0 = extended object): 0.0140434
Object size (2nd order moments along axes): 13.8394 7.73015
Object angle along major axis:): 43.0521
Total flux of spectrum: 38295.5
Number of lines: 7
Order of last curve fit: 4

5 Most significant lines:

Flux	Wavelength	FWHM	S/N	CHI2	ILF
957.141	1.15657	0.0403234	147.223	596.2	41.1161
758.077	1.64862	0.0103599	11.1825	314.4	22.704
458.424	1.94372	0.0264251	10.5146	154.2	28.51
534.383	2.05612	0.0505358	16.7858	774.5	33.45
655.240	2.28973	0.1081202	30.5714	594.3	12.17

RMS of last polynomial fit : 71.2457
Continuum (long) : 593.520
Continuum (short): 252.996

Below is an example of a log file entry.

Date Run : Fri Sep 27 10:53:43 1996
Direct image : ./data/direct.fits
Grism image : ./data/spec.fits

Objects Found : 3
Number of extended objects : 2
Number of point Sources : 1
Spectra Extracted : 3
Spectra with no lines : 0
Spectra with 1 lines : 1
Spectra with 2 lines : 2
Spectra with more than 2 lines: 0

Warnings: none
Errors : none

4.0.1 Log file

A log entry containing various statistics is made for each run of Calnic C. Table 4.1 contains a list of these log file entries.

Table 4.1: Parameters in Output Log File

Description	Notes
Date Run	Date of run of Calnic C
Direct image name	
Grism image name	
No. Objects Found	
Number of extended objects	
Number of point Sources	
Spectra Extracted	
Spectra with no lines	
Spectra with 1 lines	
Spectra with 2 lines	
Spectra with more than 2 lines	
Warnings	
Errors	

Chapter 5

Software Dependencies, System Requirements & Debugging

5.1 Calnic C

Calnic C is written as a Bourne Shell script which calls IDL. The IDL routines in turn call some external C programs. Most the software is written in IDL and therefore Calnic C requires a valid IDL license. Calnic C has been tested so far under the Solaris operating environment.

For other operating systems, the SExtractor shared object libraries have to be recompiled.

5.1.1 Directory Structure

The directory structure contains all the necessary calibration files, FITS images, etc. to run the current version of Calnic C:

The directory structure of the Calnic C Software is in figure 5.1

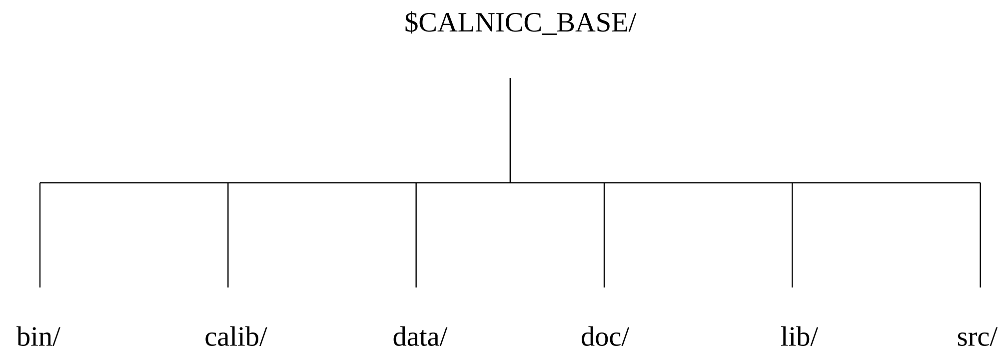


Figure 5.1: Directory Structure of Calnic C Software

Directory Contents:

1. `$CALNICC_BASE`/bin

calnicc :	The program (Bourne Shell script)
calnicc_startup.pro :	IDL startup file
calnicc_binary :	"Compiled" idl code

2. `$CALNICC_BASE`/lib

solaris/libsex.so:	Solaris SExtractor Shared object

3. `$CALNICC_BASE`/calib

bck*grismX*.fits :	Image background for the different grisms
calnicc.setup :	Calnic C default values
fitskeywds.dat :	FITS keywords
default.* :	SExtractor parameter files
grismspec.dat :	Grism specification file
calnicc_tmp.cat :	A temporary by-product of SExtractor
nicmosFF.fits :	Flat Field file (this is a cube)
*.response :	list of filter curves
*.conv :	PSF for each filter

4. `$CALNICC_BASE/data` : Two test image pairs
5. `$CALNICC_BASE`/doc : Documentation directory
6. `$CALNICC_BASE`/src : Source code for Calnic C

idlsrc :	Contains all IDL code used in Calnic C
sextractor :	Contains SExtractor source code

5.1.2 Calnic C Setup Parameters

In the calibration subdirectory mentioned in the section above, the file `calnicc.setup` contains variety of parameters that can be set to modify the behavior of Calnic C. This file can be edited with any text editor, but it is more convenient to modify these parameters by running a special parameter editor either by calling it from NICMOSlook or running it independently. From within NICMOSlook use the button Edit Setup Params button to bring up the editor. From within IDL simply type `nlk_edit_setup_params` at the prompt. All parameters are described in plain English and can be changed (see Figure 5.2). The changes are saved back to `calnicc.setup` after hitting the Save button on the editor pop-up. See table C.2 for the default values of the parameters described below.

Object Finding

`BADPIX_THRESH`
Pixel coordinates that contain object information are compared with a bad pixel map. A ratio of total pixels to the number of the bad pixels is calculated and compared to

this parameter. If the ratio is higher than this value, the object is removed from the object list.

`MAXOBJS`
For computing resource purposes, this number determines how many objects on an image set are to be examined.

`REJ_THRESH`
Currently not used.

Spectrum Extraction

`SZ_FACTOR`
Width of a spectrum in units of the 2nd order moment of the object on the direct image.

`WEIGHTPERC`
When computing the set of extraction weights for unweighted extraction, the extent of the array is derived from the object's profile. All the weights corresponding to regions that contribute less than WEIGHTPERC percents of the maximum flux are set to zero. The rest of the weights are all set to the same values and scaled to conserve the total extracted flux.

`NOFLATFIELD`
This parameter allows users to turn off (or on) the flatfielding of the extracted spectra.

`NODTQ`
This parameter allows users to turn off (or on) the use of the Data Quality information. When DTQ is used, pixels marked as bad are ignored.

`DTQ_MASK`
Mask for rejection of pixels based on the value of the Data Quality image. The mask is given as an integer value. Its corresponding binary number is used to reject pixel witch have flags set where the binary number of the mask is one. For example, if this parameter is set to 3 (which has a binary code of 11), all pixels are rejected with the first and second flat set.

`ASSOCFF`
This parameter allows users to turn off (or on) the use of a NICMOS "association file" to construct a flatfield for a particular chopping pattern.

`POS_ERR`
This field should contain an estimate of the uncertainty (in pixel) in objects' coordinates. This value is used to derive an error estimate when applying a response function to an extracted spectrum.

`PUNITSW`
The units to use for the spectral plots. Valid choices are dn, mjy, or erg. The grism response function from the file grismspec.dat will be applied if necessary.

`NOWEIGH`
This parameter has only an effect when running Calnic C. It allows a user to turn on (or off) the use of weighted extraction.

`CORRECTPRF`
This parameter allows users to turn off (or on) the use of correction for the effect of the intra-pixel response function.

`ADCGAIN`
Analog to digital gain in electrons/ADU. This value will be used as a default IF no keyword ADCGAIN is found in the image header.

Background

`BG_PIXELS`
Calculate a background estimate for an object during spectral extraction, the mean value of the pixels surrounding the edge of the object is calculated. This parameter determines the width of the region used in this calculation.

`NOBACKGND`
This parameter allows users to turn off (or on) background subtraction before spectral line extraction commences.

`NOINTERPOL`
If this parameter is turned on, then the program will use interpolated values of the surrounding image background when subtracting the background. If this parameter is turned off, then the calibration background file is scaled and subtracted from the image instead.

Line Detection

`CONT_FACTOR`
This factor determines the region for which the average of the continuum "longwards" of the line with the longest wavelength and "shortwards" of the line with the shortest wavelength is computed. The "long" region is the peak of the line with the longest wavelength added to the σ of that line multiplied with this factor. The "short" region is the peak of the line with the shortest wavelength minus the σ of that line multiplied with this factor.

`DET_THRESH`
Threshold used in the spectral line search to identify lines in units of the rms of the spectrum.

`FIT_ORDER1`
A polynomial is fitted to the continuum in each spectrum. After the first fit, deviating points are removed from the fit region and the process is iterated. This parameter is the order of the polynomial for the first `N_ITERLOW` iterations (see below).

`FIT_ORDER2`
Polynomials are iteratively fitted to the continuum in each spectrum. After each fit, deviating points are removed from the fit region for the next iteration. This parameter is the order of the polynomial for all but the first `N_ITERLOW` iterations (see below).

`MAXLINES`
This parameter determines the maximum number of lines which can be detected in any spectrum.

`MIN_NPOINTS`
This parameter is the minimum number of consecutive points which deviate from the continuum fit in the spectrum which are considered as a spectral line.

`MIN_PIX_NO`
Minimum line width for a line to be considered as real.

`BLEND_FACTOR`
In the spectral line search process, 2 lines are considered to be the same line if the separation of the peaks is smaller than the sum of the line widths multiplied with this factor. In that case, the region of the two lines are joined and a new Gaussian is fitted over the entire wavelength range of the line.

`MIN_SIGNSE`
Detected lines whose signal-to-noise ratio is less than this value are rejected.

`N_ITERATIONS`
The fit to the continuum in the spectrum is discontinued after this number of iterations.

`N_ITERLOW`
The fit of a polynomial to the continuum in the spectrum is done in two steps. For the first iterations, a low order polynomial is fitted in order to identify strong lines. After several iterations in which deviant points are rejected and a new polynomial is fit, the order of the fit is increased. This parameter determines the number of iterations before the next higher fit order is used.

Image Registration

`NOWCS`
Parameter to turn off / on the use of the WCS in the image header to register the

images. If WCS registration is turned off, the pixel coordinates in the images are used (i.e. the images are assumed to be perfectly registered).

`FROMWCS`
Parameter to turn off / on the use of ra and dec in input object lists. If this option is turned on, the pixel coordinates will be used instead.

Deblending

`NODEBL`
This parameter has only an effect when running Calnic C. It allows a user to turn on (or off) deblending.

`O_THRESH1`
Close spectra on the grism image potentially contaminate each other. This parameter is the maximum distance of 2 spectra on the grism image for them to be considered being exactly aligned. The parameter is in units of the sum of the line widths of both spectra.

`O_THRESH2`
Close spectra on the grism image potentially contaminate each other. This parameter is the maximum distance of 2 spectra on the grism image for which a contamination is expected and Calnic C attempts to "deblend".

`O_THRESH3`
Close spectra on the grism image potentially contaminate each other. This parameter is the maximum distance of 2 spectra on the grism image for which a contamination is considered to be still possible.

`DEBLEND_ERR_MIN`
When calculating the deblending error, this value is substituted for deblending errors that are of smaller value.

Debugging

`DEBUG_LEVEL`
This parameter determines the verbosity of Calnic C's output messages. The higher its value, the more information is printed to the debugging output file. This parameter allows for detailed debugging messages. Refer to section 5.5 for more information.

`MAXPIX`
For computing resource considerations, the maximum size of input images processed by Calnic C in pixels.

`MAXPIXX`
(Not Used)

`MAXPIXY`
(Not Used)

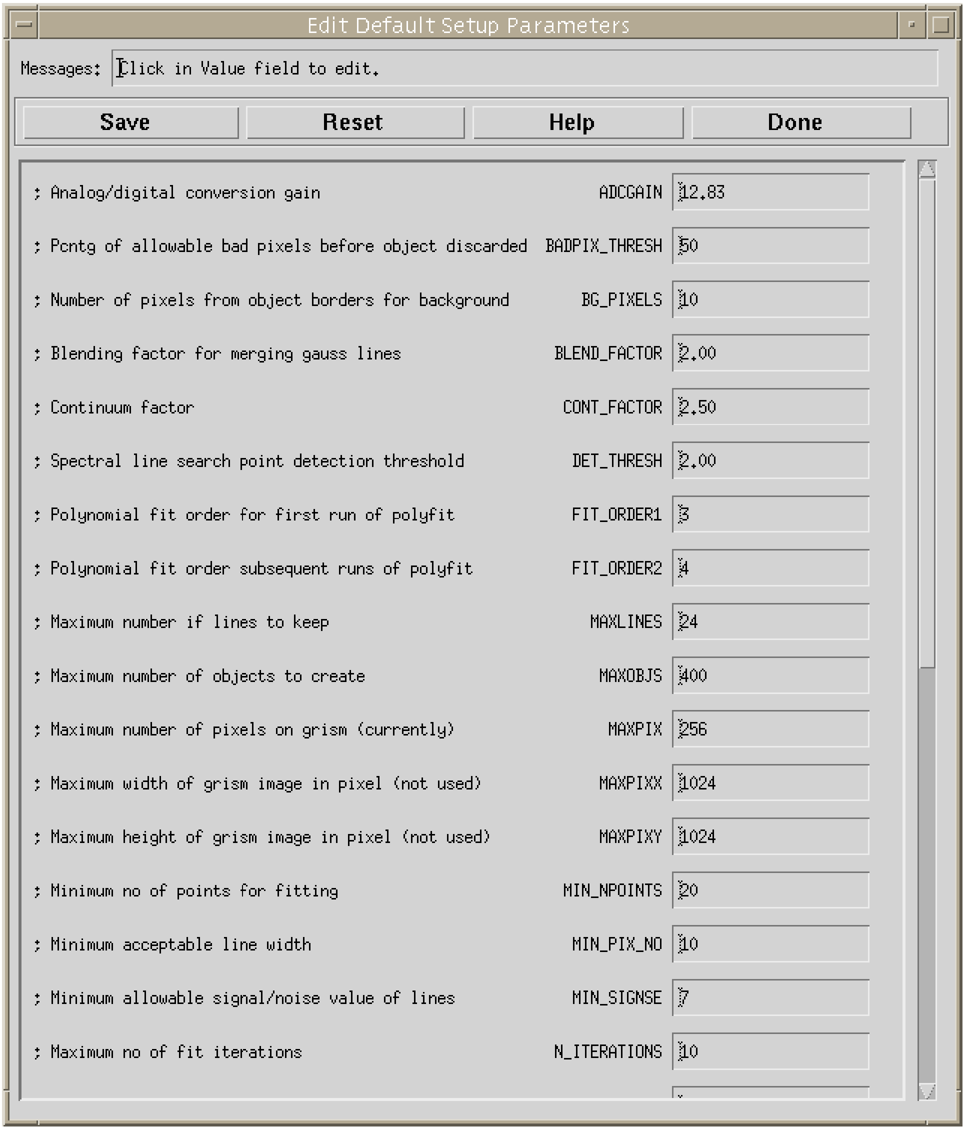


Figure 5.2: The NICMOSlook Parameter Editor. NICMOSlook provides an easy way to change the parameters of either NICMOSlook or Calnic C.

5.1.3 Calnic C Grismspec Entries

Another file in the calibration subdirectory, `grismspec.dat`, contains the filter specifications for the direct image and grism filters. It is possible to modify these parameters while running NICMOSlook. Use the button Edit Grismspec to change the values. From within

IDL, simply type `nlk_edit_grismspec` at the prompt. This will pop-up the parameter editor. All parameters are described in plain English and can be changed (see Figure 5.3). The changes take effect immediately after hitting the Save button on the editor pop-up. In addition, the values are written back to the file `$NICMOSLOOK_BASE/`grismspec.dat. See section C.1 for more details.

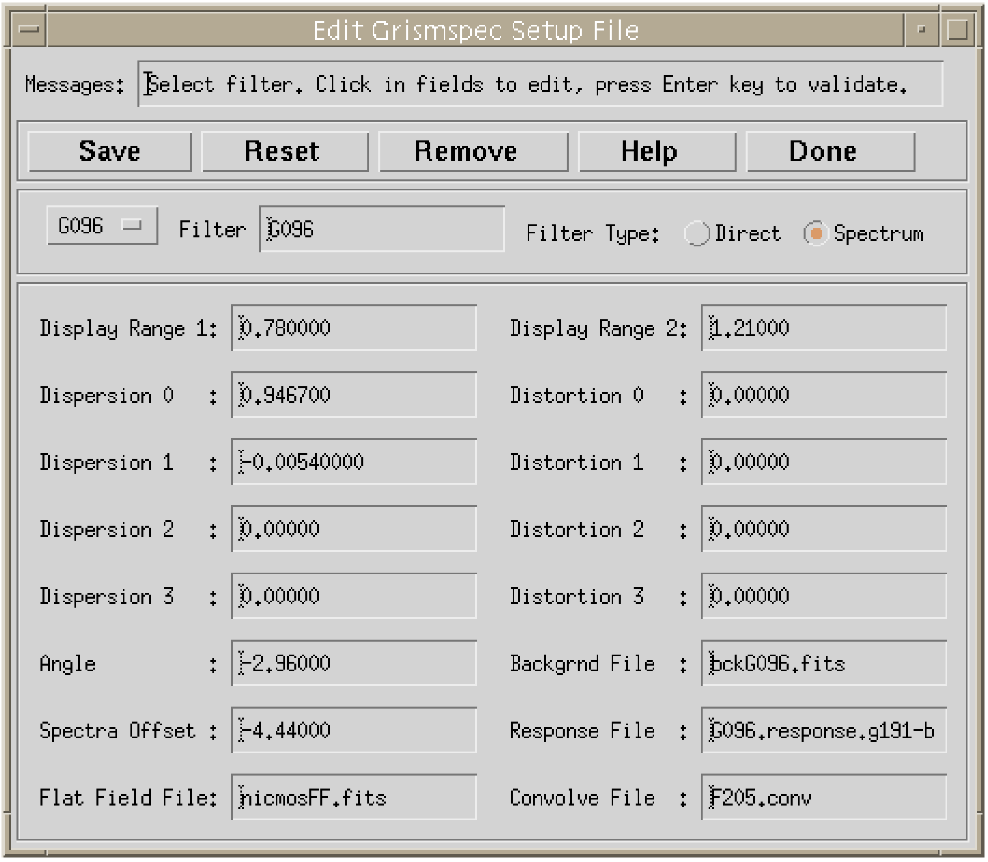


Figure 5.3: The NICMOSlook Grismspec Editor.

5.1.4 Calnic C Configurable FITS Keyword Values

The file, `fitskeywds.dat`, contains entries for FITS keywords used by Calnic C. Since it is likely that different data sets will use different keyword values for specifying the same item of interest to Calnic C, flexibility is needed. For example, some FITS files might use the keyword `OPT_ELEM` to specify the optical component used for an observation. Other FITS files on the other hand, might use the keyword `FILTER` to specify the same optical component! It is possible to edit the value of the keywords used by Calnic C (see Figure 5.4) while running NICMOSlook. Use the button Edit FITS Keywords to change these values. From within IDL, simply type `nlk_edit_fits_keywords` at the prompt. This will pop-up the parameter editor. The changes take effect immediately after hitting the Save button on the editor pop-up. See section C.3 for the default values.

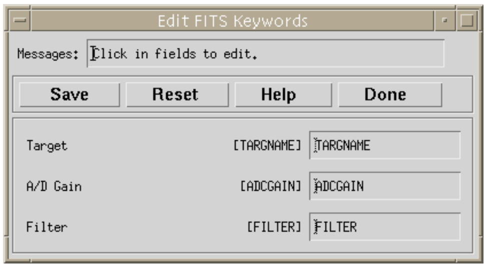


Figure 5.4: The NICMOSlook FITS Keyword Editor.

5.1.5 SExtractor Specific Files

The `calib` subdirectory also contains 4 files prefixed with `default`. These are used by SExtractor to modify its behavior in object searches, and its output formats.

See the document entitled "SExtractor 1.0 User's Guide" for a more detailed description of these files.

5.2 IDL

Calnic C is designed to set up all the necessary IDL environment variables. The only system requirement is that IDL is licensed for the workstation on which calnic C is run. It is also imperative that the machine support dynamic (run-time) linking of shared libraries. Check with your system administrator if linking problems are encountered.

5.3 Environment Variables

See the section 2.2 for information on setting up the proper `CALNICC_BASE` environment variable. Below is a description of the environment variables automatically set when Calnic C is run, along with those that should be set prior to running.

These following three variables SHOULD be set to any existing directory where the user has read/write permission. If they are not set, the CURRENT working directory (where Calnic C was started) is used as the default!

`CNC_CAT`	Catalog output directory
`CNC_DATA`	Directory Calnic C looks for input data files

`CNC_SPECOUT` Default directory to write Calnic C output files

The following variable MUST be changed so that the `CALNICC_BASE/bin` *is included in the path. Placing it in your shell startup script (e.g.* `~/.cshrc`*) is recommended*

`PATH` Command search path

The Bourne Shell (`$CALNICC_BASE`/bin/calnicc) may be edited, but this is not recommended. If it is decided to edit the file, please bear the following in mind:

The following variables MUST not be changed by the user:

`IDL_STARTUP` IDL startup script

`CALNICC_BASE` Calnic C's installation base directory

`CNC_CAL` Calibration directory

`CNC_SW` Calnic C software directory

`CNC_LIB` Library directory

The following variable MAY be changed provided that the `CALNICC_BASE/bin` *is somewhere in the path:*

`IDL_PATH` The IDL search path

5.4 Debugging

If Calnic C does not behave in the manner expected, it is possible to get some detailed debugging information and images written into the `$CNC_SPECOUT` directory. All debug-related files are written in this directory along with the other output files in order to keep the output of a single run together.

5.5 Debugging Levels

By selecting a `DEBUG_LEVEL` of greated than 0 you are telling Calnic C to open the additional output file, `DEBUG`, for writing debug information. Higher values of `DEBUG_LEVEL` cause Calnic C to write even more detailed information to this file. Levels of greater than 100 instruct Calnic C to create two new output files that contain the grism image with the pixels used by the objects set to the background level, and an image showing the pixels used in background estimation calculation. For more information regarding these output files, see section 2.6.4.

Chapter 6

Integration into Pipeline

6.0.1 CALNIC A & B

As already mentioned in section 2.6.2 CALNIC C expects a pair of calibrated direct and grism images as input.

There are two steps in the calibration pipeline for NICMOS:

> the first one, CALNIC A, is meant to remove instrumental signature from each given raw exposure;

> the second one, CALNIC B, has been introduced in the pipeline with the advent of the concept of *associations* in the data-handling of the new instruments, NICMOS and STIS. From all the exposures in the association it builds the final product.

The idea beyond an association is that an observer, in order to improve flat-fielding or to remove cosmic rays or to subtract the thermal background, can ask for multiple exposures for a given target; these exposures are logically grouped and used together to obtain a single final product.

Therefore CALNIC B, getting as input the FITS association file (provided by the stsdas pipeline) builds the final product from all the calibrated exposures part of that association.

Both CALNIC A and CALNIC B are written in ANSI C. Data I/O is accomplished using IRAF I/O library functions. A set of C to IRAF interface routines (developed at STScI) provides the link between the calibration software and the IRAF libraries. Therefore 'calnica' and 'calnicb' are stand-alone programs which can be run at the unix prompt level:

```
% cd $SOURCE_DIR (where the input files have been retrieved)
% calnica input_raw [output_cal]
```

where input_raw is the filename of a raw exposure; the default output filename is the dataset name"_cal.fits". E.g.,

```
% calnica n3s2150ao_raw.fits
```

will produced a calibrated exposure called n3s2150ao_cal.fits

% calnicb asn_id

where asn_id is the association identifier to which corresponds an association binary fits table called asn_id."_asn.fits". E.g.,

% calnicb n3s215040

will read the n3s215040_asn.fits association table and produce the final product n3s215040.fits -and all the other intermediate products.

6.0.2 Request Handler

CALNIC A and CALNIC B will be integrated into the ST-ECF On The Fly Recalibration Pipeline in such a way that an archival researcher or a CALNIC C user will be able to browse through the HST observation log via the usual web interface, and will be able to submit a request for the desired associations.

At that point the ECF archive, in an unattended way driven by the HST database, will retrieve all the raw data belonging to the marked association, along with the required reference files and association table. The automatic system will apply CALNIC A and CALNIC B to the data so retrieved.

The final product, but also all the intermediate ones, will be available to the requester via FTP, LAN, DAT, EXABYTE or CD-ROM as specified at the time of the request.

The researcher can then run CALNIC C on his/her host.

A second possibility is to integrate CALNIC C in the On The Fly recalibration pipeline.

In both cases a problem arises: CALNIC C needs both the direct and the grism images. At the moment the direct and grism exposures are not part of an association. In other words there is no knowledge in the HST database -nor anywhere else- of which direct image is associated to a given grism image.

A way to circumvent this problem would be to use the same approach which is now in development by the ST-ECF Archive Team to associate WFPC2 exposures, that is: to build an association by using the available information on the pointing -right ascension, declination and roll angle- since the direct and grism images have to be very well aligned onto the sky. This extra step is not in place yet.

Chapter 7

Acknowledgements

A number of people contributed to the development of Calnic C. Rudolf Albrecht supported the development starting from the first concepts through its final implementation. Lin Yan computed the background image and described it in this manual. Richard Hook computed the PSFs used for the weighting of the extraction.

Alberto Micol and Benoit Pirenne defined the catalog format and integrated Calnic C into the STSDAS pipeline. Alberto Micol also wrote chapter 6 chapter. Jeremy Walsh kindly took time to proofread this manual. Emanuel Bertin kindly allowed us to use the SExtractor program to be used for this work. The algortithms employed in Calnic C were repeatedly discussed with Rodger Thompson, Betty Stobie and Dave Axon. Their input and suggestions were invaluable.

Appendix A

Calnic C Utilities

A.1 Reading Spectral FITS Tables Into IRAF

It is possible to read the output FITS binary tables into an IRAF `onedspec` format with `spec_tab2im.cl`.

Within IRAF, spec_tab2im should be defined as a task by typing

```
task spec_tab2im = spec_tab2im.cl
```

Subsequently, NICMOS FITS spectra can be converted to IRAF images by typing

```
task spec_tab2im *.fits
```

Appendix B

FITS Output File

B.1 FITS File Description

A single FITS file is created for each run of Calnic C on an image pair. The first entry in this file is an extensive primary FITS header that contains general information of the observation as well as setup information for the series of binary table extensions that follow the header. Table B.5 gives a detailed description of this primary header.

B.2 FITS Binary Table Description

Following the primary FITS header in the Calnic C FITS output file are the binary tables which contain information for each object detected on the image pair. *One binary table is written out for each object detected on the image pair.* Preceding each table, is an extensive FITS header describing the object whose spectral information is contained in the table. The binary table consists of five columns which contain an object's, flux (in the units indicated in the FITS keyword `UNITS`), wavelength (in microns), statistical errors, deblending errors, and total errors. The error vectors contain values for each flux/wavelength pair for the entire spectrum. An example of a FITS table header is in table B.5

B.3 FITS Table Primary Header Example

Name	Description
SIMPLE = T	/ Written by IDL: 8-Feb-1998 18:24:08.00
BITPIX = 8	/
NAXIS = 0	/
EXTEND = T	/Binary Table follows
ADCGAIN = 6.50000	/ analog-digital conversion gain (electrons /DN)
DROOTNM = 6.50000	/ analog-digital conversion gain (electrons /DN)
Direct image header information	
DROOTNAM= 'FIELD1_F110W'	/ rootname of the observation set
DPARALLE= 'NO '	/ indicates if observation taken in parallel
DTARGNAM= 'FIELD1_F110W'	/ rootname of the observation set
DRA_TARG= 291.09258	/ right ascension of the target (deg) (J2000)
DDEC_TAR= 9.8992222	/ declination of the target (deg) (J2000)
DCAMERA = 3	/ Camera in use (1, 2, or 3)
DAPETURE= 'NIC3-FIX '	/ aperture in use (NICi,NICi-FIX,NIC2-CORON/ACQ)
DOBSMODE= 'MULTIACCUM '	/ array readout mode (ACCUM, MULTIACCUM, etc.)
DFILTER = 'F110W '	/ filter wheel element in beam during observation
DBAKEST1= 0.00000	/ background estimate number 1
DBAKEST2= 0.0000000	/ background estimate number 2
DBAKEST3= 0.0000000	/ background estimate number 3
DPHOTMOD= 'NICMOS,3,F110W,DN'	/ photometry mode
DPHOTFLA= 6.0935211e-19	/ inverse sensitivity (ergs/cm**2/Angstrom/DN)
DPHOTFNU= 2.5787892e-06	/ inverse sensitivity (JY*sec/DN)
DPHOTZPT= -21.100000	/ ST magnitude system zero point (mag)
DPHOTPLA= 11263.781	/ pivot wavelength of the photmode (Angstroms)
DPHOTBW = 1683.5078	/ RMS bandwidth of the photmode (Angstroms)
DPOHTPDG= 'INFLIGHT	
20/04/1997 - 20/05/1997'	/
DBAKPDGR= ' '	/ background model parameters table pedigree
Grism image header information	
ROOTNAME= 'FIELD1_G096'	/ rootname of the observation set
PARALLEL= 'NO '	/ indicates if observation taken in parallel
PROPOSI = 'UNKNOWN '	/Unable to determine from GRISM image
PEP_EXPO= 'UNKNOWN '	/Unable to determine from GRISM image
LINENUM = '2.012 '	/ PEP proposal line number
PR_INV_L= 'Alex Storrs '	/ last name of principal investigator
PR_INV_F= ' '	/ first name of principal investigator
PR_INV_M= ' '	/ middle initial of principal investigator
ORIENTAT= -148.06100	/ position angle of image y axis (deg. e of n)
SUNANGLE= 2.4704090	/ angle between sun and V1 axis

B.4 FITS Table Primary Header Example: continued

Name	Description
MOONANGL= 2.0739380	/ angle between moon and V1 axis
SUN_ALT = 75.552116	/ altitude of the sun above Earth's limb (deg)
FGSLOCK = 'FINE '	/ commanded FGS lock
(FINE,COARSE,GYROS,UNKNOWN)	
DATE-OBS= '24/10/97 '	/ UT date of start of observation (dd/mm /yy)
TIME-OBS= '02:57:09 '	/ UT time of start of observation (hh:mm:ss)
EXPSTART= 50745.123	/ exposure start time (Modified Julian Date)
EXPEND = 50745.124	/ exposure end time (Modified Julian Date)
EXPTIME = 95.963568	/ total integration time (sec)
EXPFLAG = 'NORMAL '	/ Exposure interruption indicator
CAMERA = 3	/ Camera in use (1, 2, or 3)
APERTURE= 'NIC3-FIX '	/ aperture in use (NICi,NICi-FIX,NIC2-CORON/ACQ)
OBSMODE = 'MULTIACCUM '	/ array readout mode (ACCUM, MULTIACCUM, etc.)
BACKEST1= 0.00000	/ background estimate number 1
BACKEST2= 0.0000000	/ background estimate number 2
BACKEST3= 0.0000000	/ background estimate number 3
FILTER = 'G096 '	/ filter wheel element in beam during observatio
PHOTMODE= 'NICMOS,3,G096,DN'	/ photometry mode
PHOTFLAM= 1.3688290e-18	/ inverse sensitivity (ergs/cm**2/Angstrom/DN)
PHOTFNU = 4.6370624e-06	/ inverse sensitivity (JY*sec/DN)
PHOTZPT = -21.100000	/ ST magnitude system zero point (mag)
PHOTPLAM= 10077.604	/ pivot wavelength of the photmode (Angstroms)
PHOTBW = 1211.7609	/ RMS bandwidth of the photmode (Angstroms)
PHOTPDGR= 'INFLIGHT	
20/04/1997 - 20/05/1997'	/
BACKPDGR= ' '	/ background model parameters table pedigree
END	

B.5 FITS Table Header Extension Example

Name	Description
XTENSION= 'BINTABLE'	/Written by IDL: 8-Feb-1998 18:24:09.00
BITPIX = 8	/
NAXIS = 2	/Binary table
NAXIS1 = 20	/Number of bytes per row
NAXIS2 = 54	/Number of rows
PCOUNT = 0	/Random parameter count
GCOUNT = 1	/Group count
TFIELDS = 5	/Number of columns
EXTNAME = ' 0'	/ Table extension (object) number
TFORM1 = '1E '	/Real*4 (floating point)
TTYPE1 = 'LAMBDA '	/Label for column 1
TFORM2 = '1E '	/Real*4 (floating point)
TTYPE2 = 'FLUX '	/Label for column 2
TFORM3 = '1E '	/Real*4 (floating point)
TTYPE3 = 'STAT_ERR'	/Label for column 3
TFORM4 = '1E '	/Real*4 (floating point)
TTYPE4 = 'DEB_ERR '	/Label for column 4
TFORM5 = '1E '	/Real*4 (floating point)
TTYPE5 = 'TOT_ERR '	/Label for column 5
Sextractor Object Information	
ROOTNAME= 'FIELD1_G096_SPC'	/
CLASS = 0.586170	/ Object type (galaxy or star)
A_OBJ = 0.787258	/ Major axis of object
B_OBJ = 0.558482	/ Minor axis of object
XCOORD = 26.6336	/ x-coord of object in pixel
YCOORD = 30.7478	/ y-coord of object in pixel
RA_OBJ = '19:24:21.00'	/ Right ascension of object (deg) (J2000)
DEC_OBJ = '+09:54:16.60'	/ Declination of object (deg) (J2000)
DRA_OBJ = 19.4061	/ Right ascension of object (deg) (J2000)
DDEC_OBJ= 9.90461	/ Declination of object (deg) (J2000)
THETA = -73.5649	/ Position angle of object
Spectral Line Extraction	
UNITS = 'mJy '	/ Units used
NODEB = 1	/NODEBLEND option
NOFLAT = '0 '	/NOFLATFIELD option
NOBACK = '0 '	/NOBACKGROUND option
NOINTER = '1 '	/NOINTERPOLATE option
EEXTR = 1	/EXT_EXTRACT option
NOWEI = '1.00 '	/NOWEIGH option
NODTQ = '0 '	/NODTQ option

B.6 FITS Table Header Extension Example : Continued

Name	Description
DIR_FN = '/vy2-2/cal_archive/field1_f110w_mos.fits'	
GRI_FN = '/vy2-2/cal_archive/field1_g096_mos.fits'	
CD1_1 = 4.8214900e-05	/Pixel size (deg) in x of direct image
CD2_2 = -4.8068300e-05	/Pixel size (deg) in y of direct image
LINEPK1 = -0.845835	/ Line peak
LINELAM1= 0.875775	/ Central wavelength
LINEWID1= 0.0496784	/ Width of line
LINESN1 = 43.0659	/ Signal to noise
CHI2_1 = 47.3240	/ Chi-square
ILF_1 = -0.0447643	/ Integrated Line Flux
LINEPK2 = 0.353261	/ Line peak
LINELAM2= 1.05525	/ Central wavelength
LINEWID2= 0.00899492	/ Width of line
LINESN2 = 7.74115	/ Signal to noise
CHI2_2 = 8.49329	/ Chi-square
ILF_2 = 0.00338510	/ Integrated Line Flux
NLINES = 2	/ Actual # of lines
CONSHORT= 0.772800	/ Continuum lambda < first line
CONLONG = 1.29911	/ Continuum lambda > last line
RMSFIT = 0.0594298	/ RMS of last fit
FORDER = 4	/ Order of last fit
MAG_BEST= -6.52138	/ Best magnitude estimate
MAG_ERR = 0.00626937	/ Error in magnitude estimate
BAKUSED = 0.153534	/ Background estimate used
BACKFIL = '/calib/bckG096.fits'	/ Background
END	

Appendix C

Setup

C.1 Grism Specification File

C.1.1 Contents

The grism specification file, `grismspec.dat`, defines the filters and grisms Calnic C recognizes and the parameters Calnic C assumes for each of them. It is found in the `$CALNICC_BASE/calib` directory.

The file is read by Calnic C and specifies a keyword to read the name of the filter or grism. For each recognized name, it list then:

1. whether the image is a direct image or a grism image
2. in case of a grism image

   the position of the spectrum in dispersion direction

   the dispersion relation of the grism

   the distortion of the spectra

   the offset of the spectrum relative to the direct image

   a 3d flatfield (FITS) file to be used for flatfield correction

   a file which defines the convolution kernel for object detection

   a FITS file containing image background data

   a file which defines the filter response curve
3. the FITS file with the appropriate PSF for the direct image or grism used to identify objects.

C.1.2 Format

Parameters in `grismspec.dat` are listed one entry (filter or grism) per line. They are written format free and have to be given in the right order. Comment lines can be placed at any place in the file. They are recognized by a ";" in the first column.

Each entry must start with the keyword in the image header which is used to identify the image. Next, the value for this keyword and the corresponding image type is given (d= direct image, s= spectrum).

The subsequent information in an entry depends on whether the entry refers to a direct image or a grism image.

Direct Image

For a direct image, the only additional information listed is the name of the FITS file with a PSF used to search for objects on the image. The location of the corresponding FITS file has to be in the same directory directory as `grismspec.dat`

Grism Image

The next two numbers are the wavelength range to consider during the actual spectral extraction process. Those two numbers are in microns. The next four numbers define the dispersion relation, parameterized with four parameters $a0$, $a1$, $a2$, and $a3$. They are used to compute the wavelength λ in microns via

$$\lambda = a0 + a1 \cdot x + a2 \cdot x^2 + a3 \cdot x^3 \tag{C.1}$$

where x is the x-position in pixels relative to the center of the direct image.

Subsequently, four parameters for the the image distortion (in y direction) $b0$, $b1$, $b2$, and $b3$ are listed. They parameterize the distortion as:

$$\delta = b0 + b1 \cdot r + b2 \cdot r^2 + b3 \cdot r^3 \tag{C.2}$$

where

$$r = \sqrt{(x^2 + y^2)} \tag{C.3}$$

is the radius position in pixels and δ is the y-deviation in pixels of the spectrum from a horizontal line.

The next parameter is the angle θ which specifies a possible small global rotation of the spectrum relative to an exact alignment along a row. The angle is measured counter-clockwise in units of degrees.

Following the angle is the spectra offset field. This is the number of pixels in the y direction that the grism image is offset from the direct image.

The next column specifies the name of the flatfield file to be used for extraction of the spectra. The flat fields to be used for the calibration of the spectra can be specified in two different ways. One is a three-dimensional FITS files, the other one is a list of flatfield files. The file type specified in the grism setup is recognised by the name of the file. Any file name which contains the string "fits" will be interpreted as FITS file. All other filenames will be interpreted as ASCII list. The ASCII input file list is assumed to be located in the `$CALNICC_BASE/calib` directory. It should contain at least 2 rows with 2 columns each separated by a single space. The first column contains the file name of a

NICMOS flatfield. The second column contains the effective wavelength of this flatfield. The NICMOS flatfield are assumed to contain the data in extension 1, as those in the STScI generated nref directory[1] do. The flatfield names are assumed to be relative to the `$CALNICC_BASE/calib` directory.

The next column lists the name of the fits file with a PSF used to search for objects on an image.

The file containing the background estimate is listed next, followed by the a file containing the response specifications for the grism.

Refer directly to `$CALNICC_BASE/calib/grismspec.dat` for examples.

C.2 Tunable Parameters

Tunable Parameters (default values)

[1] http://ecf.hq.eso.org/nicmos/nref

ADCGAIN 12.83	; Analog/digital conversion gain
BADPIX_THRESH 50.0	; Percent of allowable bad pixels before object discarded
BG_PIXELS 2	; Width of region for background estimate in pixels
BLEND_FACTOR 2	; Blending factor for merging gauss lines
CONT_FACTOR 2.5	; Continuum factor
DAO_THRESH 2	; Brightness threshold for daofind
DEBLEND_ERR_MIN 0.1	; Deblending error minimum
DET_THRESH 2	; Spectral line search point detection threshold
EXTOBJ_THRESH 0.3	; 0.0 = extended object, 1.0 = star
FIT_ORDER1 3	; Polynomial fit order for first run of polyfit
FIT_ORDER2 4	; Polynomial fit order subsequent runs of polyfit
MAXLINES 24	; Maximum number of lines to keep
MAXOBJS 400	; Maximum number of objects to create for sextractor
MAXPIX 256	; Maximum number of pixels on grism (per dimension)
MAXPIXX 1024	; Maximum width of grism image in pixel (not used)
MAXPIXY 1024	; Maximum height of grism image in pixel (not used)
MIN_NPOINTS 20	; Minimum number of points for fitting
MIN_PIX_NO 10	; Minimum acceptable line width
MIN_SIGNSE 7	; Minimum allowable signal/noise value of lines
N_ITERATIONS 10	; Maximum number of fit iterations
N_ITERLOW 4	; Iteration number where 2nd fit order param is used
O_THRESH1 0.05	; First overlap threshold for deblending
O_THRESH2 2.0	; Second overlap threshold for deblending
O_THRESH3 3.0	; Third overlap threshold for deblending
REJ_THRESH 3	; Object rejection threshold
SZ_FACTOR 2	; 2nd order mom factor of extended obj from sextractor
WEIGHTPERC 1.00	; % of max to truncate when not using weights
DEBUG_LEVEL 10	; Debugging level
NOFLATFIELD 0	; Turn off/on (1/0) flatfielding
NOBACKGND 0	; Turn off/on (1/0) background subtraction
NOINTERPOL 0	; Turn off/on (1/0) interpolated bavkground subtraction
NODTQ 0	; Turn off/on (1/0) Data Quality check
POS_ERR 0.50	; Error estimate (in pixel) of the object position.
PUNITSW dn	; Plot units (dn, mjy, or erg)
NODEBL 1.00	; Turn off/on (1/0) Deblending. (Calnic-C ONLY!)
NOWEIGH 1.00	; Turn off/on (1/0) Weighting. (Calnic-C ONLY!)

C.3 Configuring FITS Keywords

FITS Keywords (default values)

Target	ROOTNAME	Target of observation
A/D Gain	ADCGAIN	Analog to Digital conversion gain
Filter	FILTER	Filter used in observation

Appendix D

Input File Format

D.1 NICMOS Input FITS File Description

The FITS format is used for storing NICMOS science and calibrator reference data. Multiple images are stored in a single file using FITS Image extensions. The layout of the FITS input data files that NICMOSlook expects is:

Primary FITS header

Header for Science Data

Image for Science Data

Header for Error plane

Image for Error plane

Header for Quality flags

Image for Quality flags

Header for Sampling

Image for Sampling

Header for Integration time

Image for Integration time

D.2 NICMOS FITS Header Keyword Requirements

FITS Keyword Requirements

Name	Description
	The following keywords are REQUIRED for Calnic Cto run.
EXTEND	States file may contain extensions
NEXTEND	Number of extensions should be 5 for NICMOS data files
XTENSION	For the 5 extensions this should be 'IMAGE'
EXTNAME	Actual data stored in current extension (SCI/ERR/etc)
EXTVER	Extension version (not used currently but should be 1)
ROOTNAME	Rootname of observation set. Used for output file names
FILTER	Name of the filter/grism used in observation
	The following keywords are not required but some functionality depends on them.
ADCGAIN	Analog/digital conversion gain
BACKEST1	Background estimate 1
BACKEST2	Background estimate 2
BACKEST3	Background estimate 3
CRPIX1	x-coordinate of reference pixel
CRPIX2	y-coordinate of reference pixel
CRVAL1	First axis value at reference pixel
CRVAL2	Second axis value at reference pixel
CTYPE1	The coordinate type for the first axis
CTYPE2	The coordinate type for the second axis
CD1_1	Partial of first axis coordinate w.r.t. x
CD1_2	Partial of first axis coordinate w.r.t. y
CD2_1	Partial of second axis coordinate w.r.t. x
CD2_2	Partial of second axis coordinate w.r.t. y
EXPTIME	Commanded exposure duration
PHOTFNU	Inverse sensitivity (JY*sec/DN)

Appendix E

Object List

Calnic C can read an object list created and saved with NICMOSlook. This list contains information regarding the location, size (major, and minor moments along the axis), and the angle of the object's orientation. Below is an example output file of the an Object List created by NICMOSlook:

x Coord	y Coord	Maj Mom	Min Mom	Angle
15.0005	62.9947	0.00000	0.00000	-90.0000
157.000	65.0026	1.38696	0.489883	0.306816
82.9863	67.0088	2.87775	0.747376	-53.1466

Appendix F

Catalogue Output

The following shows an example of a human readable catalogue entry.
Object Number 0 from observation file ../data/direct.fits
RA = 0.00000 DEC = 0.00365056
Output FITS file: N33L0101T_0.fits
Object Type Probability (1.0 = star; 0.0 = extended object): 0.0140434
Object size (2nd order moments along axes): 13.8394 7.73015
Object angle along major axis:): 43.0521
Total flux of spectrum: 38295.5
Number of lines: 7
Order of last curve fit: 4

5 Most significant lines:

Flux	Wavelength	FWHM	S/N	CHI2	ILF
957.141	1.15657	0.0403234	147.223	596.2	41.1161
758.077	1.64862	0.0103599	11.1825	314.4	22.704
458.424	1.94372	0.0264251	10.5146	154.2	28.51
534.383	2.05612	0.0505358	16.7858	774.5	33.45
655.240	2.28973	0.1081202	30.5714	594.3	12.17

RMS of last polynomial fit : 71.2457
Continuum (long) : 593.520
Continuum (short): 252.996

Below is an example of a log file entry.

Date Run : Fri Sep 27 10:53:43 1996
Direct image : ./data/direct.fits
Grism image : ./data/spec.fits

Objects Found : 3
Number of extended objects : 2
Number of point Sources : 1
Spectra Extracted : 3
Spectra with no lines : 0
Spectra with 1 lines : 1
Spectra with 2 lines : 2
Spectra with more than 2 lines: 0

Warnings: none
Errors : none

Appendix G

Sample Data Files

The directory `$CALNICC_BASE/data` contains two direct/grism pair of sample data files called `ov10_*.fits`.
These are plotted in Figure G.1 and Figure G.2.

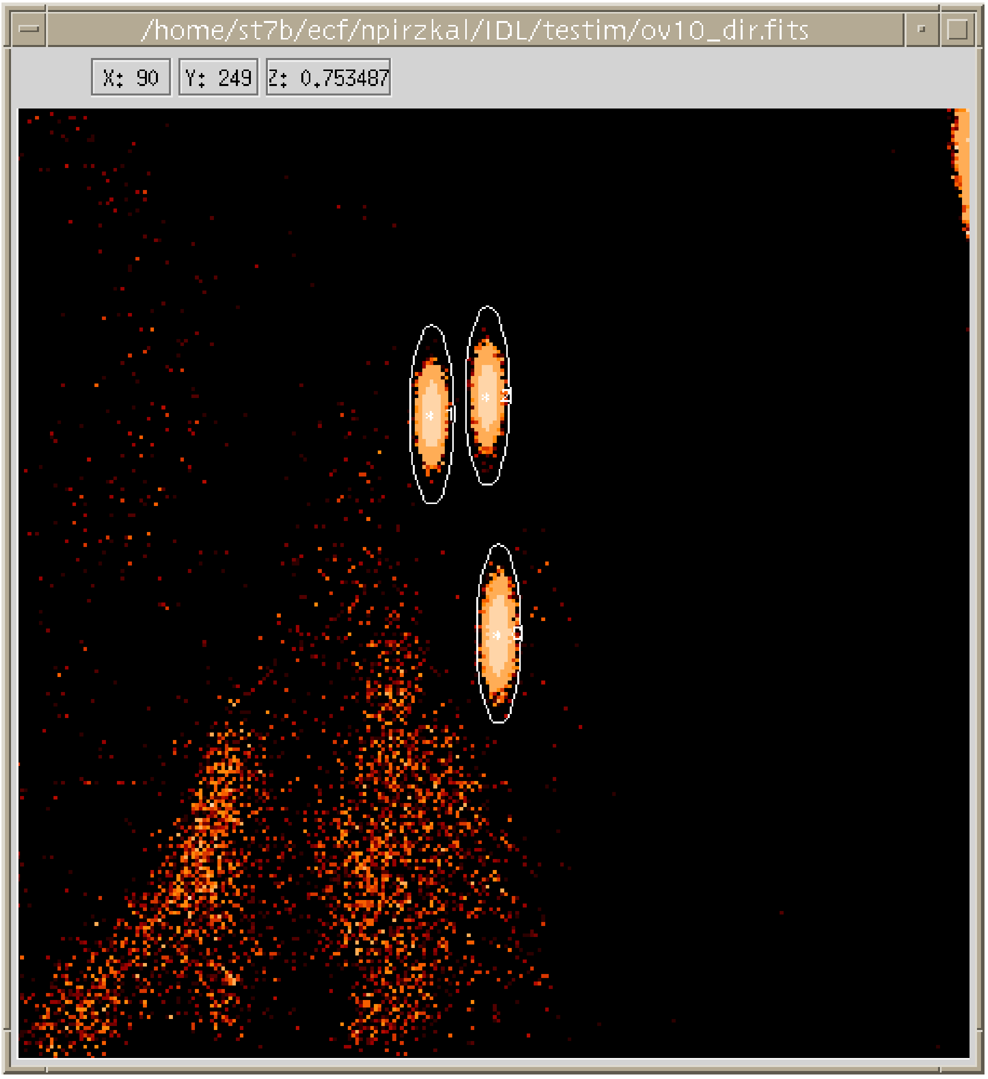


Figure G.1: Direct sample input images.

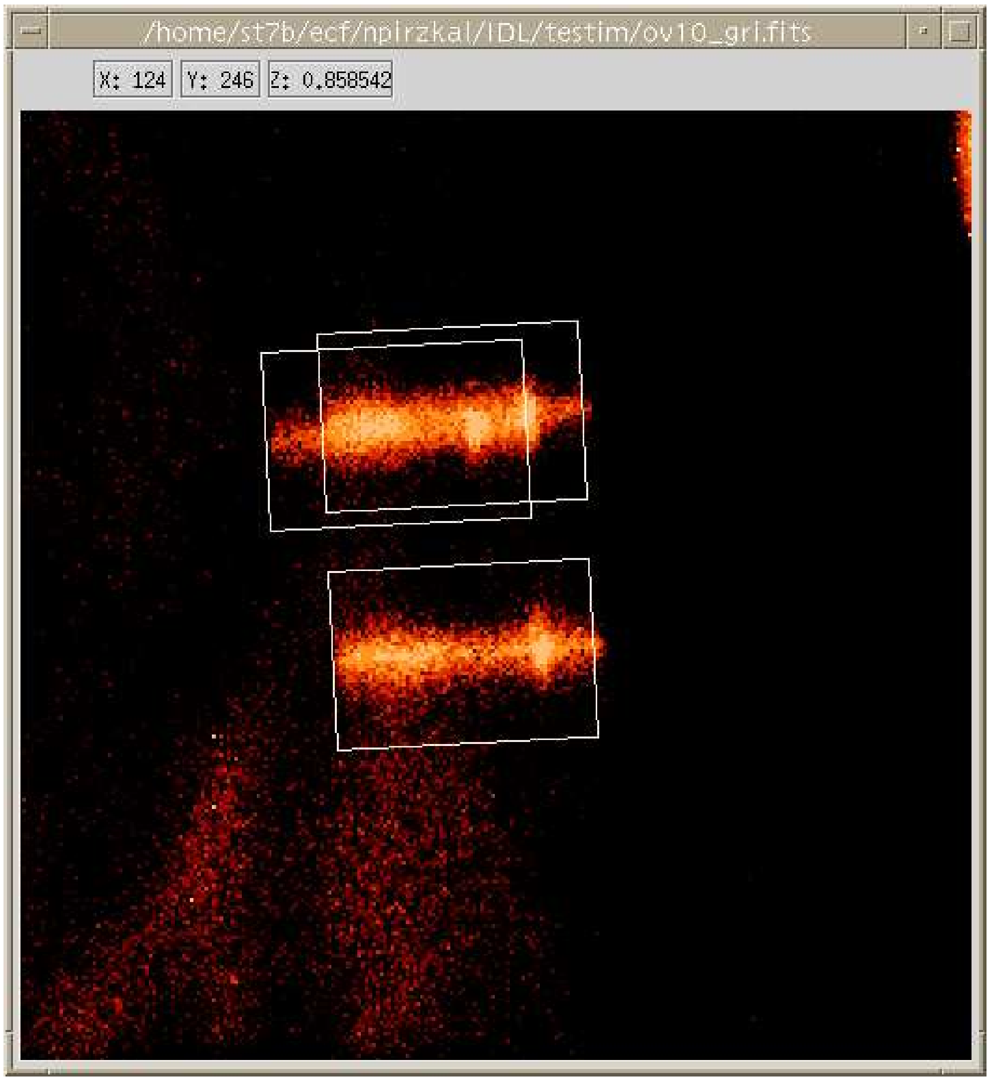


Figure G.2: Grism sample input images.

Appendix H

List of all input files

Name	Description
grismspec.dat	Descriptions of all available grisms and filters
bck*.fits	Background estimates
calnicc.setup	Tunable parameters for Calnic C
default.conv	Convolution file for SExtractor
default.nnw	Neural network weights file for SExtractor
default.param	List of SExtractor parameters to return
default.sex	Tunable parameters to modify SExtractor behavior
calnicc_tmp.cat	SExtractor output file (not used by Calnic C)
nicmosFF.fits	Flat field image for each wavelength of grism (cube)
weight_*.fits	Weight file for each grism
G*.response*	Response (Inverse sensitivity for each grism)
fitskeywds.dat	FITS keyword configuration file
F090M.conv	Convolution files used by SExtractor for each filter
F095N.conv	
F097N.conv	
F108N.conv	
F110M.conv	
F110W.conv	
F113N.conv	

Input files to Calnic C: continued

F140W.conv	Continuation of convolution files for each filter
F145M.conv	
F150W.conv	
F160W.conv	
F164N.conv	
F165M.conv	
F166N.conv	
F170M.conv	
F171M.conv	
F175W.conv	
F180M.conv	
F187N.conv	
F187W.conv	
F190N.conv	
F196N.conv	
F200N.conv	
F204M.conv	
F205W.conv	
F207M.conv	
F212N.conv	
F215N.conv	
F216N.conv	
F222M.conv	
F237M.conv	