SNAP: Speedy Near-infrared data Automatic Pipeline Version 1.1
(DRAFT)

Filippo Mannucci
IRA - CNR, Firenze, Italia

Nov 2002

Abstract. We describe the Speedy Near-infrared-data Automatic Pipeline (SNAP), a pipeline for automatic reduction of the imaging data from near-infrared cameras. It is completely automatic and works both for deep imaging and for mosaicing of fields much larger than the camera field-of-view. SNAP makes use of several other programs as IRDR, SExtractor, IRAF and Drizzle. It was developed for the camera NICS at the Italian national telescope Galileo on La Palma, but it also works with data from other cameras.

1. In brief, how to use SNAP

– Two scripts are given to reduce data from the NICS and ARNICA cameras (commands `nics` and `arnica` respectively [1]). More scripts will be prepared in the future.

– These command apply to all the *.fts images present in the working directory. Therefore the first thing to do is to copy the raw images to be reduced and coadded in an empty directory.

– In case of NICS data type:
`nics [TYPE] [MOSAIC]` , where:

- `TYPE` is type of reduction required, chose one among `single, double, full` (default: `double`). With `single` the images are coadded after a simple single-pass sky subtraction; the result is in `single.fits`; `double` performs a double-pass sky subtraction with object masking. Appropriate for imaging with the NICS small field optics or when the camera field distortion can be neglected. The results is in `double.fits`; `full` performs the full procedure, with a double-pass sky subtraction with object masking and correction for field distortion. Result in `coadd.fits`.

[1] For consistency with the previous version, the `snap` command is still present and coincides with `nics`

- `MOSAIC` is the mosaic pattern, chose one among `dither, onoff, offon` (default: `dither`); `dither` means that all the images are "on source"to be reduced,
  `onoff` means that a chopping ON-OFF technique was used, starting with a ON, therefore only even images will be used for sky subtraction and only odd images will be coadded; `offon` is the same, but starting with a OFF.

– The `arnica` command has the same format:
`arnica [TYPE] [MOSAIC]`
but in this case the option `TYPE=full` is not present because field distortion is not to be corrected.

– The commands `nics help` or `arnica help` show the available options.

– If needed, clear all the intermediate files with the command `clearsnap`.

No other interactions are needed to obtain the final coadded image. The requested time depends on the size and the number of images, on the reduction type and on the computer. For ten 1024×1024 images on an Pentium4 PC and `TYPE=double` it is about 2 min.

There are also some parameters than can be changed if needed. They are listed in the `snap.par` file, the default values are used if no such file is present in working directory.

2. What SNAP does and does not

The pipeline can perform a full reduction, with flat-fielding, sky subtraction, computation of the offsets, correction for geometrical distortion, object masking and correction for cross-talk between the quadrants. It is completely automatic, i.e., a single command is usually enough to obtain the final result. Briefly, the pipeline works as follow:

1. a master flat-field in computed by combining all the appropriate images without offsets; if the mosaic is of an ON-OFF type, only the sky frames are used.

2. a bad pixel mask, is computed by detecting, in the master flat field, the pixels with deviant values;

3. if provided, an external bad pixel mask (`badmask.pl`) is also used, adding the bad pixel in the two lists;

4. for each image, a sky frame is computed by combining a certain number of the closest images;
5. this sky frame is subtracted by the image and the result is divided by the master flat;
6. bright objects are detected by SExtractor in these cleaned images to measure the offsets among the images; the object mask are multiplied by the bad pixel mask.
7. a cross-correlation algorithm among the object masks is used to measure the relative offsets. There is no need to have part of the sky common to all the images.
8. the cleaned images are combined using the offsets; this is the end of the procedure if the `single` keyword is given. Two main problems remain to be solved: the presence of faint objects in the sky frames not removed by the combination, and the presence of field distortion which, when combined to the dithering technique, can severely degrade the image quality in the outer part of the image or introduce large distortion in large mosaics of images.
9. with the `double` and `full` commands, fainter objects are detected in the coadded image in order to have a better sky subtraction;
10. the object mask is dilated by a certain factor to remove also the undetected object tails;
11. for each image a new sky is computed by taking into account this object mask;
12. if `double` is used, these images are combined by using the old offsets and the procedure stops here;
13. if `full` is used, the field distortion is removed from these cleaned images by using the parameters contained in a file;
14. the pixels containing deviant pixels are identified and flagged;
15. the old offsets could be effected by field distortion, therefore new offsets are computed for the undistorted images;
16. finally, the cleaned corrected images are combined.

The current version 1.1 of SNAP has some limitations:

– it is not possible to use an external (dome or dawn/dusk) flat;

– the effect of saturated objects on the other quadrants and on the following images is not properly corrected cross-talk of saturated

Some of these features might be implemented in the following versions of SNAP.

SNAP was written for the NICS camera, but it can be easily be used for other cameras as long as the correct header keywords and distortion parameters are provided.

3. The tools used by SNAP

SNAP made maximum use of already existing software in order to minimize the effort and to use reliable and well known programs. Nevertheless many modifications were necessary to apply these programs to the current problem and also to correct some errors.

A few steps make use of the InfraRed Data Reduction (IRDR) software. It is a library of C routines written Sabbey et al. (2001) to reduce the data of the CIRSI camera. These routines were corrected in some points and modified for the use of a different camera. Several new pieces of software were written in order to improve their performances and add new features, as the construction of large software.

SExtractor (Bertin & Arnouts 1996) is used to detect the objects in the images both to compute accurate offsets and to create the object masks to improve the sky subtraction.

The correction of the field distortion is made by Drizzle and DitherII, programs originally written by Hook and Fruchter (2000) to correct and combine the WFPC2 data. Alessandro Marconi wrote a new piece of software to be able to use the analytic description appropriate for the NICS field distortion. This form, based on a 6th order polynomial on the radial distance, was not present in the original Drizzle code. The parameters of this polynomial were measured by Leslie Hunt and Javier Licandro.

IRAF is also used in several steps because of its easy and versatile use. In particular IRAF non-interactive host CL scripts are used to compute the master flat-field, to correct the image header keywords and to drive the use of Drizzle. This capability of IRAF CL scripts to

be executed as host commands was introduced with the IRAF version V2.11.2 and its still primitive. For this reason this part is expected to evolve rapidly.

Finally, a FORTRAN program `crt_nics.f`, written by Tino Oliva, is used to correct the cross talk between the quadrants. The parameters of the correction for the NICS camera were measured by Leonardo Testi.

4. Step-by-step description

Now we describe each step of the pipeline in a greater detail and list the parameters than can be changed if needed.

4.1. Parameters you can change

Some tasks depend on a number of parameters. The default values were chosen to have a good result in most of the situation, but is is unlikely that they work properly in every situation. A few of them were therefore put in a file, `snap.par`, that can be found in the SNAP main directory. If necessary, the user can copy this file to its current directory and edit the values. The new values are used if this file is present in the current directory, otherwise the default file is read.

4.2. Image selection

SNAP looks for all the `*.fts` images present in the current directory. Therefore the user must select the images to be reduced and combined and put them in an empty directory. This is probably a little more accurate than the common technique of defining a image name root and a range of serial number, but it is much more flexible as it allows for the combination of list of non contiguous images.

4.3. Correction for the cross-correlation

NICS has a severe cross-talk among the signal in the various quadrants. The effect is of the order of 3% of the input flux and therefore it is easily detectable in the final combined image. The FORTRAN77 program `crt_nics.f` correct for this effect by using a set of parameters contained in the program itself.

4.4. Header keywords

SNAP uses several keywords of the FITS header of the image. In particular it looks for:

- `RA` and `DEC`: the R.A. and DEC. of the current position of the telescope. They are used to have a reliable first guess position when computing the offsets. `RA` is in hours, `DEC` in deg.
- `EXPTIME`: integration time in sec., used during the final combination
- `SCALE`: pixel scale in arcsec/pix;
- `POSANG`: position angle of the image, in degrees. If 0, north is up and east is left.

The presence, the name and the format of these parameters are very machine-dependent and therefore it could be necessary to change their value often when reducing data from different telescopes. To solve this problem we decided to use IRAF to set the proper header keyword instead of putting into the C codes all the possible choices. In such a way the users can easily edit the IRAF procedure `head.cl` according to their needs without the need of recompiling the library.

4.5. Computing the flat-field

All the input images are combined by the IRAF task `imcombine`. The input images are scaled to have the same median, the pixels containing objects are rejected by an algorithm based on the measured noise (`sigclip`), and the flat field (`flat.fits`) is obtained by a median. This procedure is more robust than the corresponding IRDR task `cubemean`: the latter does not use rejection algorithms and therefore the trace of some objects can sometimes be found in the flat field.

4.6. Computing the bad-pixel mask

The flat field is searched for bad pixels. They are defined in two ways: they have either values `nsig`=$\pm 5\sigma$ from the average of the surrounding box of `nxblock`×`nyblock` pixels (default to 16×16), or have value below `mingain`=0.7 or above `maxgain`=1.4 times the average gain. The flat field is then normalized to 1 to obtain the gain map `gain.fits` and the bad pixels are set to 0. The values of `nsig, nxblock, nyblock, mingain` and `maxgain` are contained in the `snap.par` file and can be varied if necessary. `nxblock` and `nyblock` must be even numbers.

4.7. Adding and external bad-pixel mask

If the image `badmask.pl` is present in the current directory or, if not, in the SNAP base directory, then it is multiplied by `gain.fits` to obtain

the new gain mask. The mask `badmask.pl` must have the value of 1 on the good pixels and 0 on the bad ones. One such mask is supplied to correct for small flexions of the field mask, but users are encouraged to check if it is appropriate to their data.

4.8. First pass sky subtraction

The nearest 2×`halfnsky` images are combined by a median to obtain a first approximation of the sky. This running-sky technique is found to give significant better results when the sky level has rapid variations. The default value of `halfnsky` is 4, for a total number of 8 frames used to estimate the local sky. If 2×`halfnsky`+1 is larger than the number of available images, automatically all the images (except that under process) are used to estimate the local sky. To avoid unnecessary long computation and memory use, a maximum value of `halfnsky`=10 is allowed.

The sky-subtracted image can be further flatted to correct for low spatial frequency distortion of the flat field or the sky image. If the parameter `destripe` is not set to `none`, the mode value of each row/column in each quadrant is subtracted (destripped) from the row/column itself. The possible choices are `destripe=none,row,col,colrow,rowcol` to do, respectively, no destripping, destripping along the rows only, along the columns only, first along the columns and then along the rows, first along the rows and then along the columns. Destripping usually increase the image quality, but when bright and/or extended objects are present in the field some negative structures can be introduced. Destripping can only be used with 1024×1024 arrays, all the rest works for all image formats.

4.9. Object detection

SExtractor is called to look for the objects in the field. An object mask is created containing 0 in the pixels attributed to the sky and object minus background in the pixels attributed to the objects. The parameters for this step are `off_minarea`, the minimum area above threshold to have an objects, and `off_thresh`, the detection threshold in units of sigma of the background. As the resulting mask will be only used to compute the offsets between the images, in general there is no need to detect faint objects. The default values are `off_minarea=15` and `off_thresh=3`. Reducing these two values will result in the detection of a larger number of objects, as it could be useful with shallow images of high galactic latitude fields. The opposite could be advisable for very crowded fields. The files needed to run Sextractor (`default.*`) are contained in the IRDR base directory. Using object mask instead

of images for the cross-correlation is faster and more accurate as only the interesting pixels are used, and the full information on position, shape and magnitude of the objects is retained.

4.10. Offset computation

This is the most critical step of the pipeline as it depends critically on the performances of the telescope. A deeply modified version of an IRDR procedure is called to cross-correlate the object mask and compute the translations with respect to the first frame in the list. The first guess of the offsets are computed by the RA, DEC, SCALE and ROTPOS keywords in the image FITS header. NICS image headers do not contain the SCALE keywords, therefore this information is obtained from the OBJECTIV keyword: if it is LF, the scale is assumed to be 0.25 arcsec/pix; if it is SF, a value of 0.13 arcsec/pix is used. Currently, the procedure can use the data headers from NICS, ARNICA and 2MASS, identified by the keyword `INSTRUMEN`.

For each image, the expected overlap with the first image is computed. If it is larger than 25%, the offsets are computed directly by a cross-correlation with the first image itself. If, on the contrary, it is lower than this limit, the previous images in the list is searched for the image with the largest overlap, and this is used to make the cross-correlation. This technique allows for the automatic computation of the offsets even if no common part of the sky exists among all the images. As a result, large mosaics covering a field much larger than the camera field-of-view can be automatically produced.

Offset are computed to a sub-pixel accuracy bu fitting a parabola to the peak of the cross-correlation image.

If the telescope points accurately (within a few arcsec), the procedure is found to give very accurate results. If, on the contrary, the initial guesses is missing or badly wrong, the results are not guaranteed and the computation time diverges. To have better results, the difference between each guess and the resulting offset is integrated and transfered to the following guess.

The parameter `off_err` (in arcsec) controls the initial searching radius for the best cross-correlation and therefore should be set of the order or a little larger than the expected error in the offsets. A value of a few arcsec is usually adequate, the default is `off_err=4`. The time to compute the offsets is roughly proportional to the square of this value, but too small values should be avoided because if the cross-correlation within this radius fails (see below), a new try is done by using larger limits.

The ongoing computation can be checked in two ways. For each image the program outputs the final computed offsets (used in the following steps), the fraction of object pixels of the reference image recovered by the current image, the sequence number of the reference image and the initial guess of the offsets. If the reference image is not the first one, the offsets to it are also printed out. If the distance between guessed and computed offsets is up to a few pixels, as it should be, the resulting offsets are probably right. In this case the fraction of overlapping pixels should also be high, for example above 0.5 when using small dither steps. If this fraction is below 0.1, the result is assumed to be wrong, the searching radius is doubled and the cross-correlation tried again. This iteration is done twice, reaching searching radius up to four times the input value and therefore computation times 16 times longer. If the final fraction of recovered pixels is low, the offset is probably wrong.

The difference between the expected offsets and the computed ones is "accumulated" and used to compute the following guess. This is found to be very useful when the telescope tends to make always the same pointing error (for example, loose something in azimuth) which is often the case when large errors are present. This accumulation is not done if the fraction of recovered pixels is below the threshold.

4.11. First pass coaddition

Once the offsets are computed, the sky-subtracted flat-fielded images are combined into the first-step resulting image. This is also the end of the procedure with the keyword `single`. This image can be used for some application, but the sky subtraction is likely to be far from perfect as no object masking is used. For large field imaging the PSF of the objects and No correction for field distortion were applied, therefore the image quality of large field data is probably poor.

4.12. Master object list mask

SExtractor is again used to find objects in this first-pass coadded image in order to mask then during sky computation. This time the parameters controlling the detection threshold should be set to have deeper detections and mask faint objects. The parameters involved and their default values are `mask_minarea=15` and `mask_thresh=1.0`. The resulting object mask is extended by a certain fraction to reject also the undetected object tails. The fractional amount of this extension is chosen by the parameter `expand_mask=0.5`.

4.13. Second pass sky subtraction

The running sky are computed again by taking into account the master object mask computed at the preceding step. This is a very efficient technique to remove the influence of the objects in the determination of the sky background influencing the image quality and possibly the photometry. Again, destripping is controlled by the `destripe` parameter.

4.14. Field distortion correction

The NICS large field optics has large pin-cushion field distortion amounting to about 1% near the edges of the array and to 3% near the corners. Its correction is a crucial step to obtain a good image quality when dithering and a good astrometric precision. The parameters of this distortion were measured by Leslie Hunt and Javier Licandro by using some crowded stellar fields and modeled by a 6th order polynomial on the distance from the optical axis. The values of the resulting parameters depend on the alignment of the optics and therefore might change after optics realignment. The current values can be found in:

`http://www.tng.iac.es/instruments/nics/distortion_js.html`.

The file `distortion.dat` in the SNAP main directory contains the parameters as at July 2002 in the requested format. This file can be edited if necessary. The correction is applied by using Drizzle. The flux of each input pixel is distributed among all the pixels of the output corrected image in fraction proportional to the overlapping region [2]. The task is quite time-consuming but it is found not to degraded the PSF very much. Drizzle is used only to correct the field distortion, not to coadd the images, as new offsets must be computed [3]. The gain image and object mask are de-distorted in the same way to be used in the final combination.

As final step, edge effect introduced by the distortion correction are corrected by substituting the sky median to all the pixels the pixels below the sky median for more than 4 times the sky sigma are substituted with the median value itself.

4.15. Final offsets computation

The final value of the offsets are computed again after the field distortion correction. These new value can different of several pixels from

[2] for those who know Drizzle, `pixfrac=0.7` is used.

[3] It is not possible to correct the field distortion at the beginning of the process as sky subtraction and flat fielding must be applied to the original pixels

the previous values, especially when large values of the offsets are used or when doing large mosaics. Again, the parameters `off_minarea`, `off_thresh` and `off_err` are used.

4.16. Final coaddition

The resulting images are offseted to a common value, masked by using `gain.fits` and combined by using final offsets. The combination, done by the IRDR task `dithercubemean.c`, produce a unclipped average of the input pixels weighted for their gain and for the fractional overlapping area. The final image is `coadd.fits`, while `coadd.weight.fits` contains the image weight and `final.list` the list of the used images and offsets. The final images are trimmed to remove unnecessary large edges.

A combination with the IRAF task `imcombine` is also done. The final image is named `imc.fits`.

4.17. Cleaning intermediate images and files

SNAP saves all the intermediate files to allow for an accurate check of all the steps. The command `clearsnap` deletes all the intermediate files and images saving only the original raw frames and the final result.

4.18. Parameters you can chose

The SNAP parameter file `snap.par` has this format:

```
#
# Default SNAP parameters
#
# format: parameter=value <- no spaces in between

# Parameters for the bad pixel mask
set nsig=5          # number of stddev from local bkg to be
bad pixel
set nxblock=16      # X dimen. (pixels) to compute local bkg
(even)
set nyblock=16      # Y dimen. (pixels) to compute local bkg
(even)
set mingain=0.67    # pixel below this gain are considered bad
set maxgain=1.50    # pixel above this gain are considered bad

# Running sky subtraction
```

```
set halfnsky=4      # half-width for running sky subtraction
set destripe=none   # destripe image (none|row|col|colrow|rowcol)

# Offsets computation
set off_minarea=15  # SExtractor DETECT_MINAREA used for offsets
set off_thresh=2.5  # SExtractor DETECT_THRESH used for offsets
set off_err=6       # offset precision [arcsec]

# Object mask
set mask_minarea=15 # DETECT_MINAREA used for object masking
set mask_thresh=1.0 # DETECT_THRESH used for object masking
set expand_mask=0.5 # amount to expand the object mask regions
```

5. Results and Checks

5.1. Noise

The sky noise in the final coadded image is as expected. For example, in a mosaic of 30 images each with sky level of about 4400 counts and `coadds=3`, the expected final noise in ADU is 2.49 (using gain=8 e-/ADU and read-out-noise=25 e-) and , while the measured noise was 2.54, i.e., only about 2% above the theoretical limit [4].

5.2. Offsets

No check done yet.

5.3. PSF

No check done yet.

5.4. Photometry

No check done yet.

[4] To measure this noise no field distortion correction was applied and a final combination with integer offsets was used

6. Installing SNAP

The installation is fast and completely automatic once a few parameters are defined by the user. The used computer must have a valid installation of IRAF version 2.11 or newer. STSDAS is also needed if the field correction is to be used. Only Linux-Redhat and Solaris systems were tested.

Installation proceeds as follows:

1. Uncompress and untar the files with the command: `tar xvzf snap1.1.tar.gz` or `gunzip snap1.1.tar.gz` and `tar xvf snap1.1.tar`. It creates a `snap` directory (which we will cal SNAPDIR) in the working directory.

   A `make` file is supplied to do the full installation, i.e., insert the right directories into the command files, compile the C IRDR and SExtractor libraries, compile the FORTRAN Drizzle and crt_nics programs, and set the right path.

2. The file `Makefile` must be edited to set some information. The parameters to be edited are:

   - The location of your C and FORTRAN compilers. You can get them by `which f77` for the FORTRAN, and `which gcc` (or, if missing, `which cc`). Put the results into the Makefile, as:
     `FF = /usr/bin/f77`
     `CC = /usr/local/bin/gcc`
   - `STSDAS` : the IRAF/STSDAS directory. This is only needed to compile Drizzle to correct the field distortion. Under IRAF type: `show stsdas` and insert the result into the Makefile, closing the string with a #. Some example:
     `STSDAS = /iraf/extern/stsdas#`,
     `STSDAS = /scisoft/iraf/extern/stsdas#`; `STSDAS = //usr/im/stsdas2.1.1#`;
   - `SEXMACHINE`: if the executable `sex` for SExtractor is already present in your path you don't need to install it again. If you need to install it you have to supply your machine type, chasing one of the following:
     `SEXMACHINE = aix` for IBMs RS6000 running AIX;
     `SEXMACHINE = alpha` for DEC-ALPHAs with Digital UNIX;
     `SEXMACHINE = hpux` for HP/UX systems;
     `SEXMACHINE = linuxpc` for PCs running LINUX, using gcc;
     `SEXMACHINE = linuxp2` for Pentium2/3/4 LINUX PCs;
     `SEXMACHINE = linuxk7` for Athlon PCs running LINUX;

`SEXMACHINE = sgi` for SGI platforms;
`SEXMACHINE = solaris` for SUN-Solaris machines;
`SEXMACHINE = sunos` for SUN-OS platforms;
`SEXMACHINE = ultrix` for DEC stations running ULTRIX.

3. Under IRAF, give the command: `!make`. It is important to start from an IRAF shell as only in this case all the necessary links are defined.

4. If the program SExtractor is not installed in your system, give the command (NOT from the IRAF window): `make sex`. Otherwise you don't need to install it again.

5. Add the SNAPDIR directory to the path of your commands, for example by adding a line as:
`set path = ($path /home/guest/snap)`
to your .cshrc or .tcsrch file. Apply this change with the command `rehash`.

Your are now ready to start.

7. Timing

The amount of time need by SNAP depends on the computer, on the amount of images and on the procedure used. On a computer have a Pentium4 processor with a 1.7GHz clock, 256MB of RAM, a Linux Redhat 7.3 operative system and a SciSoft IRAF version, the combination of 10 images takes the time reported in table 1.

Table I. Time to combine 10 images on a computer with P4/1.7G and 256MB RAM

Command	min
nics single	1:00
nics double	2:00
nics full	3:20

8. Warning and limitations

– Number of images: it is unlikely to obtain a good result with less that 6−10 images.

– Sky level: you must have well exposed, unsaturated images.

– Weather condition: when clouds are present, the sky transparency changes rapidly, the flat fielding procedure is usually poor and the number of detected objects changes a lot from one image to the other. In this case the procedure to measure the offsets usually fails. Usually the situation gets better when using a small value of `halfnsky`.

– Memory requirements: the final combination is quite memory expensive, as it loads all the images in the memory. About 22MB are needed for each image, therefore a maximum of about 45 images can be combined on a computer with total memory of 1GB (512MB RAM + 512MB swap).

9. History and changes from previous versions

– Version 1.0: Aug 2002

– Version 1.1: Oct 2002

- Added the `arnica` command.
- Corrected a wrong output in `mosaic.c`.
- Changed the routine to increase the search radius when computing the offsets

Acknowledgements

Several other people contributed to the SNAP pipeline. As already described Tino Oliva, Alessandro Marconi, Leslie Hunt, Javier Licandro, Roberto Maiolino, Leonardo Testi, Paolo Saracco, Marcello Lodi, Andrea Zacchei and Alessio Checcucci contributed to this pipeline with software, comments, suggestions and help.

References

Sabbey, C. N., McMahon, R. G., Lewis, J. R. & Irwin, M. J., in *Astronomical Data Analysis and Systems X*, eds. F.R. Harnden, Jr., F. A. Primini, & H. E., Payne 2001, ASP Conf. Ser. 238, 317

Bertin, E.; Arnouts, S. 1996, A&AS, 117, 393

Hook, R. N., and Fruchter, , A. S., 2000, in *Astronomical Data Analysis Software and Systems IX*, ASP Conference Proceedings, Vol. 216, N. Manset, C. Veillet, and D. Crabtree eds., p.521