SWarp
$v2.21$
User's guide

E. BERTIN

Institut d'Astrophysique de Paris

December 17, 2010

Contents

1   What is SWARP?

SWARP is a program that resamples and co-adds together FITS images using any arbitrary astrometric projection defined in the WCS standard[1]. The main features of SWARP are:

- FITS format (including multi-extensions) in input and output,

- Full handling of weight-maps in input and output,

- Ability to work with very large images (up to 500 Mpixels on 32-bit machines and 10^6 Tpixels with 64-bits), thanks to customized virtual-memory-mapping and buffering,

- Works with arrays in up to 9 dimensions (including or not two spherical coordinates),

- Selectable high-order interpolation method (up to 8-tap filters) in any dimension,

- Compatible with WCS and TNX (IRAF) astrometric descriptions,

- Support for equatorial, galactic and equatorial coordinate systems,

- Astrometric and photometric parameters are read from FITS headers or external ASCII files,

- Built-in background subtraction,

- Built-in noise-level measurement for automatic weighting,

- Automatic centering and sizing functions of the output field,

- Multi-threaded code with load-balancing to take advantage of multiple processors.

- XML VOTable-compliant output of meta-data.

2   Skeptical Sam's questions

Skeptical Sam doesn't have time to test software extensively but is always keen on asking agressive questions to the author to find out if a program could fit his needs.

S.Sam: What's the point in releasing another image co-addition software? We already had the DRIZZLE (Fruchter & Hook 1997, 2002) and MSCRED (Valdes 1998) packages. And there are now MOPEX (Makovoz & Khan 2005) and Montage (Berriman et al. 2008)...

Author: Co-addition is most-certainly the most critical step in reducing modern CCD mosaic data. Although several powerful packages are available, they did not meet all the requirements we had at the TERAPIX[2] data centre in Paris, where the project was initiated.

S.Sam: SWARP doesn't perform the astrometric calibration, does it?

Author: No it doesn't. You will have to use SCAMP[3] (Bertin 2006) for that.

S.Sam: I am the kind of guy who does high precision astrometry and photometry. My sources are detectable on raw frames. Resampling, you know, would just wreck the signal, so I prefer to combine measurements from individual images than to co-add pixels.

---

[1]see http://www.cv.nrao.edu/fits/documents/wcs/wcs.html
[2]http://terapix.iap.fr
[3]http://astromatic.net/software/scamp

Author: It is true that resampling distorts slightly both signal and noise. However, if done properly, and if the data are not undersampled (Full Width at Half Maximum > 2.5 pixels), the degradation is generally very small. For instance, resampling a single ground-based image (atmospheric seeing of FWHM 3 pixels, and conversion factor $2e^-/$ADU) twice with SWARP, and comparing fluxes and positions measured using PSF-fitting, one gets for bright stars *rms* differences of less than 5.10^{-4} mag and 10^{-3} pixel between the original and the twice-resampled images. This is already much better than what photon-noise allows for. Hence apart from situations of strongly non-stationary noise or undersampled data, the consequences of resampling are expected to be negligible.

3   License

SWARP is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. SWARP is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with SWARP. If not, see `http://www.gnu.org/licenses/`.

4   Installing the software

4.1   Obtaining SWARP

The easiest way to obtain SWARP is to download it from the official website[4], or alternatively from the current official anonymous FTP site[5]. At this address the latest versions of the program are available as standard `.tar.gz` Unix source archives as well as documentation and RPM binary packages for various architectures. It is strongly advised to install one of the RPM packages if you intend to run the software for production purposes on a Linux machine with x86 architecture and RPM-support, as the RPM executables have been optimized for execution speed.

4.2   Software and hardware requirements

SWARP has been developed on Unix machines (Compaq Tru-64 and GNU/Linux), and should compile on any POSIX-compliant system. The software is run in (ANSI) text-mode from a shell. A window system is therefore unnecessary with present versions.

Memory requirements are fairly modest in most cases, as they do not depend on the size of the output images. 100MB is sufficient when co-adding images (even mosaics) involving current CCD chips ($2k\times4.5k$). More memory may be helpful for co-adding bigger maps. Although the built-in virtual-memory feature will almost always allow one to work with any image size, the performance hit caused by file-swapping may be important in some cases.

---

[4] `http://astromatic.net/software/swarp`
[5] `ftp://ftp.iap.fr/pub/from_users/bertin/swarp/`

4.3 Installation

To install, you must first uncompress and unarchive the archive:

```
gzip -dc swarp-x.x.tar.gz | tar xvf -
```

A new directory called `swarp-x.x` should now appear at the current position on your disk. You should then just enter the directory and follow the instructions in the file called "`INSTALL`".

The software is also available as a precompiled RPM for Linux systems with an x86 architecture. The simplest way to install an RPM package is to log as root and use the following command

```
rpm -U swarp-x.x-dist.arch.rpm
```

5   Using SWARP

SWARP is run from the shell with the following syntax:

% `swarp` *Input_image1* [*Input_image2* ...] [`@`*Image_list1* [`@`*Image_list2* ...]] `-c` *configuration-file* [-*Parameter1 Value1*] [-*Parameter2 Value2* ...]

The part enclosed within brackets is optional. The file names of input images can be directly provided in the command-line or in lists preceded with '@'. Lists are ASCII files containing the input file names (one per line). One should use lists instead of image file names if the number of input images is too large to be handled directly by the shell. Any "-*Parameter Value*" statement in the command-line overrides the corresponding definition in the configuration-file or any default value (see below).

5.1   The Configuration file

Each time SWARP is run, it looks for a configuration file. If no configuration file is specified in the command-line, it is assumed to be called "`default.swarp`" and to reside in the current directory. If no configuration file is found, SWarp will use its own internal default configuration.

5.1.1   Creating a configuration file

SWARP can generate an ASCII dump of its internal default configuration, using the "`-d`" option. By redirecting the standard output of SWARP to a file, one creates a configuration file that can easily be modified afterward:

% `swarp -d >default.swarp`

A more extensive dump with less commonly used parameters can be generated by using the "`-dd`" option.

5.1.2   Format of the configuration file

The format is ASCII. There must be only one parameter set per line, following the form:

Config-parameter   *Value(s)*

Extra spaces or linefeeds are ignored. Comments must begin with a "#" and end with a linefeed. Values can be of different types: strings (can be enclosed between double quotes), floats, integers,

keywords or boolean (`Y/y` or `N/n`). Some parameters accept zero or several values, which must then be separated by commas. Values separated by commas, spaces, tabs or linefeeds may also be read from an ASCII file if what is given is a filename preceded with `@` (e.g. `@values.txt`). Integers can be written as decimals, in octal form (preceded by digit `O`), or in hexadecimal (preceded by `0x`). The hexadecimal format is particularly convenient for writing multiplexed bit values such as binary masks. Environment variables, written as `$HOME` or `${HOME}` are expanded.

5.1.3 Configuration parameter list

Here is a list of all the parameters known to SWARP. Please refer to next section for a detailed description of their meaning. Some "advanced" parameters (indicated with an asterisk) are also listed. They must be used with caution, and may be rescoped or removed without notice in future versions.

BACK_DEFAULT   0.0   *floats* $(n \leq n_{\mathrm{ima}})$
    Default background value to be subtracted in `BACK_TYPE MANUAL` mode.

BACK_FILTERSIZE   3   *integers* $(n \leq n_{\mathrm{ima}})$
    Size (in background meshes) of the background-filtering mask.

BACK_FILTTHRESH   0.0   *integers* $(n \leq n_{\mathrm{ima}})$
    Difference threshold (in ADUs) for the background-filtering.

BACK_SIZE   128   *integers* $(n \leq n_{\mathrm{ima}})$
    Size (in pixels) of a background mesh.

BACK_TYPE   AUTO   *keywords* $(n \leq n_{\mathrm{ima}})$
    Tells SWARP what background is subtracted from the images:

AUTO	The internal interpolated background-map
MANUAL	A user-supplied constant value provided in `BACK_DEFAULT`

BLANK_BADPIXELS*   N   *boolean*
    If true, pixels with a weight of zero are set to zero in the combined image.

CELESTIAL_TYPE   NATIVE   *keyword*
    Celestial coordinate system in output:

NATIVE	Same as first input file
PIXEL	No (de-)projection (faster)
EQUATORIAL	Equatorial α, δ coordinates
GALACTIC	Galactic l, b coordinates
ECLIPTIC	Ecliptic λ, β coordinates

CENTER_TYPE   ALL   *keywords* $(n \leq n_{\mathrm{dim}})$
    Tells SWARP how to center the output frame:

ALL	Center on the region that contains all input fields
MOST	Center on the region with most overlap between input fields
MANUAL	Manual centering using the `CENTER` parameter

CENTER   0.0   *strings* $(n \leq n_{\mathrm{dim}})$
>   Position of the center in `CENTER_TYPE MANUAL` mode. Can be given in floating point nota-
>   tion, in *hh:mm:ss* (for right ascension/longitude), or *dd:mm:ss* (for declination/latitude).

COMBINE   Y   *boolean*
>   If true, resampled images will be combined.

COMBINE_BUFSIZE   256   *integer*
>   Maximum amount of buffer memory (in MB) used for the co-addition process.

COMBINE_TYPE   MEDIAN   *keyword*
>   Tells SWARP how to combine resampled images:
>
> | | |
> |---|---|
> | MEDIAN | Take the median of pixel values |
> | AVERAGE | Take the average |
> | MIN | Take the minimum |
> | MAX | Take the maximum |
> | WEIGHTED | Take the weighted average |
> | CHI2 | Take the weighted, quadratic sum |
> | SUM | Take the sum |

COPY_KEYWORDS   OBJECT   *strings* $(n \leq 1024)$
>   Coma-separated list of FITS keywords that will be propagated from the input FITS head-
>   ers to the coadded and resampled image headers.

DELETE_TMPFILES   Y   *boolean*
>   If true, resampled, temporary image files are deleted if `COMBINE` is set to `Y`

FSCALASTRO_TYPE   FIXED   *keyword*
>   Tells SWARP how to compute the astrometric part of the flux-rescaling:
>
> | | |
> |---|---|
> | NONE | Ignore the effects of re-projection |
> | FIXED | Apply a constant rescaling fluxes based on the ra-tio of pixel scales at the field's geometrical center |
> | VARIABLE | Apply a rescaling of fluxes based on the local ratio of pixel scales (variable throughout the image) |

FSCALE_DEFAULT   1.0   *floats* $(n \leq n_{\mathrm{ima}})$
>   Default flux scale to adopt for each image if the `FSCALE_KEYWORD` keyword is not found in
>   the FITS header

FSCALE_KEYWORD   FLXSCALE   *string*
>   FITS keyword that should contain the flux scale factor in input images.

GAIN_DEFAULT   0.0   *floats* $(n \leq n_{\mathrm{ima}})$
>   Default gain (conversion factor in e$^-$/ADU) to adopt for each image if the `GAIN_KEYWORD`
>   keyword is not found in the FITS header. 0 means "infinite" gain.

GAIN_KEYWORD   GAIN   *string*
>   FITS keyword that should contain the gain in input images.

HEADER_ONLY   N   *boolean*

If true, SWARP does not do anything besides creating the FITS header in the combined image. This header can later be duplicated as `.head` files to provide an identical target frame on several machines.

HEADER_SUFFIX   .head   *string*
         Filename extension for external ASCII "headers" that override internal FITS parameters.

IMAGEOUT_NAME   coadd.fits   *string*
         Name of the output image file.

IMAGE_SIZE   0   *integers* $(n \leq n_{\mathrm{dim}})$
         Dimensions of the output image (in `PIXELSCALE_TYPE MANUAL` or `FIT` mode). `0` means "automatic".

MEM_MAX   256   *integer*
         Maximum amount of megabytes allowed for storing input images in memory.

NTHREADS   0   *integer*
         Number of threads (processes) allowed to run simultaneously during the resampling and combination phases. `0` automatically sets one thread per CPU core.

OVERSAMPLING   0   *integers* $(n \leq n_{\mathrm{dim}})$
         Amount of oversampling in each dimension. `0` means "automatic".

PIXEL_SCALE   0.0   *floats* $(n \leq n_{\mathrm{dim}})$
         Step between pixels in each dimension (in `PIXELSCALE_TYPE MANUAL` mode). Must be expressed in arcseconds for angular coordinates.

PIXELSCALE_TYPE   MEDIAN   *keywords* $(n \leq n_{\mathrm{dim}})$
         Tells SWARP how to adjust the output pixel size:

MEDIAN	Take the median of pixel scales at the center of input frames
MIN	Take the minimum of pixel scales at the center of input frames
MAX	Take the maximum of pixel scales at the center of input frames
MANUAL	User-defined pixel scale at image center (with the `PIXEL_SCALE` keyword)
FIT	Sets the pixel scale such as the full output field fits the user-defined `IMAGE_SIZE`

PROJECTION_ERR   0.001   *floats* $(n \leq n_{\mathrm{ima}})$
         Maximum position error (in pixels) allowed for bicubic-spline interpolation of the astrometric reprojection. `0` turns off interpolation.

PROJECTION_TYPE   TAN   *string*
         Projection system used in output, in standard WCS notation (see Table 2).

RESAMPLE   Y   *boolean*
         If true, resampling is performed on the input images.

RESAMPLE_DIR   .   *string*

Path of the directory where resampled images are written.

RESAMPLE_SUFFIX   `.resamp.fits`   *string*
    Filename extension given to resampled images produced by SWarp.

RESAMPLING_TYPE   `LANCZOS3`   *keywords ($n \le n_{\rm dim}$*

    Image resampling method:

	`NEAREST`	Take the nearest neighbour
	`BILINEAR`	Bi-linear interpolation
	`LANCZOS2`	Lanczos-2 4×4-tap filter
	`LANCZOS3`	Lanczos-3 6×6-tap filter
	`LANCZOS4`	Lanczos-4 8×8-tap filter

RESCALE_WEIGHTS   `Y`   *booleans ($n \le n_{\rm ima}$)*
    Tells SWarp whether input weight maps and variance maps should be automatically
    rescaled or not.

SATLEV_KEYWORD*   `SATURATE`   *string*
    FITS keyword that contains the saturation level (in ADUs).

SATLEV_DEFAULT*   `50000`   *floats ($n \le n_{\rm ima}$)*
    Default saturation level (in ADUs) set for images for which no **SATLEV_KEYWORD** FITS
    header keyword is found.

SUBTRACT_BACK   `Y`   *booleans ($n \le n_{\rm ima}$)*
    If true, input images are background-subtracted prior to resampling.

VMEM_DIR   `/tmp`   *string*
    Path of the directory where virtual-memory and other temporary files are written.

VMEM_MAX   `2047`   *integer*
    Maximum amount of megabytes allowed for virtual-memory storage.

VERBOSE_TYPE   `NORMAL`   *keyword*
    Tells SWarp how well operations should be commented:

	`QUIET`	Run silently
	`LOG`	Log essential information
	`NORMAL`	Display information dynamically (using control characters)
	`FULL`	Display more complete information

WEIGHT_IMAGE   *strings ($n \le n_{\rm ima}$)*
    List of input weight-map filenames

WEIGHTOUT_NAME   `coadd.fits`   *string*
    File name of the output weight-map.

WEIGHT_THRESH*   *floats ($n \le n_{\rm ima}$)*
    Threshold below or above which input weights are defined equivalent to zero (infinite
    variance, i.e. a bad pixel).

`WEIGHT_TYPE`	`NONE`	*keywords* $(n \leq n_{\mathrm{ima}})$

Sets the type of input weight-maps:

`NONE`	No weighting
`MAP_WEIGHT`	Relative weights (i.e. inverse variance)
`MAP_VARIANCE`	Relative variance
`MAP_RMS`	Absolute standard deviation

`WRITE_FILEINFO`	`N`	*boolean*

If true, extended information about input files is written in the header of the output FITS image.

`WRITE_XML`	`Y`	*boolean*

If true, meta-data are written in XML-VOTable format.

`XML_NAME`	`swarp.xml`	*string*

File name for the XML VOTable output of SWARP. Use "`STDOUT`" for piping to standard output.

`XSL_NAME`*	`.`	*string*

URL of an XSL style-sheet for the XML output of SWARP. The URL will appear in the `href` attribute of the `style-sheet` tag.

6 How SWARP works

6.1 Overview of the software

What SWARP does is basically to read a set of input FITS images, resample and combine them, and finally save the resultant FITS image to disk. The work can be decomposed in several steps:

1. Input image headers are read and checked for content. If configured in fully automatic mode, SWARP will set the characteristics of the output frame based on this information.

2. Input images (and their weight-maps, if available) are read one-by-one. Background-maps are built, and subtracted from the images if required.

3. Images are resampled, projected into subsections of the output frame, and saved as FITS files. "Projected" weight-maps are created too, even if no weight-maps were given in input.

4. A combined output image is created using the information stored in the "projected" weight-maps. It consists of a composite of the resampled sub-sections. A composite output weight-map is also written in the process.

The global layout of SWARP is presented in Fig. 1. Let us now describe each of the important steps.

6.2 Image mapping and memory constraints

How does SWARP projects input images into the output frame space? There are two ways of applying a geometric transformation to an image (see Wolberg 1992). The most intuitive is
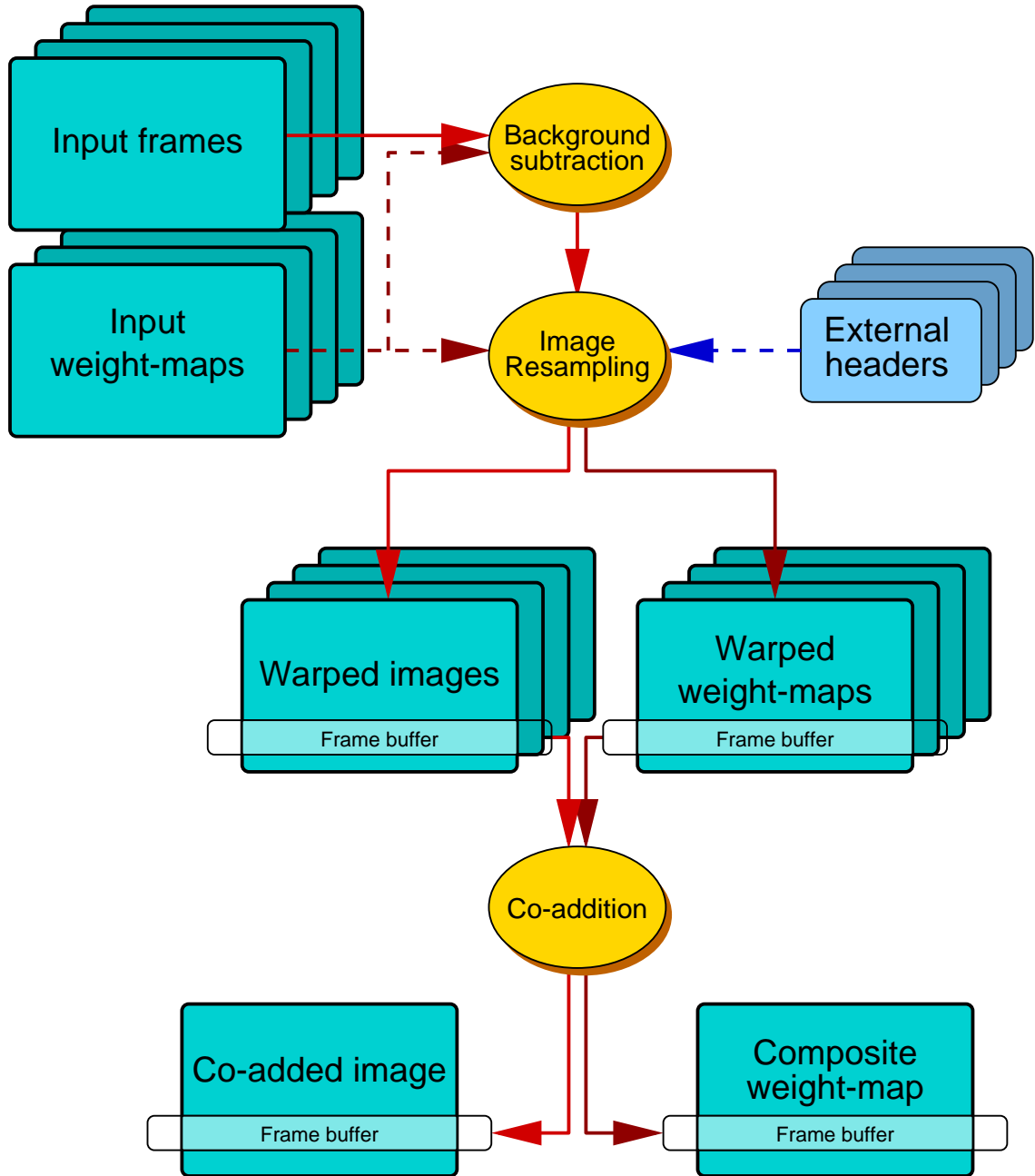
Figure 1: Global Layout of SWarp.

called "forward mapping". It consists in scanning the input image pixel-per-pixel, line-by-line. Each pixel is simply "thrown" to the position it is supposed to occupy in the output grid. Although this technique can be used for geometric resampling or "drizzling" (Fruchter & Hook 1997), it is totally cumbersome with high order interpolation techniques. "Inverse mapping" is far more efficient in this case. In this procedure the output frame is scanned pixel-per-pixel and line-by-line. Using the inverse projection, each output pixel center is associated a position in the input frame, at which the image is interpolated. This technique has been implemented in SWARP (Fig. 2); it possesses several advantages. The output image is accessed sequentially, and thus can be arbitrarily large. Also, only positions corresponding to pixels (or sub-pixels) within the output frame have to be mapped.
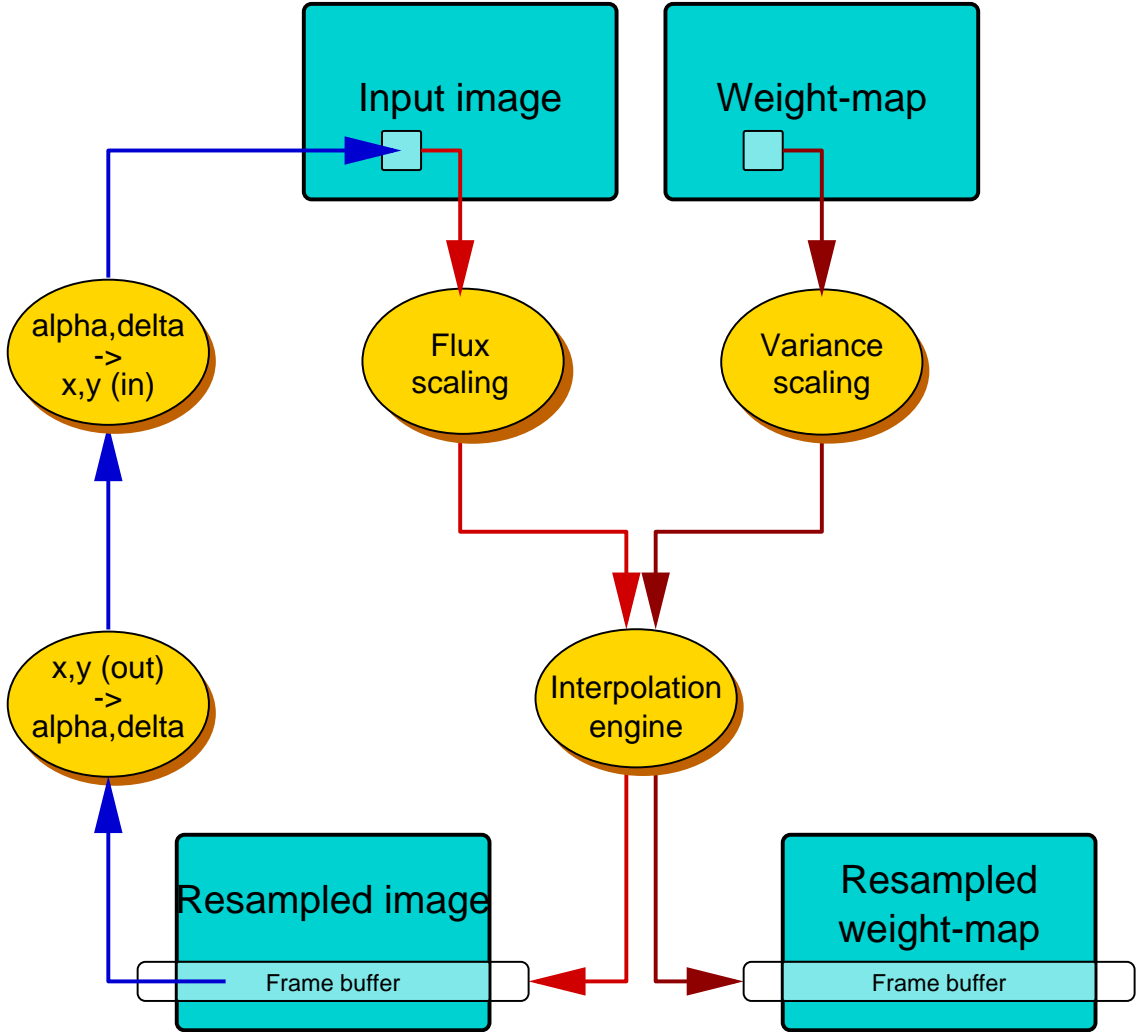
Figure 2: Layout of the image mapping section.

The most potentially critical part is the pseudo-random access in the input image. In most cases, it will be an individual imaging array (like an individual CCD) and will therefore fit in memory. For much larger input images however we rely on the efficiency of virtual-memory-mapping. SWarp's virtual memory engine works in the following way: each input image, stored as a single precision, 4-byte array is loaded in physical memory if the required amount of megabytes doesn't exceed MEM_MAX. If it does, a temporary file called *vmxxxxx_xxxxx.tmp* is written in the directory specified by VMEM_DIRECTORY. The program exits with an error message if this file would exceed VMEM_MAX Megabytes. The 3 "memory" parameters are mostly hardware-dependent. It is

advised to set MEM_MAX to 50-100% of the actual amount of memory present in your machine. If disk-space is not the limiting factor, VMEM_MAX should be set to a higher value, 2048 on a 32-bit machine, or even more on a 64-bit machine. The choice of the VMEM_DIRECTORY is critical. First, this write-enabled directory must be large enough to contain each input image in floating-point format. Second, it is strongly recommended to have the data on a fast disk. Note that the default path for VMEM_DIRECTORY is */tmp*, which on many systems is on a partition simply not large enough to handle the typical quantities of data to process. An alternative is to use ".", the current directory, at the expense of disk thrashing!

De-projecting and co-adding simultaneously all input images would frequently imply many files open at the same time, and large amounts of (virtual) memory. SWARP takes a more sequential approach: each input image is mapped in a tightly-fitting rectangular subsection of the output frame. All subsections are written to disk (in the RESAMPLE_DIRECTORY) as *swarp.xxx.fits* FITS files, and read back later during the co-addition phase, to be stacked together. Individual subsection files are automatically removed after processing or abortion. It is possible to disable the deletion by setting the DELETE_TMPFILES configuration parameter to N (the default is Y). This can be useful for diagnostic purposes.

Note that although SWARP does not use much memory, the amount of temporary disk-space needed during processing can be quite large. In addition to the output image and weight-map, one should provide disk-space for the individual projected images and their weight-maps. In the case of mappings done at unit scale, this involves storing more than twice the amount of input pixels as temporary data.

6.3   Propagating FITS keywords

During the re-gridding and co-addition processes involved in SWARP, not all FITS keywords present in the input image headers are automatically copied to the output image headers, as many of them become irrelevant. Table 1 lists the non-essential keywords that can be found in the output FITS header after being propagated/updated by SWARP.

Table 1: Non-essential FITS keywords managed by SWARP

Name	Output value
EXPTIME	Sum of exposure times in the part of the coadd with the most overlaps
GAIN	Effective gain (flux and weight scaling apply, see §6.9.2)
MJD-OBS	Modified Julian day of the earliest start of exposures
SATURATE	Minimum of all input saturation levels (flux scaling applies)

For data management purposes it is often useful to propagate other selected FITS keywords (such as FILTER or TELESCOP for instance) and their values from the input image headers to the resampled and coadded image headers. To this aim, a COPY_KEYWORDS configuration parameter is provided. It accepts a list of FITS keywords that shall be copied in the headers of all the images created by SWARP; by default, only the OBJECT keyword and its content are copied. Because the coadded image can result from the combination of many input files, only the keyword found in the first image header from the input file list is propagated up to the final coadded image. It is important to note that SWARP does not check the content of the list of COPY_KEYWORDS; therefore one should be cautious not to propagate "structural" FITS keywords like NAXIS1,BITPIX,... that may interfere with the interpretation of the output data.

6.4   Parallel processing

Versions $\geq$ 1.32 of SWarp are "multi-threaded". Multi-threading allows CPU-intensive tasks in SWarp to be run in parallel on multi-core or Hyper-Threaded (HT) machines. Best performance is generally achieved when the number of active threads is equal to the number of processor cores in the machine (Fig. 3). This is automatically done with the default settings (`NTHREADS = 0`). However, if other cpu-intensive tasks have to run on the same machine at the same time, it can be useful to set `NTHREADS` manually, to have SWarp using a lower number of threads.
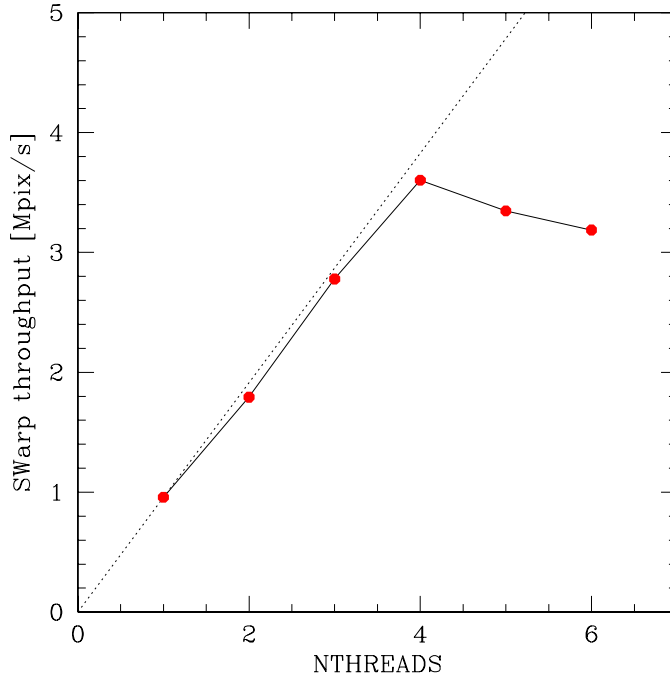
Figure 3: Pixel throughput of SWarp 2.0 resampling (Lanczos3) as a function of the number of threads on an SMP machine with 4 Opteron-242 (1.6GHz) processors. Departure from perfect linear scaling (indicated by the dashed line for reference) is due to I/O limitations and parts of code that are not multithreaded.

6.5   Astrometry

The astrometric engine at the heart of SWarp is based on M. Calabretta's *WCSlib* library[6], to which we added the handling of polynomial distorsion parameters (FITS keywords `PV_xx_xx`) as proposed in the latest WCS documents[7]. We included IRAF's `TNX` astrometric projection too (for inputs only), although it is not part of the WCS standard.

All celestial coordinate computations are performed in the equatorial system. Galactic or ecliptic coordinates are supported in input and output.

---

[6]Available at `http://www.cv.nrao.edu/fits/src/wcs/`
[7]`http://www.cv.nrao.edu/fits/documents/wcs/wcs.html`

6.5.1 Input frames

(De-)projection parameters of input images are extracted from their respective FITS headers. These are the usual `CTYPEx CRVALx`, `CRPIXx`, `CDELTx` and/or `CD_xx_xx` WCS parameters. External "header" files can also be provided by the user; for every input *xxxx.fits* image, SWARP looks for a *xxxx.head* header file, and loads it if present. A *.head* suffix is the default; it can be changed using the `HEADER_SUFFIX` configuration parameter. External headers may either be real FITS header cards (no carriage-return), or ASCII files containing lines in FITS-like format, with the final line starting with "END⎵⎵⎵⎵⎵". Multiple extensions in ASCII files must be separated by an "END⎵⎵⎵⎵⎵" line; there should not be any primary header in that case. External "headers" need not contain all the FITS keywords normally required. The keywords present in external headers are only there to override their counterparts in the original image headers, or add new ones.

With SWARP it is possible to process a single extension from an input Multi-Extension FITS (MEF) file, be it an image or a weight-map. For instance, to process only the 3rd extension from the MEF file called `mef.fits`, one should use `mef.fits[2]`. Note that in command lines, the `[` and `]` characters must be escaped with `\`, or the complete name put between double quotes, to avoid expansion by the shell.

6.5.2 Output frames

The celestial pair of components of the output coordinate system is specified with the `CELESTIAL_TYPE` configuration parameter, and can be selected among `NATIVE`, `PIXEL`, `EQUATORIAL`, `GALACTIC` and `ECLIPTIC`. In `NATIVE` mode, the output celestial coordinate system is taken from that of the *first* file of the input list. This is the default. The `PIXEL` option forces SWARP to ignore all the "celestial" aspects (projection, de-projection, sky coordinates) of both input and output images. It provides a major speed-up to the warping engine by bypassing all the trigonometric operations normally involved in the other modes. It is useful for quickly combining mosaic images whenever astrometric information is not needed. In `PIXEL` mode, degrees are interpreted as dimensionless Cartesian coordinates.

The output (celestial) projection is set by the `PROJECTION_TYPE` configuration parameter. The list of all presently supported projections is shown in Table 2, and illustrated in Fig. 4 and 5 using a gridded map of the Earth.

Now, what projection is the best? With "small" fields (< 10 degrees in their maximum dimension), the choice is not critical as long as the projection center lies within the frame. For compatibility reasons, it is advised to stay with the traditional gnomonic (`TAN`, for tangential) projection in such cases. With larger fields the pure tangential projection is inappropriate, and one is faced with the usual problems confronted by cartographers. It would be outside the scope of this document to discuss the merits of each projection. For detailed information about the different projection systems, the user should refer to the latest WCS document. Let us just mention that equal-area projections (those that conserve relative areas) are often to be preferred for mapping large sky surveys, because they naturally conserve surface-brightness and/or allow summing pixel values to measure fluxes. The following are equal-area projections: `ZEA`, `CEA`, `COE`, `BON`, `GLS`, `PAR`, `MOL`, `AIT`, `QSC`. `AIT` (Aitoff) is one of the most popular projections for all-sky maps.

Note that some of the projections (`CYP`, `CEA`, `COD`, `COE`, `COO`, `COP` and `BON`) *require* additional `PV_xx_xx` parameters. These parameters can easily be included in a *xxxx.head* header file with the same prefix as the output coadded image (which is *coadd.fits* by default, see the example at the end of this document).

13

Table 2: Valid `PROJECTION_TYPE`s in SWarp

Zenithal projections

AZP	Zenithal perspective
TAN	Distorted tangential
STG	Stereographic
SIN	Slant orthographic
ARC	Zenithal equidistant
ZPN	Zenithal polynomial
ZEA	Zenithal equal-area
AIR	Airy

Cylindrical projections

CYP	Cylindrical perspective
CEA	Cylindrical equal-area
CAR	Plate carrée
MER	Mercator

Conic projections

COP	Conic perspective
COE	Conic equal-area
COD	Conic equidistant
COO	Conic orthomorphic

Pseudoconic and polyconic projections

BON	Bonne's equal-area
PCO	Polyconic

Pseudocylindrical projections

GLS	Global sinusoidal (Sanson-Flamsteed)
PAR	Parabolic
MOL	Mollweide
AIT	Hammer-Aitoff

Quad-cube projections

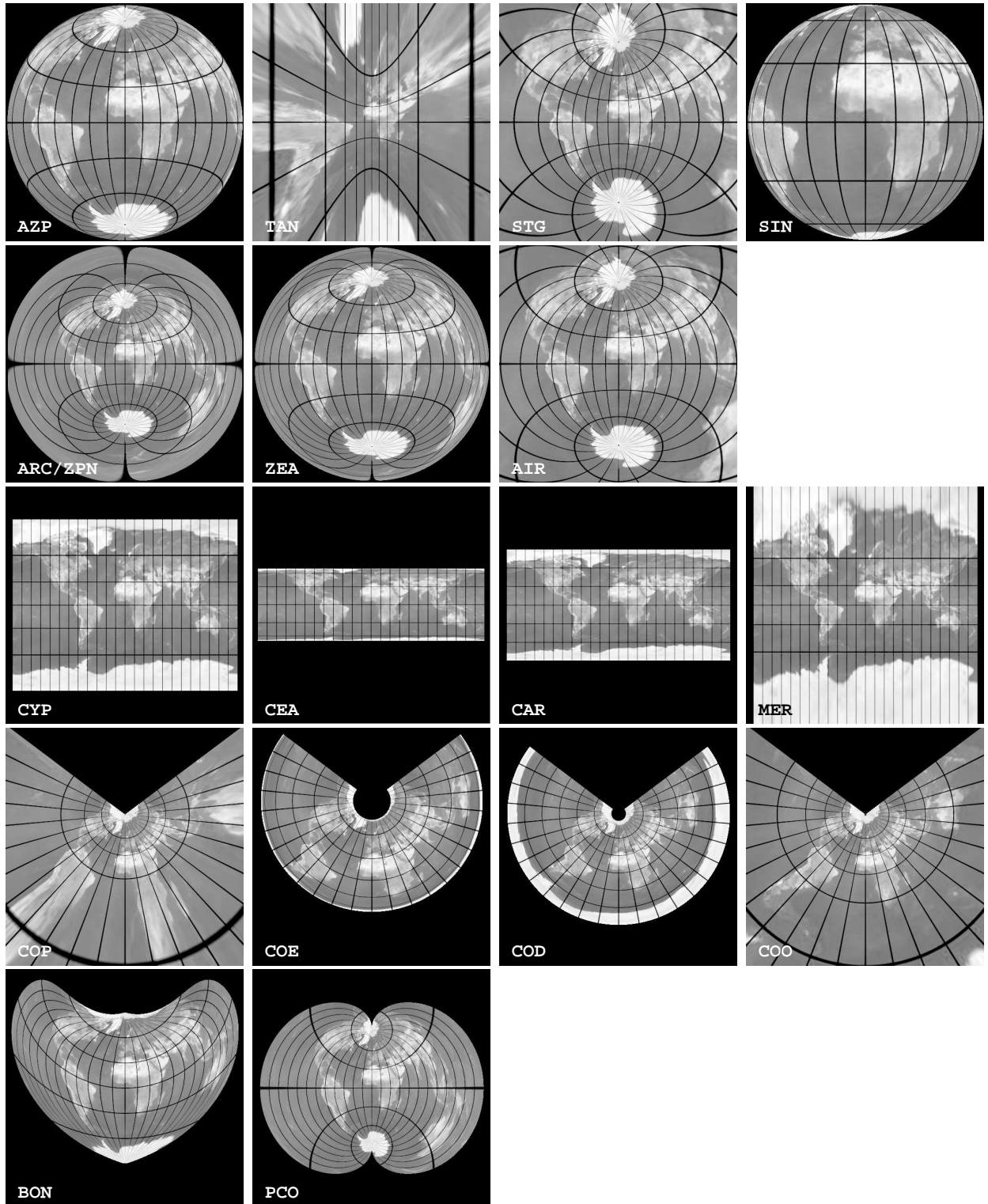|	
TSC	Tangential spherical cube
CSC	COBE quadrilateralized spherical cube
QSC	Quadrilateralized spherical cube

Figure 4: Graphic illustration of projections available in the WCS library (see text).
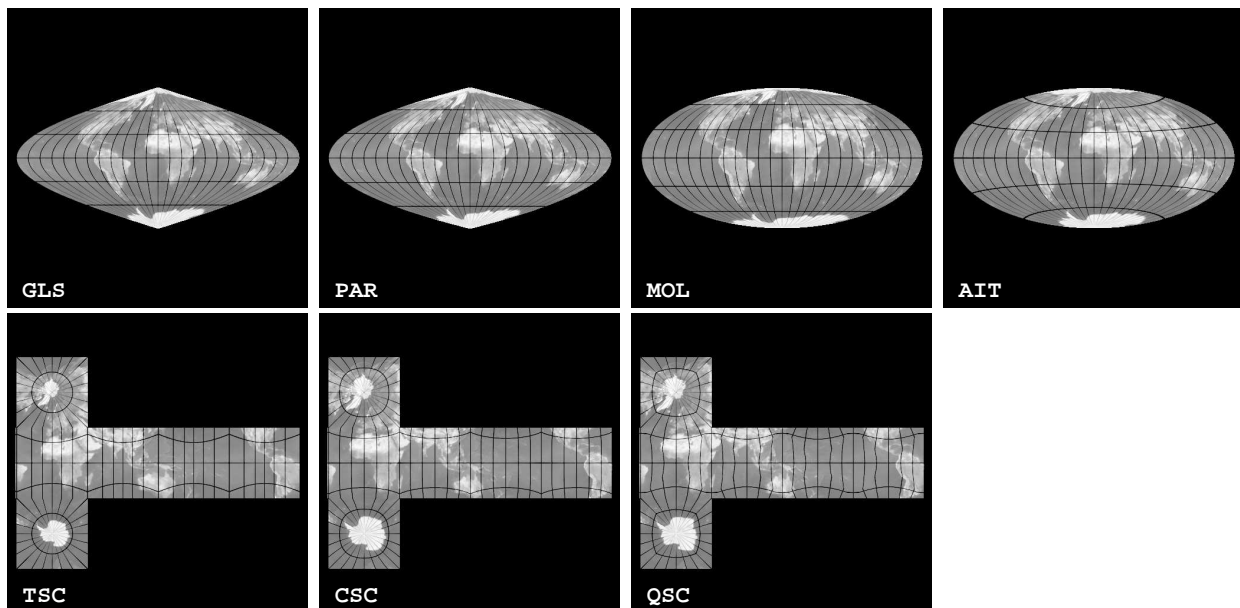
Figure 5: Graphic illustration of projections available in the WCS library (continued from Fig. 4).

Centering of the output frame is controlled by the CENTER_TYPE parameter. There are three centering modes:

- ALL: the field is centered in a way that all input images fit into the output frame. This is the default.

- MOST: the field is centered on the zone of maximum overlap between input images.

- MANUAL: manual centering with the CENTER parameter.

A different centering mode can be used in each dimension; for instance, in 2D images with α, δ coordinates, "CENTER_TYPE ALL,MOST" will apply the ALL mode in α and the MOST mode in δ. If a single mode is specified, it is applied to all available dimensions.

The CENTER parameters are active in CENTER_TYPE MANUAL mode only, and must be used to specify the actual center of the output field, in world units. In the case of angular coordinates, both the floating point (in degrees) and sexagedecimal formats are accepted: right ascension/longitude may be written as *hh:mm:ss.ss*, and declination/latitude as ±*dd:mm:ss.ss*.

The pixel "scale" (which is the step between pixels at the center of the output frame) can be computed automatically in each dimension by SWARP. There are five modes specified by the PIXELSCALE_TYPE configuration parameter:

- MEDIAN (the default): the median value of all pixel scales at the center of input frames is taken as the output pixel scale.

- MIN: the smallest of all pixel scales at the center of input frames is taken.

- MAX: the largest of all pixel scales at the center of input frames is taken.

- MANUAL: manual scaling with the PIXEL_SCALE configuration parameter.

- **FIT**: Pixel scales are automatically computed to have the projected data fitting the output frame dimensions specified with the `IMAGE_SIZE` configuration parameter.

When right ascension/longitude and declination/latitude are both present, pixel scales computed by SWARP are made equal in both dimensions to avoid anamorphosis.

The `PIXEL_SCALE` parameters are active in `PIXELSCALE_TYPE MANUAL` mode only, and must be used to specify the actual pixel step for each dimension, in world units. Note that in the case of angular coordinates, `PIXEL_SCALE` values are read in arcseconds, not degrees.

The dimensions of the output frame (in number of pixels per axis) are set using the `IMAGE_SIZE` configuration parameter. A value of 0 for any axis results in an automatic dimensioning of this axis; but obviously this is not possible in `PIXELSCALE_TYPE FIT` mode.

Note that the current simple algorithms used for automatic centering and scaling routines can get confused rather easily close to the pole or with some very wide field projections. In particular, it is recommended to turn off automatic settings when making all-sky projections.

6.5.3   Bi-cubic spline interpolation

The trigonometric calculations involved in SWARP re-projections have a major impact on processing speed. To accelerate the resampling phase, versions ≥ 2.06 of SWARP implement a bi-cubic spline interpolation[8] of the astrometric mapping between the input and the output frames. Interpolation is used by default for large images, with a *maximum* allowed positional error of 10^{-3} pixel (as measured in the output frame). This error tolerance can be changed independently for each input image with the `PROJECTION_ERR` configuration parameter. A `PROJECTION_ERR` of 0 turns interpolation off. Interpolation is also automatically deactivated for smaller images or inappropriate mappings like all-sky projections or projection with singularities.

6.6   Resampling

The action of projecting a grid of pixels on another grid is called resampling. Ideal image resampling involves both filtering and interpolation between pixels. In SWARP, filtering is "naturally" implemented by oversampling the destination grid. Because SWARP uses reverse-mapping, interpolation is made on the input images.

6.6.1   Image data

At each position $\mathbf{x}$, the dot-product between a local kernel $\mathbf{k}(\mathbf{x})$ and neighbouring pixel values $\mathbf{f}$ yields a local, interpolated value:

$$\tilde{f}(\mathbf{x}) = \mathbf{k}(\mathbf{x}).\mathbf{f} \tag{1}$$

The kernel is derived locally from an *interpolation function h*:

$$k_i(\mathbf{x}) = h(\mathbf{x} - \mathbf{x}_i) \tag{2}$$

The `RESAMPLING_TYPE` configuration parameter allows the user to choose among several symmetric, compact interpolation functions:

---

[8]This feature is currently limited to 2D images.

- NEAREST: a "square box" response function, with width 1 pixel. Applying this function produces *nearest-neighbour interpolation* (also known as *sample-and-hold*). The kernel extends over a single input pixel.

- BILINEAR: a pyramidal response function with Full-Width at Half Maximum 1 pixel. This results in a *bilinear interpolation*. The kernel extends over 2^{n_dim} pixels.

- LANCZOS2: a $\prod_d \mathrm{sinc}(\pi x_d)\mathrm{sinc}(\frac{\pi}{2}x_d)$ response function with $(-2 < x_d \leq 2)$ (Lanczos2 window). The kernel extends over 4^{n_dim} pixels.

- LANCZOS3: a $\prod_d \mathrm{sinc}(\pi x_d)\mathrm{sinc}(\frac{\pi}{3}x_d)$ response function with $(-3 < x_d \leq 3)$ (Lanczos3 window). The kernel extends over 6^{n_dim} pixels.

- LANCZOS4: a $\prod_d \mathrm{sinc}(\pi x_d)\mathrm{sinc}(\frac{\pi}{4}x_d)$ response function with $(-4 < x_d \leq 4)$ (Lanczos4 window). The kernel extends over 8^{n_dim} pixels.

As demonstrated in Fig. 6, the Lanczos4 interpolation function provides the best resampling for correctly sampled data. In theory one could use an even larger kernel to get a closer-to-perfect resampling. However in practice large kernels with a sharply-limited bandpass carry more problems than advantages. Artifacts, image borders or undersampled data generate extended ripples (*Gibbs' phenomenon*). These ripples are obvious on the saturation trail and the cosmic ray impact of the Lanczos interpolations in Fig. 6. In addition, the computational cost becomes prohibitive with multi-dimensional data.

Nearest-neighbour interpolation provides a good conservation of the noise spectrum at scales close to unity; unfortunately, it generates a terrible aliasing when zooming in, and can distort a lot object shapes at places. Its usage should therefore be restricted to images such as flag- or weight-maps.

Bilinear interpolation is fast and doesn't generate negative artifacts. However, it creates a lot of smoothing by correlating the values of neighbour pixels. On images with white noise, this may lead to obvious "moiré" effects (Fig. 7). Nevertheless, bilinear interpolation can be useful for processing undersampled data.

In general, Lanczos3 resampling represents the best compromise. As can be seen in Fig. 8, it brings a substantial benefit over bilinear interpolation in preserving the signal, while creating relatively modest artifacts around image discontinuities.

6.6.2 Oversampling

Unfortunately the procedure described above generates aliasing when "zooming out" sufficiently an image by resampling it at a lower resolution. Moreover, the intensity of the resampled background white noise stays constant instead of being proportional to the zooming factor (see Fig. 10). This is because the algorithm essentially *decimates* the data instead of binning them within the output pixel footprint: a similar effect applies in the panning windows of astronomical visualization tools, for instance. This problem can be approximately solved by dilating appropriately the interpolation kernel, or by pre-filtering the input image (like textures in 3D hardware). A more exact and more efficient solution is to *oversample* the output pixel grid (Fig. 10), in order to obtain a density of samples per unit area (or hypervolume) at least equal to that of the input image.

Oversampling is controlled by the OVERSAMPLING configuration parameter. If OVERSAMPLING is set to 1, no oversampling is applied. An OVERSAMPLING of 2 oversamples the data by 2^{n_dim}
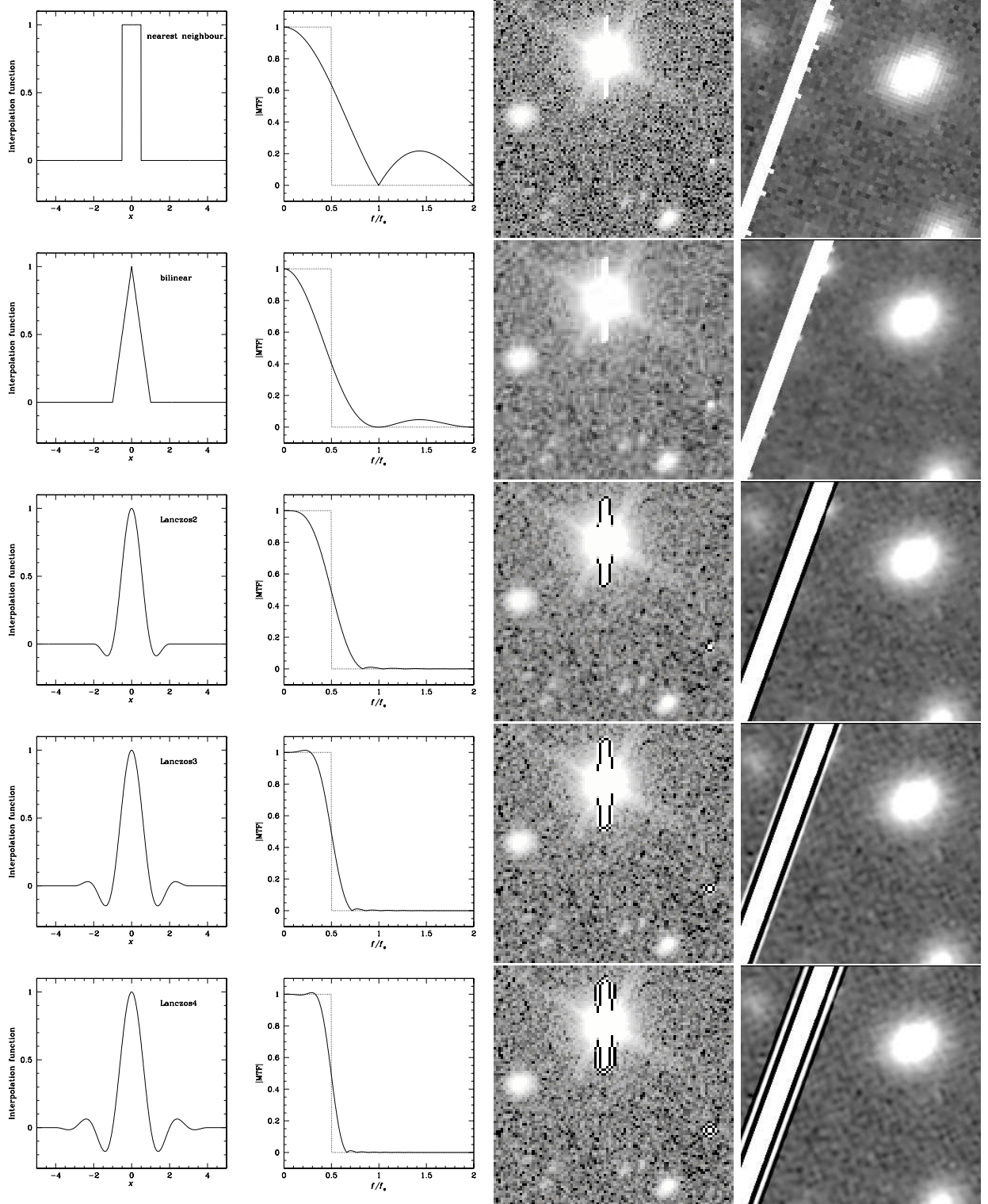
Figure 6: Comparison between resampling methods. *From top to bottom*: nearest-neighbour, bilinear, Lanczos2, Lanczos3, Lanczos4. *From left to right*: Interpolation function profile, Modulation Transfer Function, result from shifting an image by half a pixel in both direction, and result for a 5× zoom + rotation by 20 degrees.
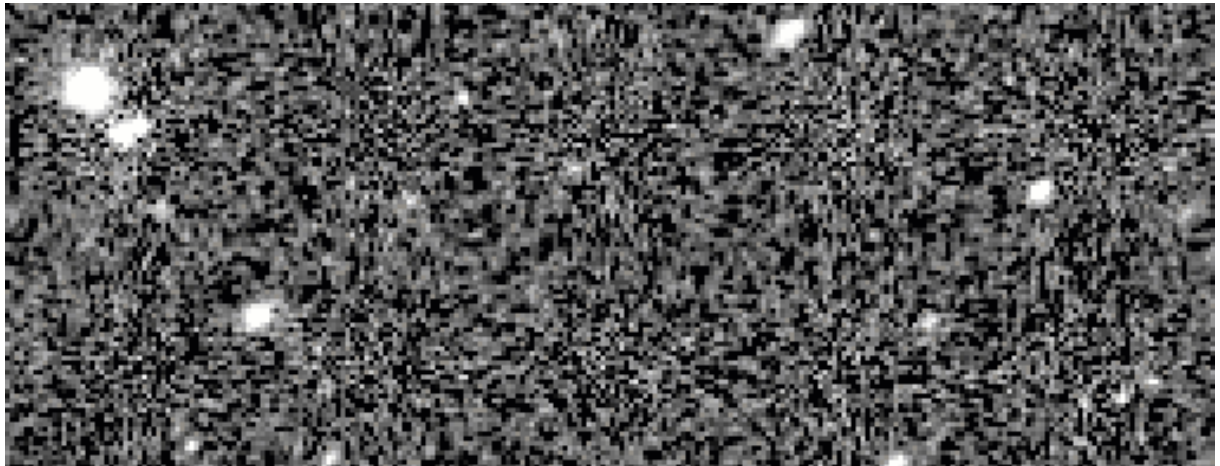
19

Figure 7: Example of moiré pattern on the background noise, generated by bilinearly resampling an image containing white noise at a pixel scale slightly different from 1.

samples per pixel, and so on. Oversampling can be different in each dimension: `OVERSAMPLING` 2,3 will oversample each pixel in a 2×3 grid for instance. An `OVERSAMPLING` of 0 (the default) lets SWARP select automatically the most appropriate oversampling factor in each dimension, by comparing pixel scales at the reference point. Although it works fairly well in many cases, situations where the pixel scale varies a lot over the image — like in all-sky projections — are not yet properly handled, and manual setting should then be prefered.

Note that oversampling considerably slows down the processing; `OVERSAMPLING` values should therefore be selected with some caution!

6.6.3 Noise stability issues

So far we have ignored the influence of noise variations in the resampling process. In theory, the interpolation schemes described above apply only if the noise is stationary (in the wide sense) over the extent of the interpolation kernel. Artifacts aside, this can be considered as true for the background noise, since the weight-maps are reasonably stable at the interpolation scale. However, the photon noise associated with the sources themselves may vary strongly over the scale of the PSF FWHM. As most astronomical images are barely oversampled, the hypothesis of noise stationarity breaks down on bright point sources for which intrinsic photon noises dominates[9] In the most severe cases, resampled noise peaks may generate distorsions in the resampled profiles.

Low-background-noise simulations were conducted in order to evaluate the amplitude of these distorsions on correctly-sampled data (PSF FWHM = 3 pixels). The effect is small, although not totally negligible on sources with intermediate intensity. On profile-fitting measurements for instance, photometry can be affected at the level of a few millimag *rms*. The degradation of astrometric precision was found not to exceed a few millipixels *rms*. On typical background-noise limited images the effects are even smaller.

---

[9]It is possible to stabilize the noise variance using a non-linear dynamic scale transform (Anscombe 1948, see also Stark *et al.* 1998). The transformed signal is still bandpass-limited but unfortunately, resampling and transforming it back biases significantly the data.
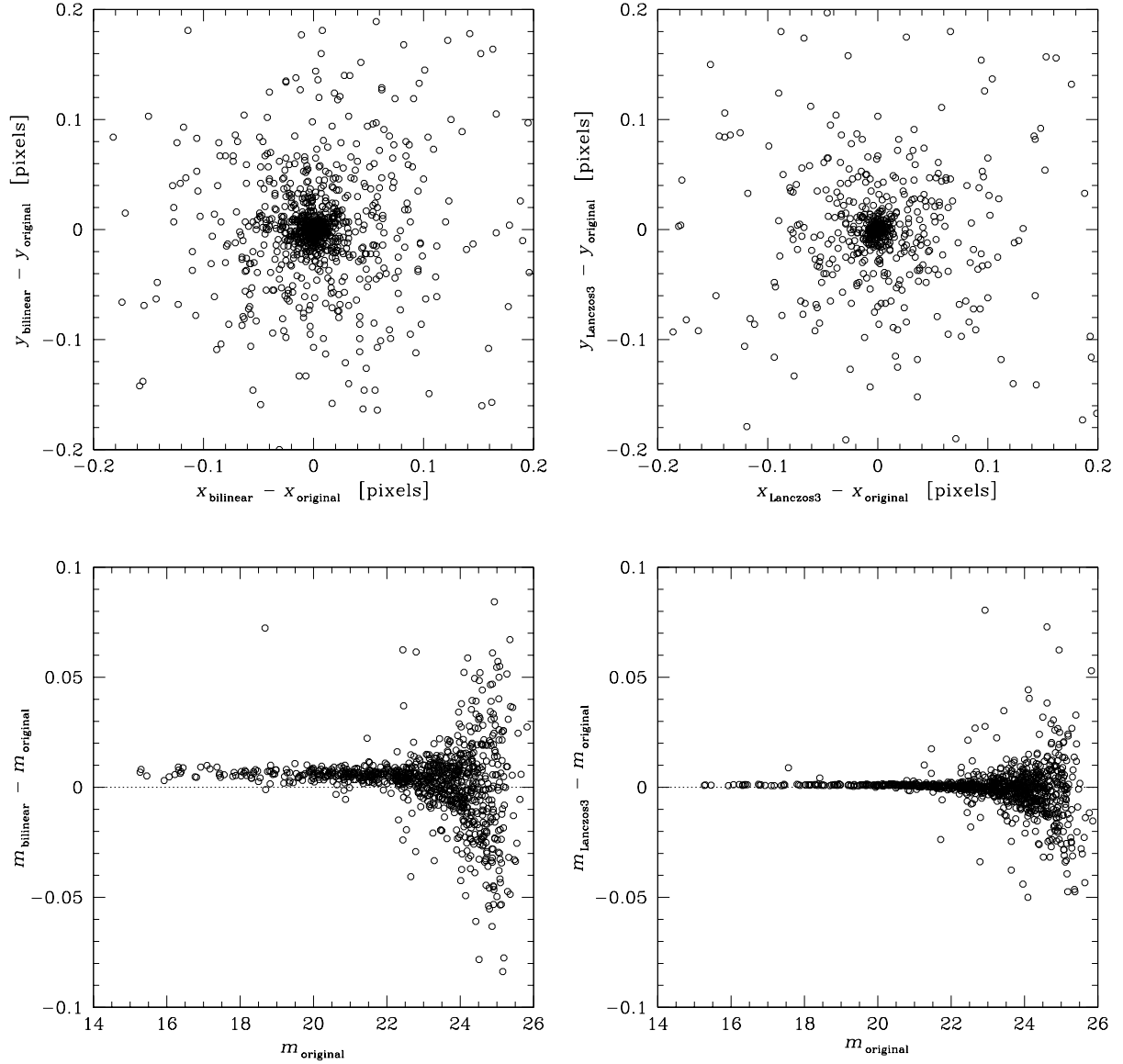
Figure 8: Effects of the resampling on position (*top*) and flux (*bottom*) measurements. *Left*: bilinear interpolation; *right*: Lanczos3 interpolation. In both cases, a simulated deep sky image with 0.7" seeing, containing stars and white background noise, was rotated by 20 degrees and then rotated back to match the original image. Fluxes were measured in a fixed 2" aperture. The dispersions seen here reflect the differences between *measurements* on the original and resampled images. These dispersions are much smaller than what one would observe by comparing the measurements on the resampled images with the theoretical (noise-free) positions or fluxes of the simulation. Note however the significant magnitude offset and flux dispersion in the bilinear case, consequences of the stronger smoothing induced by bilinear interpolation.
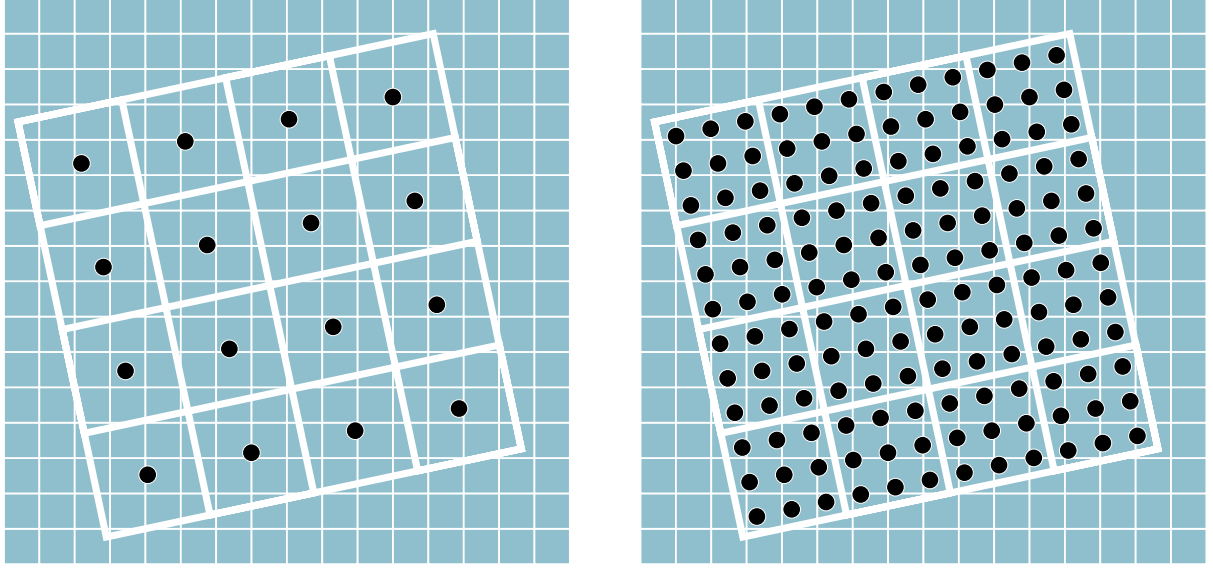
Figure 9: Oversampling in SWARP. The input grid is shown as small grey squares, whereas the output grid (resampled image) is represented by the large tilted ones. *Left*: Without oversampling, only one interpolation (dark spot) of the input image is done at the center of each output pixel. *Right*: with oversampling, several interpolated samples are obtained on a regular subgrid, and then binned in each output pixel. Here a 3×3 oversampling is sufficient.

Figure 10: The effect of oversampling in SWARP. A deep, real image with a 0.2" pixel scale and 0.8" seeing FWHM is resampled at 1" resolution. *Left*: no oversampling. *Right*: with 5×5 oversampling. Note the lower noise level and higher depth in the right image.

6.6.4 Weight-maps

The processing of the weight-maps (see §6.9.2) follows that of the data images, except that one is dealing now with variances instead of fluxes. The resampled weight at position $\mathbf{x}$ may be written as

$$\tilde{w}(\mathbf{x}) = \frac{1}{\sum_i \frac{k_i^2(\mathbf{x})}{w_i}} \tag{3}$$

Therefore when an input weight within the range of the interpolation function is zero, the interpolated weight is also zero. The general consequence is that the borders of interpolated images are trimmed by half the range of the interpolation function. Similarly, small "holes" in a provided weight-map are dilated by the interpolation function footprint. For example, once interpolated with a Lanczos4 kernel, a single, isolated zero-weight pixel will yield a clump of about 64 pixels in the resampled image! This is another illustration of the disadvantages in using large interpolation kernels.

6.7 Background subtraction

The flux at each pixel is a function of the sum of a "background" signal and light coming from the objects of interest. At most wavelengths, the strongest contribution to the background is of instrumental/atmospheric/ecliptic origin, and is therefore prone to changes between exposures. If not subtracted, the compositing of all these exposures will often produce an ugly patchwork created by all the different individual backgrounds. A solution to this problem is to apply background-subtraction prior to resampling and co-adding the data. Large-scale gradients of instrumental origin are commonly found on astronomical images, hence subtracting a constant from each frame will generally yield poor results, as shown in Fig. 11). It is therefore necessary to subtract a smooth *background map* which contains the low-spatial-frequency noise components of the data, including any offset. Subtracting a background-map may alter or destroy the signal of scientific interest; thus some caution is needed in choosing parameters for this procedure.

Background subtraction is activated by setting the SUBTRACT_BACK configuration parameter to Y (which is the default). Setting it to N will disable subtraction, but background *estimation* will still take place (it is needed by other SWARP tasks), and the processing time will stay approximately the same.

Background estimation uses SExtractor's algorithm and is controllable with the same keywords[10]. The following is largely copied from SExtractor documentation (Bertin 1999).

To construct the background map, SWARP makes a first pass through the pixel data, computing an estimator for the local background in each mesh of a grid that covers the whole frame. The background estimator is a combination of $\kappa.\sigma$ clipping and mode estimation, similar to the one employed in Stetson's DAOPHOT program (see e.g. Da Costa 1992). Briefly, the local background histogram is clipped iteratively until convergence at $\pm 3\sigma$ around its median; if σ is changed by less than 20% during that process, we consider that the field is not crowded and we simply take the mean of the clipped histogram as a value for the background; otherwise we estimate the mode with:

$$\text{Mode} = 2.5 \times \text{Median} - 1.5 \times \text{Mean} \tag{4}$$

This expression is different from the usual approximation

$$\text{Mode} = 3 \times \text{Median} - 2 \times \text{Mean} \tag{5}$$

---

[10]In SWARP All background configuration keywords accept a list of values, one value for each input frame.
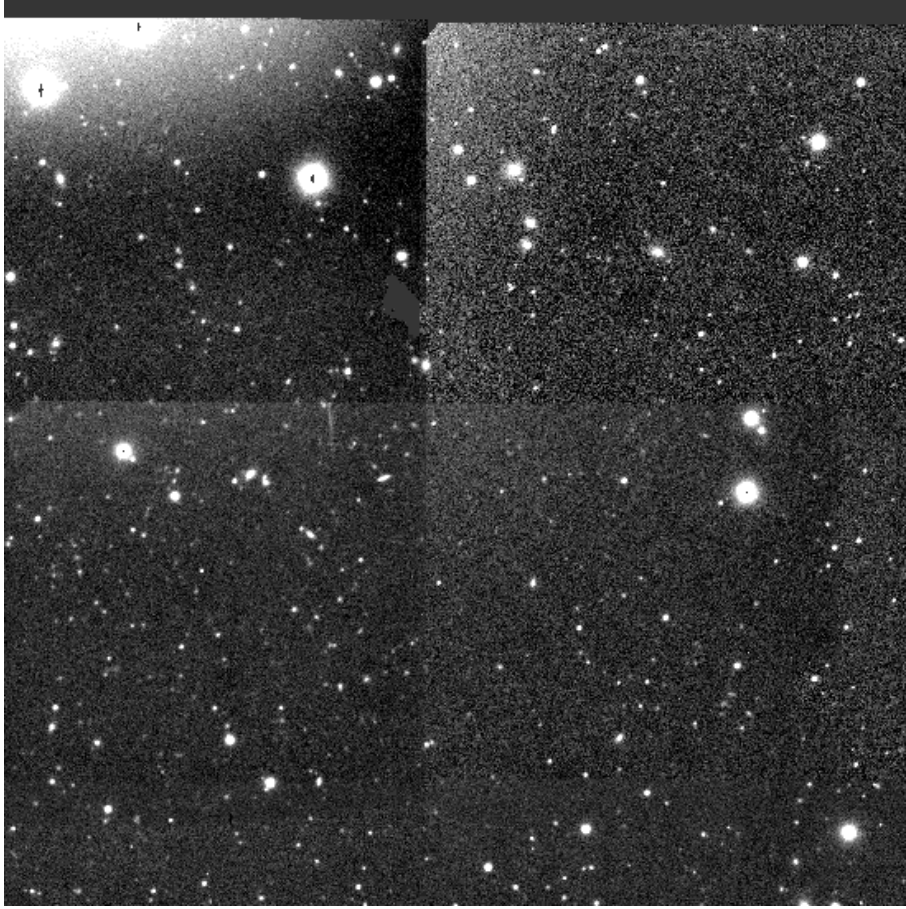
Figure 11: Example of residual gradients in a co-addition after a constant has been subtracted from input images.

(e.g. Kendall and Stuart 1977), but was found to be more accurate with our clipped distributions, from the simulations we made. Fig. 12 shows that the expression of the mode above is considerably less affected[11] by crowding than a simple clipped mean — like the one used in FOCAS (Jarvis and Tyson 1981) or by Infante (1987) — but is $\approx 30\%$ noisier. For this reason we revert to the mean in non-crowded fields.

The choice of the mesh size (in pixels), `BACK_SIZE`, is critical. If it is too small, the background estimation is affected by the presence of objects and random noise. But more important is the fact that part of the flux of extended objects can be absorbed in the background map. The effect may be almost unnoticeable on individual input images (where the signal-to-noise ratio is low), and have measurable photometric consequences on the deep, coadded image. It is therefore advised to use large `BACK_SIZE`s in SWarp. Of course if the mesh size is too large, it will not be able to reproduce all the variations of the background; a good compromise has to be found by the user. Typically, for reasonably sampled images, a size of 128 (the default) to 512 pixels should work well.

The final background map is a (natural) bicubic-spline interpolation between the meshes of the grid. Before interpolating, a median filter can be applied to suppress possible local overestimations due to bright stars or artifacts. `BACK_FILTERSIZE` sets the size (in background meshes) of

---

[11]Obviously in some very unfavorable cases (like small meshes falling on bright stars), it leads to totally inaccurate results.
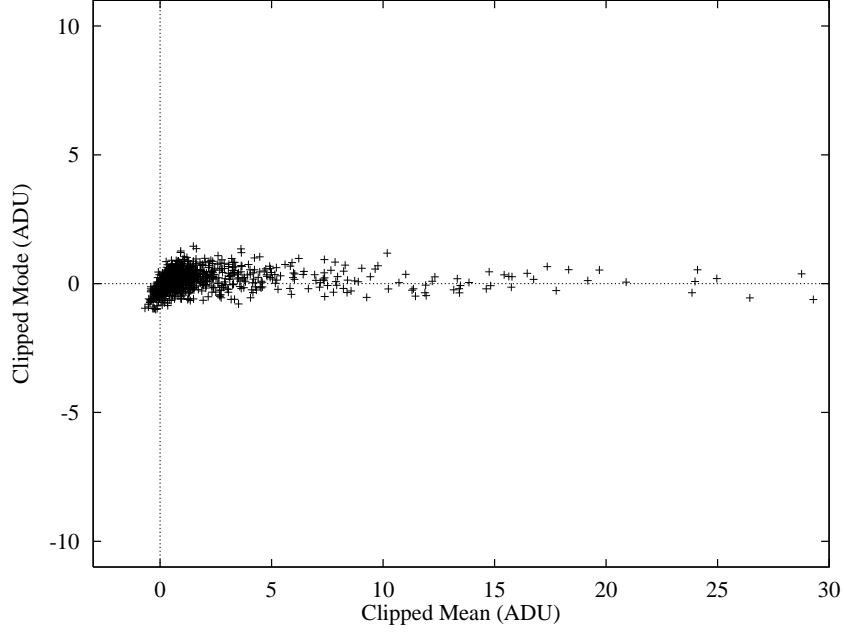
Figure 12: Simulations of 32×32 pixels background meshes polluted by random Gaussian profiles. The true background lies at 0 ADU. While being slightly noisier, the clipped "Mode" gives a more robust estimate than a clipped Mean in crowded regions.

the median filter. "1" means no filtering applied to the background grid. Usually a size of 3 meshes (the default) is sufficient, but it may be necessary to use larger dimensions, especially to compensate, in part, for small background mesh sizes, or in the case of large artifacts in the images. Median filtering also helps reducing possible ringing effects of the bicubic-spline around bright features. In some specific cases it might be desirable to median-filter only background meshes whose original values exceed some threshold above the filtered-value. This differential threshold is set by the BACK_FILTERTHRESH parameter, in ADUs. The default is 0.

By default the computed background-map is automatically subtracted from the input image. But there are some situations where it is more appropriate to subtract a *constant* from the image (e.g., images where the background noise distribution is strongly skewed). The BACK_TYPE configuration parameter (set by default to "AUTO") can be switched to MANUAL to allow for the value specified by the BACK_DEFAULT parameter to be subtracted from the input image. The default value is 0.

As said before, the background estimation procedure is used not only for background subtraction, but also for other tasks in SWARP, such as weight calibration. Thus even if SUBTRACT_BACK is set to N or BACK_TYPE is in MANUAL mode, "reasonable" values for other background parameters must be given to ensure proper working of the software.

Note that the present version of background subtraction doesn't work on non-2D images.

6.8 Scaling the flux

How are fluxes modified by image warping? Let us assume F is the integrated flux (in units of e^- for simplification) of a source of finite extent $\mathcal{S}$ that would be recorded on a perfect detector

25

array. In the continuous limit, we define

$$F \equiv \int_{\mathcal{S}} f(\boldsymbol{x})\, d^2x, \tag{6}$$

where $f(\boldsymbol{x})$ is the pixel value at physical position $\boldsymbol{x}$ on the detector. In real life, pixel values are affected by a variable efficiency q, yielding a measured "raw" flux

$$F_r = \int_{\mathcal{S}} q(\boldsymbol{x}) f(\boldsymbol{x})\, d^2x. \tag{7}$$

Digital images are generally divided by a "flat-field" and even a "super-flat" prior to SWARPing. The assumption behind flat-fielding is that the light received from the sky or the dome, and recorded to form the flat-field, has uniform radiance (i.e. constant flux per solid angle). The flux measured on the flat-fielded image is therefore

$$F_f = \int_{\mathcal{S}} \frac{q(\boldsymbol{x}) f(\boldsymbol{x})}{f_0 q(\boldsymbol{x}) \delta\Omega(\boldsymbol{x})}\, d^2x = \int_{\mathcal{S}} \frac{f(\boldsymbol{x})}{f_0 \delta\Omega(\boldsymbol{x})}\, d^2x, \tag{8}$$

where $\delta\Omega$ is the local sky area sustained by a pixel (area $\equiv 1$ in pixel units), and f_0 the scaling factor of the flat-field, which we will set to 1 for the sake of simplicity. As can be seen, flat-fielding does not make the image "flat" in terms of sensitivity; it introduces a dependency with astrometrical distorsion. With most cameras the effect is generally small (≤ 1 millimag). The resampling operations described in §6.6 are designed to conserve surface brightness per pixel, hence the flux recorded on the warped image, with new physical coordinates $\boldsymbol{x}'$ is now

$$F_{fw} = \int_{\mathcal{S}} \frac{f(\boldsymbol{x})}{\delta\Omega(\boldsymbol{x})}\, d^2x' = \int_{\mathcal{S}} \frac{f(\boldsymbol{x})}{\delta\Omega(\boldsymbol{x})} \left| \frac{\partial x_i'}{\partial x_j} \right|\, d^2x = \int_{\mathcal{S}} f(\boldsymbol{x}) \left| \frac{\partial x_i'}{\partial \xi_j} \right|\, d^2x. \tag{9}$$

$\boldsymbol{\xi}$ represents the local (angular) sky coordinate vector. We have made use of the fact that if pixel size is small compared to the rate of change of plate scale (which is almost always true), $\delta\Omega(\boldsymbol{x})$ is equal to the Jacobian of the de-projection $\left| \frac{\partial \xi_i}{\partial x_j} \right|$. Now if an equal-area projection is selected for the output image, $\left| \frac{\partial x_i'}{\partial \xi_j} \right|$ is constant and we have the nice relation $F_{fw} \propto F$. This means that swarping properly flatfielded data using an equal-area output projection produces an image with a perfectly flat response to the incoming flux[12]. This is why equal-area projections should be prefered to other projections whenever possible.

Immediately following resampling, the intensity of each image is scaled according to the configuration parameters. The flux-scaling parameter p_i is the product of two factors: a "photometric" factor, and an "astrometric" one. Currently, the photometric part must be specified by the user (for instance in a pipeline it is generally produced by the photometric calibration process). The photometric factor can be set directly using the configuration parameter FSCALE_DEFAULT: there must be one value per input image, separated by a coma. It can also be read from the FITS header. The FSCALE_KEYWORD configuration parameter tells SWARP what FITS keyword to look for in each input image. The default is "FLXSCALE". If the FSCALE_KEYWORD is not found in the image header, then the FSCALE_DEFAULT value is taken instead. FSCALE_DEFAULT is defaulted to 1 for all images.

The astrometric part of the flux-scaling factor corrects for the difference in pixel size between the input and output images. The FSCALASTRO_TYPE configuration parameter controls the behaviour of this "astrometric" flux-scaling. The default behaviour (FSCALASTRO_TYPE FIXED) is to apply

---

[12]Warning: this flux correction can be extremely inaccurate — 10% error or even more! — for sources that are undersampled on the output image.

a *constant* correction factor to account for possible mismatches in pixel size. Hence flux is conserved only when equal-area projections are used for both input and output. `FSCALASTRO_TYPE VARIABLE` applies a pixel-scale correction variable throughout the field and can therefore be used on any kind of projection. The impact on processing speed is negligible when bi-cubic spline interpolation is used (§6.5.3). "Astrometric" flux-scaling can be deactivated by using the `FSCALASTRO_TYPE NONE` option.

6.9   Combining resampled images

This is the last part of the processing. Now at each pixel position of the output image, SWARP has to combine data values coming from all the resampled image, each one coming with a rough estimate of its variance (from the resampled weight-map). Many combinations are therefore possible.

6.9.1   Various types of image combination

The user can choose between the following options, as arguments to the `COMBINE_TYPE` configuration parameter:

- `AVERAGE`: The output is simply an unweighted average of all pixel values with non-zero weights:

$$F = \frac{\sum_i p_i f_i}{n_{\neq 0}}, \tag{10}$$

where p_i is the flux scaling factor (see §6.8), and the composite weight is

$$W = \frac{n_{\neq 0}^2}{\sum_i \frac{1}{q_i w_i}}, \tag{11}$$

where the w_i are proportional to the inverse of the scaled variance: $\frac{1}{p_i^2 \sigma^2}$. Needless to say that this combination is not optimum in terms of S/N, unless all input images have identical Gaussian noise.

- `CHI2`: The output is the square-root of the reduced χ^2 of all pixel values with non-zero weights:

$$F = \sqrt{\frac{\sum_i w_i f_i^2}{n_{\neq 0}}}. \tag{12}$$

By construction, the composite weight (the absolute one) is

$$W \equiv 1. \tag{13}$$

The result of the combination is a so-called "χ^2 image". Although it does not respond linearly to the input signals, it can be used for detecting sources. As shown by Szalay et al. (1999), the "χ^2 image" is indeed the optimum combination to achieve panchromatic detection on a set of images taken at different wavelengths, provided the data sets are background-noise limited and that the noise is uncorrelated between frames. This assumes further that the *Point Spread Function* (PSF) has been homogenized in all channels. χ^2 images are most often used in deep panchromatic surveys requiring photometric redshift analyses. The double-image mode of SExtractor allows one to detect on the χ^2 image while making the photometric measurements on each of the single-band images.

- `MEDIAN`: The output is the median of all scaled pixel values with non-zero weights:

$$F = \text{median}(f_i). \tag{14}$$

Assuming Gaussian noise distribution, we obtain the following approximation to the composite weight (see e.g. Kendall & Stuart 1977):

$$W = \begin{cases} \frac{2}{\pi} \left(\frac{\sum_i \sqrt{w_i}}{n_{\neq 0}} \right)^2 \left(n_{\neq 0} + \frac{\pi}{2} - 1 \right) & \text{if } n_{\neq 0} \text{ is even,} \\ \frac{2}{\pi} \left(\frac{\sum_i \sqrt{w_i}}{n_{\neq 0}} \right)^2 \left(n_{\neq 0} + \pi - 2 \right) & \text{otherwise.} \end{cases} \tag{15}$$

This approximation can become inaccurate if w_j varies by large proportions (a factor of 3 or more) from frame to frame. The median is convenient for combining data polluted by unidentified glitches or noise spikes. It generally provides "safe" (robust) co-additions even with strongly non-Gaussian noise distributions. However it is suboptimal for Gaussian noise: the resulting variance is increased by $\approx 60\%$ compared to what is obtained with an average. As with all non-linear combinations, one should check that input images have approximately the same Point Spread Function if point-sources are to be co-added. One should also avoid the `MEDIAN` option when combining images with very different depths.

- `MIN`: The output is the minimum of all pixel values with non-zero weights:

$$F = \min(p_i f_i). \tag{16}$$

The variance of F is too "noise-distribution dependent" to allow some estimation, hence we set

$$W = \begin{cases} 1 \text{ if } n_{\neq 0} \neq 0, \\ 0 \text{ otherwise.} \end{cases} \tag{17}$$

- `MAX`: The output is the maximum of all pixel values with non-zero weights:

$$F = \min(p_i f_i). \tag{18}$$

The variance of F is too "noise-distribution dependent" to allow some estimation, hence we set

$$W = \begin{cases} 1 \text{ if } n_{\neq 0} \neq 0, \\ 0 \text{ otherwise.} \end{cases} \tag{19}$$

Maxima and minima can be useful for identifying defects or rare events in a set of data.

- `SUM`: The output is an unweighted sum of all pixel values with non-zero weights:

$$F = \sum_i p_i f_i. \tag{20}$$

The composite weight is

$$W = \frac{1}{\sum_i \frac{1}{q_i w_i}}. \tag{21}$$

Subtractions can be carried out instead of additions by using negative flux scaling factors.

- `WEIGHTED`: The output is a weighted average of input values:

$$F = \frac{\sum_i w_i p_i f_i}{\sum_i w_i}. \tag{22}$$

The output weight is just the sum of input weights:

$$W = \sum_i w_i. \qquad (23)$$

This combination should be the most appropriate for detecting and measuring faint sources on properly weighted, homogeneous images. Because it is a linear processing, new data can be added later if needed.

6.9.2 Weighted coaddition

Weight-maps provide a convenient way to store the standard flux error assigned to each pixel.

For each input image $1 \leq i \leq N$ entering image combination, one can define the following parameters in an arbitrarily small "pixel" j:

- The local, uncalibrated flux $f_{ij} = \overline{f_{ij}} + \Delta f_{ij}$, where $\overline{f_{ij}}$ is the flux contributed by the sky background, and Δf_{ij} that contributed by celestial sources,

- the local, uncalibrated variance of the flux $\sigma_{ij}^2 = \overline{\sigma_{ij}^2} + \Delta\sigma_{ij}^2$, where $\overline{\sigma_{ij}^2}$ is the flux variance contributed by the background noise, and $\Delta\sigma_{ij}^2$ that contributed by the photon statistics of celestial sources,

- the local, normalized weight[13] w_{ij},

- the electronic gain of the CCD g_i, in e$^-$/ADU (defined at $w_{ij} = 1$),

- the relative flux scaling factor p_i deduced from the photometric solution to calibrate the images:
$$p_i \Delta f_{ij} = p_l \Delta f_{lj} \quad \forall i, j, l, \qquad (24)$$

- and the relative weight scaling factor q_i derived from the comparison of the background noise level with the normalized weight; input images will be weighted with $q_i w_i$.

Δf_i, $\Delta\sigma_i^2$, w_i and g_i are related through:

$$g_i w_{ij} = \frac{\Delta f_{ij}}{\Delta\sigma_{ij}^2}. \qquad (25)$$

Now, to optimally co-add calibrated images, one could weight them using

$$q_i w_{ij} = \frac{1}{p_i^2 \sigma_{ij}^2}. \qquad (26)$$

However, such weight maps exhibit strong variations on small scales, in the presence of celestial objects (σ_{ij}^2 increases a lot on bright pixels). The modulating effect of weighting, combined with variations of the PSF and sampling errors would lead to significant distorsions of stellar profiles. It is therefore more appropriate to weight pixels according to the intensity of the local background noise, which is far smoother:

$$q_i w_{ij} = \frac{1}{p_i^2 \overline{\sigma_{ij}^2}}. \qquad (27)$$

---

[13]Both fluxes and weights may have gone through resampling, but for the sake of clarity we shall from now drop the " ~ " from §6.6

This has also the advantage that one can use normalized flat-fields as w_{ij}'s, without prior knowledge of the CCD gain in the case where instrumental noise is negligeable. For faint objects, this weighting scheme is as efficient as that of (26), and is only suboptimum for the objects with very high surface-brightness, when the q_i's vary a lot from exposure to exposure. But as these objects are easily detected on individual exposures, the most accurate photometry is still possible by combining the N measurements.

In practice, SWARP can read several types of weight-maps, although they are all internally converted to variance for processing. The input weight-map format must be specified with the WEIGHT_TYPE keyword. WEIGHT_TYPE NONE is for no input weight-map (the default), MAP_RMS for indicating that the data contain the absolute standard deviation of pixel values, MAP_VARIANCE, for weight-map data stored as relative variances, and MAP_WEIGHT for relative weights. Relative variance maps and weight maps are re-scaled internally using local variance measurements made directly on the input images. This has the advantage of making it possible to use for instance a single flat-field image as a weight-map for a whole series of background-noise-limited images with different exposures. But automatic re-scaling may sometimes lead to inaccurate weightings, because of source crowding or "complicated" weight-maps for instance. In cases where accurate weighting is important, one should either use absolute standard deviation maps (WEIGHT_TYPE MAP_RMS), or turn off weight/variance rescaling by setting the RESCALE_WEIGHTS configuration switch to N (default is Y).

When producing composite fields larger than the input images, the latter must be background-subtracted prior to coaddition, to avoid generating discontinuites in the output image. Thus one can write the output coadded flux as

$$\Delta f_j = \frac{\sum_i q_i w_{ij} p_i \Delta f_{ij}}{\sum_i q_i w_{ij}} = p_l \Delta f_{lj} \quad \forall l, \tag{28}$$

and the resulting variance as

$$\Delta \sigma_j^2 = \frac{\sum_i q_i^2 w_{ij}^2 p_i^2 \Delta \sigma_{ij}^2}{\left(\sum_i q_i w_{ij}\right)^2}. \tag{29}$$

Using (25), an equivalent "local gain" G_j in the coadded image can be computed:

$$G_j w_j = \frac{\Delta f_j}{\Delta \sigma_j^2} = \frac{\sum_i q_i w_{ij} p_i \Delta f_{ij}}{\sum_i q_i^2 w_{ij}^2 p_i^2 \Delta \sigma_{ij}^2} \cdot \sum_i q_i w_{ij}, \tag{30}$$

where w_j is the composite weight-map of the coadded image. From our definition of weights, w_j is inversely proportional to $\overline{\sigma_j^2}$ and must be 1 if all input weights are at 1; hence it is easy to show that

$$w_j = \frac{\sum_i q_i w_{ij}}{\sum_i q_i}. \tag{31}$$

Substituting (31) in (30), and using (24) and (25), one gets

$$G_j = \frac{\sum_i q_i w_{ij}}{\sum_i q_i^2 w_{ij} p_i g_i^{-1}} \cdot \sum_i q_i. \tag{32}$$

Unfortunately, as can be seen, this "coadded gain" will vary with position in the general case. Nevertheless, some approximations can be made to simplify this expression. First of all, in most cases, g_i will be almost constant from one input image to another. Second, if exposures are taken under photometric conditions with constant sky brightness and negligible instrumental noise, one should have $q_i \propto p_i^{-1}$, removing the dependence with input weights, and therefore position in the coadded image. In that very case, the resulting gain is simply

$$G \approx g \sum_i p_i^{-1}. \tag{33}$$

Sadly, in many bands, the presence of clouds does not decrease the sky brightness as much as source brightness, and doesn't act at all like a decrease in global sensitivity or exposure time. Coadded regions of a survey that are taken under non-photometric conditions experience fluctuating "gains". But this effect is generally small: taking the quite unfavourable case of $q_i \propto p_i^{-2}$ (constant background noise) with p_i's varying by a factor of 4 between two exposures, and weights varying from 0.5 to 1, makes G_j to vary by 20% at most. This should not cause significant difficulties in any profile fitting routine. Therefore (33) still remains a very good approximation in the general case, and the "coadded gains" provided by this method are still more stable than what unweighted co-addition offers.

This does not prevent regions with lower coverage (in which N is smaller) having lower gains. To avoid large gain drops in the gaps of CCD mosaic images, it is recommended for the observations to use large, random dithering patterns consisting of at least 4-5 exposures per covered sky area.

6.9.3 Image buffering and memory constraints

In order to maximize the efficiency of the image combination process, SWARP version 2.0 and above allocates a significant amount of memory to buffer input and output data. This amount of memory can be set by the user with the COMBINE_BUFSIZE configuration parameter. The default value for COMBINE_BUFSIZE is 256 (Megabytes). If your machine is a bit short of memory you should decrease this value. Conversely if you need to combine a large number of overlapping images, you might want to set COMBINE_BUFSIZE to a substantial fraction of the memory available on your machine to avoid "disk thrashing".

6.9.4 Overlap information

SWARP uses the Bron and Kerbosh algorithm (1973) based on FITS header information to estimate the maximum number of input frames that overlap and optimize the management of its internal buffers. The maximum overlap density is displayed on screen prior to image combination and can be found in the output XML meta-data. Note that this estimate is somewhat approximative as it ignores input weights (which can be zero over significant fraction of the overlap), and that there can only be one single overlapping frame per multi-extension FITS input file.

The maximum overlap density is also used for computing the maximum accumulated exposure time and maximum equivalent gain in the output frame. The maximum exposure time is simply the sum of all the exposure times from the individual overlapping frames. The maximum equivalent gain is computed using (32) for COMBINE_TYPEs AVERAGE, SUM and WEIGHTED. For all other COMBINE_TYPEs the equivalent gain is just the average input gain in the overlap after the scaling of pixel values.

The saturation level of the combined data (in ADU) is also computed if saturation values are provided in the input image headers through the SATLEV_KEYWORD FITS keyword, or directly with the SATLEV_DEFAULT configuration parameter. For any COMBINE_TYPE the output saturation level is defined as the minimum of all input saturation values after astrometric and photometric rescaling have been applied.

7 Two-step co-addition and resampling

All the operations described so far can be done in one single run of SWARP. This generally sufficient for small projects, which usually involve observations conducted over a short period of time. However, for large sky surveys that can extend over years, this implies "waiting" for the data to accumulate after passing through the reduction pipeline. In this case SWARP would only be started once all the fields that cover a given sky area are available. Much of SWARP processing time is spent in resampling the data; therefore for projects that extend over a long period of time, it would be more efficient to resample the images as they come out of the reduction pipeline. The remaining of the work, co-addition, could be done at a later date.

To solve this problem, versions ≥ 1.32 of SWARP allow the internal processing pipeline to be split in two: image-resampling and image-combining. If the `COMBINE` configuration parameter is set to `N`, SWARP stops right after having background-subtracted and resampled the input images. The `DELETE_TMPFILES` option is then automatically deactivated.

To combine images resampled at an earlier stage, `RESAMPLE` should be set to `N`: in that case SWARP will skip all the background-subtraction and resampling stage, and jump directly to the combine process. Prior to version 2.0 this feature only worked with input resampled images produced by SWARP, because of some specific information needed in the FITS header. Since version 2.0, any FITS image can be used. However, when `RESAMPLE` is set to `N`, SWARP combines input images with the implicit assumption that they all share the same `CRVAL` and `CDELT` WCS parameters: images are placed in the final frame according to their `NAXISN`'s and the integer part of their `CRPIX`'s. The `RESAMPLE N` feature can be used for instance to extract sub-images, or paste input images within an arbitrary empty frame.

Setting both `RESAMPLE` and `COMBINE` to `N` will not produce any output file, but can be useful to check the content of input data or to adjust output astrometric parameters. By default, `RESAMPLE` and `COMBINE` are both set to `Y`.

It may sometimes be useful to create only the header of what will become the combined image; for instance for generating an output "*.head*" file that will define the output projection system. Such a "*.head*" file can then be copied to several machines that will resample input images to a common projection, for later co-addition. This can be done by setting the `HEADER_ONLY` configuration parameter to `Y`; processing will stop before resampling like in the case where `RESAMPLE` and `COMBINE` are set to `N`, but this time the header of the output image will be written to disk.

8 Examples

In the following, examples of use of SWARP are given, together with commented configuration files.

8.1 Example 1

Let us assume one wants to produce a full-sky Aitoff representation in galactic coordinates of a series of observed fields stored in `*.fits` 2D files(with WCS info), for illustration purposes. The files might be dummy ones, supplemented with hand-made headers, or obtained from virtual telescopes such as *SkyView*[14], or real ones. In all cases an additional full sky map is useful to

---

[14]`http://skyview.gsfc.nasa.gov`

delimit the full-sky: one may for instance download a 360 degrees Aitoff projection of COBE-DIRBE data from SkyView. The syntax is

% swarp *.fits

A possible default.swarp configuration file is

```
#----------------------------- Output -----------------------------------

IMAGEOUT_NAME    coadd.fits
WEIGHTOUT_NAME   coadd.weight.fits


#----------------------------- Input Weights ----------------------------

WEIGHT_TYPE      MAP_WEIGHT            # Not used here
WEIGHT_SUFFIX    .weight.fits
WEIGHT_IMAGE


#----------------------------- Co-addition ------------------------------

COMBINE_TYPE     AVERAGE              # This coaddition is for illustration
                                      # only: the weight-map will contain
                                      # a sum of field footprints


#----------------------------- Astrometry -------------------------------

CELESTIAL_TYPE   GALACTIC             # Coordinate system forced to galactic
PROJECTION_TYPE  AIT                  # Code for Aitoff
CENTER_TYPE      MANUAL               # Imposed to alpha = delta = 0.0
CENTER           00:00:00.0, +00:00:00.0 #
PIXELSCALE_TYPE  MANUAL               # The full sky area will exceed the
                                      # fraction that contains the fields
PIXEL_SCALE      1800.0               # in arcsec: Half a degree per pixel
                                      # at image center, on both axes
IMAGE_SIZE       800,400              # 360/0.5 = 720, 180/0.5 = 360,
                                      # plus a margin


#----------------------------- Resampling -------------------------------

RESAMPLING_TYPE  BILINEAR             # For illustration purposes, no need
                                      # for a sophisticated interpolation
OVERSAMPLING     4                    # A small oversampling only to have
                                      # pretty, antialiased field limits
INTERPOLATE      N
GAIN_KEYWORD     GAIN
GAIN_DEFAULT     0.0


#----------------------- Background subtraction -------------------------

SUBTRACT_BACK    N                    # No background subtraction
BACK_TYPE        AUTO
```

```
BACK_DEFAULT    0.0
BACK_SIZE       128
BACK_FILTERSIZE 3


#----------------------- Virtual memory management -----------------------


VMEM_DIR        .
VMEM_MAX        2047
MEM_MAX         256                    # 256 MB should be enough to avoid
                                       # swapping


#-------------------------- Miscellaneous --------------------------------


DELETE_TMPFILES Y                      # Delete temporary resampled FITS files
VERBOSE_TYPE    NORMAL
```

In this application, coverage maps can be generated using the output weight-map instead of the image itself.

8.2   Example 2

In this example, one has a set of CCD images taken with a standard dithering strategy, `input*.fits`, and the related set of weight-maps `input*.w.fits` However the unusual thing is that for some reason the output has to be tilted by 30 degrees with respect to the local north-south axis, and the pixels must have an aspect ratio of 16:9!

First, one starts with a fairly standard configuration file:

```
#------------------------------- Output ----------------------------------


IMAGEOUT_NAME   coadd.fits             # Output filename
WEIGHTOUT_NAME  coadd.weight.fits      # Output weight-map filename


#---------------------------- Input Weights ------------------------------


WEIGHT_TYPE     MAP_WEIGHT             #
                                       # (all or for each weight-map)
WEIGHT_SUFFIX   .w.fits                # Suffix to use for weight-maps
WEIGHT_IMAGE                           # Weightmap filename if suffix not used
                                       # (all or for each weight-map)


#---------------------------- Co-addition --------------------------------


COMBINE_TYPE    WEIGHTED               # weight-maps are provided


#---------------------------- Astrometry ---------------------------------


CELESTIAL_TYPE  NATIVE                 # Standard stuff
PROJECTION_TYPE TAN                    # A tangent projection will do
CENTER_TYPE     ALL                    # We want all the data to fit in
CENTER          00:00:00.0, +00:00:00.0 # Not used in CENTER_TYPE ALL mode
```

```
PIXELSCALE_TYPE MEDIAN                          # Will be overriden by coadd.head
PIXEL_SCALE    0.0                              # Not used in MEDIAN mode
IMAGE_SIZE     0                                # Automatic sizing


#---------------------------- Resampling --------------------------------


RESAMPLING_TYPE LANCZOS3                         # High quality resampling
OVERSAMPLING    0                                # Auto. oversampling (=1 in that case)
INTERPOLATE     N
GAIN_KEYWORD    GAIN
GAIN_DEFAULT    0.0


#------------------------- Background subtraction ------------------------


SUBTRACT_BACK   Y                                # Needed for co-adding dithered fields
BACK_TYPE       AUTO
BACK_DEFAULT    0.0
BACK_SIZE       128
BACK_FILTERSIZE 3


#----------------------- Virtual memory management -----------------------


VMEM_DIR        .
VMEM_MAX        2047
MEM_MAX         128                              # 128 MB should be enough to avoid
                                                 # swapping


#--------------------------- Miscellaneous ------------------------------


DELETE_TMPFILES Y                                # Delete temporary resampled FITS files
VERBOSE_TYPE    NORMAL
```

To implement the unusual output features required, one must write a `coadd.head` ASCII file that contains a custom anisotropic scaling matrix. A `coadd.head` for pixels 0.2" large, that tilts the image by 30 degrees and applies a 16:9 anamorphosis to the data would be:

```
CD1_1  =   6.4150E-5
CD1_2  =   2.0833E-5
CD2_1  =  -3.7037E-5
CD2_2  =   3.6084E-5
END
```

9 Troubleshooting

In case you face a problem which not listed below, please do not hesitate to discuss it in the SWARP section of the AstrOmatic forums[15].

My window terminal crashes during a long SWARP run!

---
[15]http://astromatic.net/forum

Unexpected crashes of XTERM windows have been reported. This seems to be caused by the large number of ANSI control sequences that SWARP sends to the terminal. You may either set the SWARP configuration keyword VERBOSE_TYPE to QUIET or FULL, and/or redirect the output to a file.

SWARP crashes with error messages like "> *Error*: pthread_create() failed...".

The multithreaded version of SWARP requires a fairly large stack, that may exceed the maximum value allowed by your shell. Use the shell command `limit` to increase the `stacksize` parameter if required (you might need the root privileges to change this if it is a "hard" limit).

SWARP crashes with error messages like "> *Error*: Not enough memory for...", although I have properly set the MEM_MAX VMEM_MAX parameters.

The maximum value allowed by your shell for memory use might be set too low. Use the shell command `limit` to increase the `datasize`, `memoryuse` and `vmemoryuse` parameters if required (you might need the root privileges to change this if it is a "hard" limit).

SWARP crashes with error messages like "> *Error*: cannot open for reading..." during the co-addition phase, although it had no problem accessing the same files for resampling.

The maximum number of open files allowed by your shell might be set too low. Use the shell command `limit` to increase the `descriptors` and `openfiles` parameters if required (you might need the root privileges to change this if it is a "hard" limit).

10   Acknowledging SWARP

Please use the following reference: Bertin E., Mellier Y., Radovich M., Missonnier G., Didelon P., Morin B., 2002, in *Astronomical Data Analysis Software and Systems XI*, ASP Conf. Series **281**, 228.

11   Acknowledgements

Many thanks go to Mark Calabretta (ATNF/CSIO, Epping) for his great astrometric library, Nicolas Devillard (ESO, Garching) for introducing me to memory-mapping techniques, Chiara Marmo (TERAPIX/IAP) for adding the computation of saturation levels and doing many bugfixes, Mireille Dantel-Fort, Laurent Domisse, Frédéric Magnard, Yannick Mellier (TERAPIX/IAP), Mario Radovich (OAC, Naples), Roeland Rengeling (Sterrewacht, Leiden), Roy Williams (CACR, CalTech), Jan Kohnert (AIP, Postdam), Jörg P. Dietrich (ESO, Garching) Dafydd Wyn Evans (IoA, Cambridge),Joe Mohr, Shantanu Desai (University of Illinois) and all the AstrOmatic forum users for testing and suggestions, and Henry Joy McCracken (TERAPIX/IAP) for help with the manual and additional testing.

References

[1] Anscombe F.J., 1948, *Biometrika* **15**, 246

[2] Berriman G.B., Good J.C., Laity A.C., Kong M., 2008, *ASP Conference Series* **394**, 83

[3] Bertin E., 1999, *SExtractor 2.1, User's manual*, IAP

[4] Bertin E., 2006, *ASP Conference Series* **351**, 112

[5] Bron C., Kerbosch J., 1973, *Communications of the ACM* **v.16, n.9**, 575

[6] Da Costa G.S., 1992, in *Astronomical CCD Observing and Reduction Techniques*, ed. Howell S.B. (ASP Conf. Series)

[7] Fruchter A., Hook R.N., 1997, *SPIE* **3164**, 120

[8] Fruchter A., Hook R.N., 2002, *PASP* **114**, 144

[9] Infante L., 1987, *A&A* **183**, 177

[10] Jarvis J.J., Tyson J.A., 1981, *AJ*, **86**, 476

[11] Kendall M., Stuart K., 1977, *The Advanced Theory of Statistics*, **Vol. 1**, (Charles Griffin & Co., London)

[12] Makovoz D., Khan I., 2005, *ASP Conference Series* **347**, 81

[13] Starck J.-L., Murtagh F., Bijaoui A., 1998, *Image Processing and Data Analysis* (Cambridge University Press)

[14] Szalay A.S., Connolly A.J., Szokoly G.P., 1999, *AJ* **117**, 68

[15] Valdes F., 1998, *ASP Conference Series* **145**, 53

[16] Wolberg G., 1992, *Digital Image Warping* (IEEE Computer Society Press)