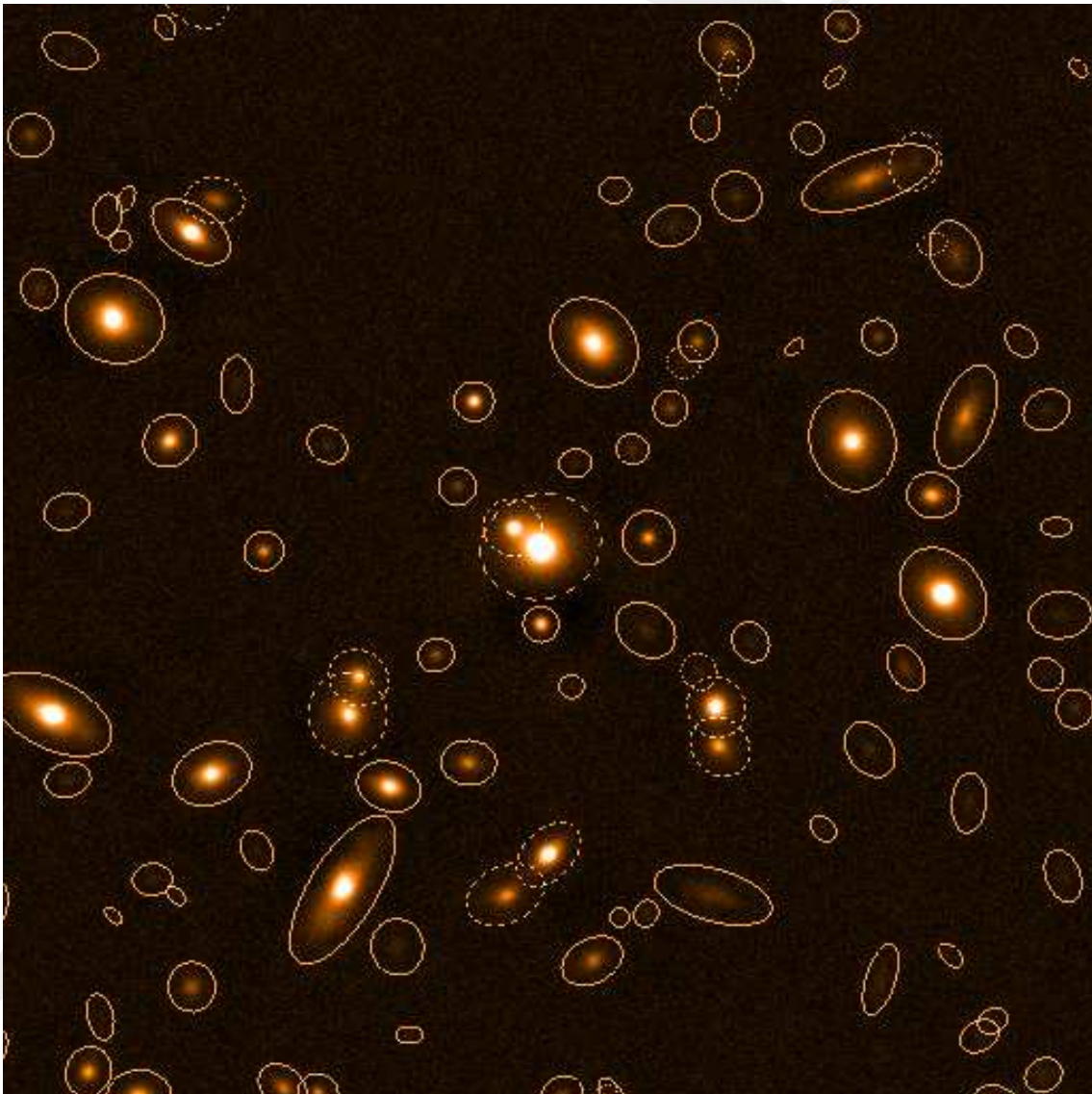


SExtractor v2.13

User's guide

Emmanuel BERTIN
Florence Durret
Gary MAMON
Institut d'Astrophysique de Paris
CNRS UMR7095/UPMC

February 28, 2012



Contents

| | | |
|----------|--|-----------|
| 1 | What is SExtractor? | 2 |
| 2 | Installing the software | 3 |
| 2.1 | Obtaining SExtractor | 3 |
| 2.2 | Software and hardware requirements | 3 |
| 2.3 | Installation | 4 |
| 2.3.1 | Installation from the source archive | 4 |
| 2.3.2 | Installation from an RPM archive | 4 |
| 3 | Using SExtractor | 5 |
| 3.1 | The configuration file | 5 |
| 3.1.1 | Creating a configuration file | 5 |
| 3.1.2 | Format of the configuration file | 5 |
| 3.1.3 | Configuration parameter list | 6 |
| 3.2 | The catalogue parameter file | 11 |
| 3.2.1 | Format | 11 |
| 3.2.2 | Variants | 12 |
| 3.3 | Example of configuration | 12 |
| 4 | Overview of the software | 13 |
| 5 | Handling of image data | 15 |
| 5.1 | Image format | 15 |
| 5.2 | Double image mode | 15 |
| 6 | Detection and segmentation | 17 |
| 6.1 | Background estimation | 17 |
| 6.1.1 | Configuration parameters and tuning | 19 |
| 6.1.2 | CPU cost | 19 |

| | | |
|-----------|---|-----------|
| 6.2 | Filtering | 19 |
| 6.2.1 | Convolution | 19 |
| 6.2.2 | Non-linear filtering | 21 |
| 6.2.3 | What is filtered, and what isn't | 21 |
| 6.2.4 | Image boundaries and bad pixels | 21 |
| 6.2.5 | Configuration parameters | 21 |
| 6.2.6 | CPU cost | 22 |
| 6.2.7 | Filter file formats | 22 |
| 6.3 | Thresholding | 23 |
| 6.3.1 | Configuration parameters. | 23 |
| 6.4 | Deblending | 23 |
| 7 | Weighting | 27 |
| 7.1 | Weight map formats | 27 |
| 7.2 | Weight threshold | 28 |
| 7.3 | Effects of weighting | 28 |
| 7.4 | Combining weight maps | 29 |
| 7.5 | Interpolation | 29 |
| 8 | Cleaning | 30 |
| 9 | Masking | 31 |
| 10 | Flags | 32 |
| 10.1 | Internal flags | 32 |
| 10.1.1 | Extraction flags: <code>FLAGS</code> | 32 |
| 10.1.2 | Weight-map flags: <code>FLAGS_WEIGHT</code> | 33 |
| 10.1.3 | Weight-map flags: <code>FLAGS_WIN</code> | 33 |
| 10.2 | External flags | 33 |
| 11 | Measurements | 34 |
| 11.1 | Positional parameters derived from the isophotal profile | 34 |
| 11.1.1 | Limits: <code>XMIN</code> , <code>YMIN</code> , <code>XMAX</code> , <code>YMAX</code> | 34 |
| 11.1.2 | Barycenter: <code>X</code> , <code>Y</code> | 35 |
| 11.1.3 | Position of the peak: <code>XPEAK</code> , <code>YPEAK</code> | 35 |
| 11.1.4 | 2nd order moments: <code>X2</code> , <code>Y2</code> , <code>XY</code> | 35 |
| 11.1.5 | Basic shape parameters: <code>A</code> , <code>B</code> , <code>THETA</code> | 36 |


| | | |
|-----------|--|-----------|
| 11.1.6 | Ellipse parameters: CXX, CYY, CXY | 37 |
| 11.1.7 | By-products of shape parameters: ELONGATION and ELLIPTICITY | 37 |
| 11.1.8 | Position errors: ERRX2, ERRY2, ERRXY, ERRA, ERRB, ERRTHETA, ERRCXX, ERRCYY, ERRCXY | 38 |
| 11.1.9 | Handling of “infinitely thin” detections | 39 |
| 11.2 | Windowed positional parameters | 39 |
| 11.2.1 | Windowed centroid: XWIN, YWIN | 40 |
| 11.2.2 | Windowed 2nd order moments: X2, Y2, XY | 40 |
| 11.2.3 | Windowed ellipse parameters: CXXWIN, CYYWIN, CXYWIN | 41 |
| 11.2.4 | Windowed position errors: ERRX2WIN, ERRY2WIN, ERRXYWIN, ERRAWIN, ERRBWIN, ERRTHETAWIN, ERRCXXWIN, ERRCYYWIN, ERRCXYWIN | 41 |
| 11.3 | Astrometry and WORLD coordinates | 42 |
| 11.3.1 | Celestial coordinates | 42 |
| 11.3.2 | Use of the FITS keywords for astrometry | 43 |
| 11.4 | Photometry | 43 |
| 12 | Model fitting | 47 |
| 13 | Checking the output | 50 |
| 14 | Cross-identification within SExtractor | 51 |
| 14.1 | The ASSOC list | 51 |
| 14.2 | Controlling the ASSOC process | 51 |
| 14.3 | Output from ASSOC | 52 |
| A | FAQs (Frequently Asked Questions) | 56 |

Licenses

Software

SEXTRACTOR 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. SEXTRACTOR 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 SEXTRACTOR. If not, see <http://www.gnu.org/licenses/>.

Documentation

This documentation is licensed under the Creative Commons Attribution-Non Commercial 3.0 Unported License , which permits unrestricted, non-commercial use, distribution and reproduction in any medium, providing that the work is properly cited. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

Chapter 1

What is SEXTRACTOR?

SEXTRACTOR (*Source-Extractor*) is a program that builds a catalogue of objects from an astronomical image. It is particularly oriented towards the reduction of large scale galaxy-survey data, but it also performs well on moderately crowded star fields. Its main features are:

- Support for multi-extension FITS.
- Speed: typically 1–10 Mpixel/s or 10–1000 sources/s with a 2 GHz processor.
- Ability to work with very large images (up to $65k \times 65k$ pixels on 32 bit machines, or $2G \times 2G$ pixels on 64 bit machines), thanks to buffered image access.
- Real-time filtering of images to improve detectability.
- Robust deblending of overlapping extended objects.
- Flexible catalogue output of desired parameters only.
- Pixel-to-pixel photometry in dual-image mode.
- Fast Point-Spread-Function and galaxy model fitting.
- Handling of weight maps and flag maps.
- Optimum handling of images with variable S/N.
- Built-in catalogue cross-identification.
- Special mode for photographic scans.
- XML VOTable-compliant catalogue output.

Additional information about what the software can or cannot do is provided in the Frequently Asked Questions section (§A).

Chapter 2

Installing the software

2.1 Obtaining SExtractor

The easiest way to obtain SExtractor is to download it from the official website¹. At this address, the latest versions of the program (source code, configuration files, and documentation) are available as standard `.tar.gz` Unix source archives as well as RPM binary packages for various architectures.²

2.2 Software and hardware requirements

SExtractor has been developed on Unix machines (GNU/Linux), and should compile on any POSIX-compliant system (this should include Mac OS X and Cygwin under Windows, at the price of some difficulties with the configuration), provided that the following libraries/packages have been installed:

- ATLAS V3.6 and above³ (<http://math-atlas.sourceforge.net/>)
- FFTW V3.0 and above⁴ (<http://www.fftw.org/>)

Note that ATLAS and FFTw are not necessary for the binary versions of SExtractor which come with these libraries statically linked.

The software is run in (ANSI) text-mode from a shell. A graphical environment is not necessary to operate SExtractor.

Memory requirements depend on the size of the images to be processed. Processing a single image should typically require about 100MB of memory. For large images (hundreds of Mpixels or more), or in double-image / weighted mode, SExtractor's memory footprint should be around 500MB, and up to 2GB in the worst cases. Swap-space can be put to contribution, although a strong performance hit is to be expected.

¹<http://astromatic.net/software/sextractor>

²Mac OS X dmg files should be available soon.

³Use the `--with-atlas` and/or `--with-atlas-incdir` options to specify the ATLAS library and include paths if the software is installed at unusual locations.

⁴Make sure that FFTW has been compiled with the `configure` options `--enable-threads --enable-float`.

2.3 Installation

2.3.1 Installation from the source archive

To install from the source, you must first uncompress and “untar” the archive:

```
tar zxvf sextractor-<version>.tar.gz
```

A new directory called `sextractor-<version>` should now appear at the current location on your disk. You should then enter the directory and follow instructions given in the file called “INSTALL”.

2.3.2 Installation from an RPM archive

SEXTRACTOR is also available as a binary RPM package for both Linux INTEL x86 (32-bit) and x86-64 (64-bit) architectures. To check which one matches your system, use the shell command

```
uname -a
```

To install SEXTRACTOR, type as a root user the following command in your shell (preceded with `su` if you don't have root access but the system administrator trusts you well enough to make you part of the `wheel` group):

```
rpm -U sextractor-<version>-1.<arch>.rpm
```

Under some circumstances, it may be necessary to force installation with

```
rpm -U --force --nodeps sextractor-<version>-1.<arch>.rpm
```

You may now check that the software is properly installed by simply typing in your shell

```
sex
```

(note that some shells require the `rehash` command to be run before making a freshly installed executable accessible in the execution path).

Chapter 3

Using SExtractor

SExtractor is run from the shell with the following syntax:

```
% sex image1 [ image2 ] [ -c configuration-file ] [ -Parameter1 Value1 ] [ -Parameter2 Value2 ] [ ... ]
```

The parts enclosed within brackets are optional. The file names of input catalogues can be directly provided in the command-line, or in lists that are ASCII files with each catalogue name preceded by '@' (one per line). One should use lists instead of the catalogue file names if the number of input catalogues 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).

3.1 The configuration file

Each time it is run, SExtractor looks for a configuration file. If no configuration file is specified in the command-line, it is assumed to be called “default.sex” and to reside in the current directory. If no configuration file is found, SExtractor will use its own internal default configuration.

3.1.1 Creating a configuration file

SExtractor can generate an ASCII dump of its internal default configuration, using the “-d” option. By redirecting the standard output of SExtractor to a file, one creates a configuration file that can easily be modified afterwards:

```
% sex -d > default.sex
```

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

3.1.2 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. Integers can be given as decimals, in octal form (preceded by digit 0), 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, and not only for string parameters. Some parameters are assigned default values in SExtractor and can therefore be omitted from the configuration file; they are listed in §3.1.3.

3.1.3 Configuration parameter list

Here is a complete list of all the *configuration* parameters known to SExtractor. Many of them should be used with their default values. Please refer to the next sections for a detailed description of their meaning.

| Parameter | default | type |
|---|----------|---|
| Description | | |
| ANALYSIS_THRESH | 1.5 | <i>floats</i> ($n \leq 2$) |
| Threshold (in surface brightness) at which CLASS_STAR and FWHM_ operate. 1 argument: relative to Background RMS (before filtering). 2 arguments: surface magnitude μ (mag arcsec ⁻²), Zero-point (mag). | | |
| ASSOC_DATA | 2,3,4 | <i>integers</i> ($n \leq 32$) |
| # of the columns in the ASSOC file that will be copied to the catalogue output. | | |
| ASSOC_NAME | sky.list | <i>string</i> |
| Name of the ASSOC ASCII file. | | |
| ASSOC_PARAMS | 2,3,4 | <i>integers</i> ($2 \leq n \leq 3$) |
| # of the columns in the ASSOC file that will be used as coordinates and weight for cross-matching. | | |
| ASSOC_RADIUS | 2.0 | <i>float</i> |
| Search radius (in pixels) for ASSOC. | | |
| ASSOC_TYPE | MAG_SUM | <i>keyword</i> |
| Method for cross-matching in ASSOC: | | |
| FIRST | | – keep values corresponding to the first match found, |
| NEAREST | | values corresponding to the nearest match found, |
| MEAN | | weighted-average values, |
| MAG_MEAN | | exponentially weighted-average values, |
| SUM | | sum values, |
| MAG_SUM | | exponentially sum values, |
| MIN | | keep values corresponding to the match with minimum weight, |
| MAX | | keep values corresponding to the match with maximum weight. |
| ASSOCSELEC_TYPE | MATCHED | <i>keyword</i> |

What sources are printed in the output catalogue in case of ASSOC:

| | | | |
|-----------------|--|--|--------------------------|
| | ALL | all detections, | |
| | MATCHED | only matched detections, | |
| | -MATCHED | only detections that were not matched. | |
| BACK_FILTERSIZE | 3 | | <i>integers (n ≤ 2)</i> |
| | Size, or Width,Height (in background meshes) of the background-filtering mask. | | |
| BACK_SIZE | 64 | | <i>integers (n ≤ 2)</i> |
| | Size, or Width,Height (in pixels) of a background mesh. | | |
| BACK_TYPE | AUTO | | <i>keywords (n ≤ 2)</i> |
| | What background is subtracted from the images: | | |
| | AUTO | the internal, automatically interpolated back- | |
| | | ground map, | |
| | MANUAL | a user-supplied constant value provided in | |
| | | BACK_VALUE. | |
| BACK_VALUE | 0.0,0.0 | | <i>floats (n ≤ 2)</i> |
| | in BACK_TYPE MANUAL mode, the constant value to be subtracted from the images. | | |
| BACKPHOTO_THICK | 24 | | <i>integer</i> |
| | Thickness (in pixels) of the background LOCAL annulus. | | |
| BACKPHOTO_TYPE | GLOBAL | | <i>keyword</i> |
| | Background used to compute magnitudes: | | |
| | GLOBAL | taken directly from the background map, | |
| | LOCAL | recomputed in a “rectangular annulus” around the | |
| | | object. | |
| CATALOG_NAME | test.cat | | <i>string</i> |
| | Name of the output catalogue. If the name “STDOUT” is given and CATALOG_TYPE is set to ASCII, ASCII_HEAD, ASCII_SKYCAT, or ASCII_VOTABLE, the catalogue will be piped to the standard output (<i>stdout</i>) | | |
| CATALOG_TYPE | — | | <i>keyword</i> |
| | Format of output catalogue (note that ASCII is space and time consuming): | | |
| | ASCII | ASCII table, | |
| | ASCII_HEAD | as ASCII, preceded by a header containing infor- | |
| | | mation about the content, | |
| | ASCII_SKYCAT | SkyCat ASCII format (WCS coordinates re- | |
| | | quired), | |
| | ASCII_VOTABLE | XML-VOTable format, together with meta-data, | |
| | FITS_1.0 | FITS format as in SExtractor 1, | |
| | FITS_LDAC | FITS “LDAC” format (the original image header | |
| | | is copied). | |
| CHECKIMAGE_NAME | check.fits | | <i>strings (n ≤ 16)</i> |
| | File name for each “check-image”. | | |
| CHECKIMAGE_TYPE | NONE | | <i>keywords (n ≤ 16)</i> |

Type of information in the “check-images”:

| | | | |
|-----------------|---|--|-----------------------|
| | NONE | no check-image, | |
| | IDENTICAL | identical to input image (useful for converting formats), | |
| | BACKGROUND | full-resolution interpolated background map, | |
| | BACKGROUND_RMS | full-resolution interpolated background noise map, | |
| | MINIBACKGROUND | low-resolution background map, | |
| | MINIBACK_RMS | low-resolution background noise map, | |
| | -BACKGROUND | background-subtracted image, | |
| | FILTERED | background-subtracted filtered image (requires FILTER = Y), | |
| | OBJECTS | detected objects, | |
| | -OBJECTS | background-subtracted image with detected objects blanked, | |
| | APERTURES | MAG_APER and MAG_AUTO integration limits, | |
| | SEGMENTATION | display patches corresponding to pixels attributed to each object. | |
| CLEAN | Y | | <i>boolean</i> |
| | If true, the catalogue is “cleaned” (§8) before being written to disk. | | |
| CLEAN_PARAM | 1.0 | | <i>float</i> |
| | Efficiency of “cleaning”. | | |
| DEBLEND_MINCONT | 0.005 | | <i>float</i> |
| | Minimum contrast parameter for deblending. | | |
| DEBLEND_NTHRESH | 32 | | <i>integer</i> |
| | Number of deblending sub-thresholds. | | |
| DETECT_MINAREA | 5 | | <i>integer</i> |
| | Minimum number of pixels above threshold triggering detection. | | |
| DETECT_MAXAREA | 0 | | <i>integer</i> |
| | Maximum number of pixels above threshold triggering detection (0 for infinite). | | |
| DETECT_THRESH | 1.5 | | <i>floats (n ≤ 2)</i> |
| | Detection threshold. 1 argument: (ADUs or relative to Background RMS, see THRESH_TYPE). 2 arguments: μ (mag arcsec ⁻²), Zero-point (mag). | | |
| DETECT_TYPE | CCD | | <i>keyword</i> |
| | Type of device that produced the image: | | |
| | CCD | linear detector like CCDs or NICMOS, | |
| | PHOTO | photographic scan. | |
| FILTER | — | | <i>boolean</i> |
| | If true, filtering is applied to the data before extraction. | | |
| FILTER_NAME | — | | <i>string</i> |
| | Name of the file containing the filter definition. | | |

| | | | |
|-----------------------|---|--|--------------------------------|
| FILTER_THRESH | | | <i>floats</i> ($n \leq 2$) |
| | Lower and higher thresholds (in background standard deviations) for a pixel to be considered in filtering (used for retina-filtering only). | | |
| FITS_UNSIGNED | N | | <i>boolean</i> |
| | Force 16-bit FITS input data to be interpreted as unsigned integers. | | |
| FLAG_IMAGE | flag.fits | | <i>strings</i> ($n \leq 4$) |
| | File name(s) of the “flag-image(s)”. | | |
| FLAG_TYPE | OR | | <i>keyword</i> |
| | Combination method for flags on the same object: | | |
| | OR | arithmetical OR, | |
| | AND | arithmetical AND, | |
| | MIN | minimum of all flag values, | |
| | MAX | maximum of all flag values, | |
| | MOST | most common flag value. | |
| GAIN | | | <i>float</i> |
| | “Gain” (conversion factor in e^-/ADU) used for error estimates of CCD magnitudes . | | |
| INTERP_MAXXLG | 16 | | <i>integers</i> ($n \leq 2$) |
| | Maximum x gap (in pixels) allowed in interpolating the input image(s). | | |
| INTERP_MAXYLG | 16 | | <i>integers</i> ($n \leq 2$) |
| | Maximum y gap (in pixels) allowed in interpolating the input image(s). | | |
| INTERP_TYPE | ALL | | <i>keywords</i> ($n \leq 2$) |
| | Interpolation method from the variance map(s) (or weight map(s)): | | |
| | NONE | no interpolation, | |
| | VAR_ONLY | interpolate only the variance map (detection threshold), | |
| | ALL | interpolate both the variance map and the image itself. | |
| MAG_GAMMA | | | <i>float</i> |
| | γ of the emulsion (takes effect in PHOTO mode only). | | |
| MAG_ZEROPOINT | | | <i>float</i> |
| | Zero-point offset to be applied to magnitudes. | | |
| MASK_TYPE | CORRECT | | <i>keyword</i> |
| | Method of “masking” of neighbours for photometry: | | |
| | NONE | no masking, | |
| | BLANK | put detected pixels belonging to neighbours to zero, | |
| | CORRECT | replace by values of pixels symmetric with respect to the source center. | |
| MEMORY_BUFSIZE | 1024 | | <i>integer</i> |

Number of scan-lines in the image-buffer. Multiply by 4 the frame width to get equivalent memory space in bytes.

| | | |
|-----------------|--|--|
| MEMORY_OBJSTACK | 3000 | <i>integer</i> |
| | Maximum number of objects that the object-stack can contain. Multiply by 300 to get equivalent memory space in bytes. | |
| MEMORY_PIXSTACK | 300000 | <i>integer</i> |
| | Maximum number of pixels that the pixel-stack can contain. Multiply by 16 to 32 to get equivalent memory space in bytes. | |
| PARAMETERS_NAME | default.param | <i>string</i> |
| | The name of the file containing the list of parameters that will be computed and put in the catalogue for each object. | |
| PHOT_APERTURES | 5 | <i>floats (n ≤ 32)</i> |
| | Aperture diameters in pixels (used by MAG_APER). | |
| PHOT_AUTOPARAMS | 2.5,3.5 | <i>floats (n = 2)</i> |
| | MAG_AUTO controls: scaling parameter k of the 1st order moment, and minimum R_{min} (in units of A and B). | |
| PHOT_AUTOAPERS | 0.0,0.0 | <i>floats (n = 2)</i> |
| | MAG_AUTO minimum (circular) aperture diameters: estimation disk, and measurement disk. | |
| PHOT_FLUXFRAC | 0.5 | <i>floats (n ≤ 32)</i> |
| | Fraction of FLUX_AUTO defining each element of the FLUX_RADIUS vector. | |
| PIXEL_SCALE | 1.0 | <i>float</i> |
| | Pixel size in arcsec (for surface brightness parameters, FWHM and star/galaxy separation only). | |
| SATUR_LEVEL | 50000.0 | <i>float</i> |
| | Pixel value above which it is considered saturated. | |
| SEEING_FWHM | 1.2 | <i>float</i> |
| | FWHM of stellar images in arcsec (only for star/galaxy separation). | |
| STARNNW_NAME | default.nnw | <i>string</i> |
| | Name of the file containing the neural-network weights for star/galaxy separation. | |
| THRESH_TYPE | RELATIVE | <i>keywords (n ≤ 2)</i> |
| | Meaning of the DETECT_THRESH and ANALYSIS_THRESH parameters: | |
| | RELATIVE | scaling factor to the background RMS, |
| | ABSOLUTE | absolute level (in ADUs or in surface brightness). |
| VERBOSE_TYPE | NORMAL | <i>keyword</i> |
| | How much SExtractor comments its operations: | |
| | QUIET | run silently, |
| | NORMAL | display warnings and limited info concerning the work in progress, |

| | | | |
|--------------|---|--|-------------------------|
| | EXTRA_WARNINGS | like NORMAL, plus a few more warnings if necessary, | |
| | FULL | display a more complete information and the principal parameters of all the objects extracted. | |
| WEIGHT_GAIN | Y | | <i>boolean</i> |
| | If true, weight maps are considered as gain maps. | | |
| WEIGHT_IMAGE | weight.fits | | <i>strings (n ≤ 2)</i> |
| | File name of the detection and measurement “weight-image”, respectively. | | |
| WEIGHT_TYPE | NONE | | <i>keywords (n ≤ 2)</i> |
| | Weighting scheme (for single image, or detection and measurement images): | | |
| | NONE | no weighting, | |
| | BACKGROUND | variance map derived from the image itself, | |
| | MAP_RMS | variance map derived from an external RMS map, | |
| | MAP_VAR | external variance map, | |
| | MAP_WEIGHT | variance map derived from an external weight map, | |
| WRITE_XML | N | | <i>boolean</i> |
| | If true, meta-data will be written in XML-VOTable format. | | |
| XML_NAME | sex.xml | | <i>string</i> |
| | File name for the XML output of SExtractor. | | |

3.2 The catalogue parameter file

In addition to the configuration file detailed above, SExtractor needs a file containing the list of parameters that will be listed in the output catalogue for every detection. This allows the software to compute only catalogue parameters that are needed. The name of this catalogue-parameter file is traditionally suffixed with `.param`, and must be specified using the `PARAMETERS_NAME` config parameter. The full set of parameters can be queried with the command

```
% sex -dp
```

3.2.1 Format

The format of the catalogue parameter list is ASCII, and there must be *only one keyword per line*. Presently two kinds of keywords are recognized by SExtractor: scalars and vectors. Scalars, like `X_IMAGE`, yield single numbers in the output catalogue. Vectors, like `MAG_APER(4)` or `VIGNET(15,15)`, yield arrays of numbers. The order in which the parameters will be listed in the catalogue are the same as that of the keywords in the parameter list. Comments are allowed, they must begin with a “#”.

¹Optional parameter

3.2.2 Variants

For many catalogue parameters, especially those related to flux, position, or shape, several variants of the same measurement are available:

Fluxes may be expressed in linear (ADU) units or as Pogson (Pogson, 1856) magnitudes. Flux measurements in ADUs are prefixed with `FLUX_`, for example: `FLUX_AUTO`, `FLUX_ISO`, etc.. Magnitudes are prefixed with `MAG_` (e.g., `MAG_AUTO`, `MAG_ISO`,...). In `SEXTRACTOR` the magnitude m of a source is derived from its flux f :

$$m = \begin{cases} m_{ZP} - 2.5 \log_{10} f & \text{if } f > 0 \\ 99.0 & \text{otherwise,} \end{cases} \quad (3.1)$$

where m_{ZP} is the magnitude zero-point set with the `MAG_ZEROPOINT` configuration parameter.

Flux uncertainties follow a scheme similar to that of fluxes. Flux uncertainties are prefixed with `FLUXERR_`, as in `FLUXERR_AUTO` or `FLUXERR_ISO`. Magnitude uncertainties start with `MAGERR_`, for instance: `MAGERR_AUTO`, `MAGERR_ISO`,... Magnitude uncertainties σ_m are derived from the estimated 1- σ flux error σ_f :

$$\sigma_m = \begin{cases} (2.5/\ln 10)(\sigma_f/f) & \text{if } f > 0 \\ 99.0 & \text{otherwise.} \end{cases} \quad (3.2)$$

Positions and small distances can be expressed in image pixels, world coordinates, or in celestial coordinates. Measurements in units of image pixels are indicated by the suffix `_IMAGE`, for example: `Y_IMAGE`, `ERRWIN_IMAGE`,... Following the FITS convention, in `SEXTRACTOR` the center of the first image pixel has coordinates (1.0,1.0). Positions and small distances may also be expressed in so-called “world coordinates”, if World Coordinate System (WCS) metadata (Greisen & Calabretta, 2002) are present in the image FITS header.

3.3 Example of configuration

Chapter 4

Overview of the software

The complete analysis of an image is done in two passes through the data. During the first pass, a model of the sky background is built, and several global statistics are estimated. During the second pass, the image is background-subtracted, filtered and thresholded “on-the-fly”. Detections are then deblended, pruned (“CLEANed”), photometered, classified and finally written to the output catalogue. The following sections describe each of these operations in more detail.¹

¹In the text, uppercase keywords in typewriter font refer to parameters from the configuration file or from the parameter file

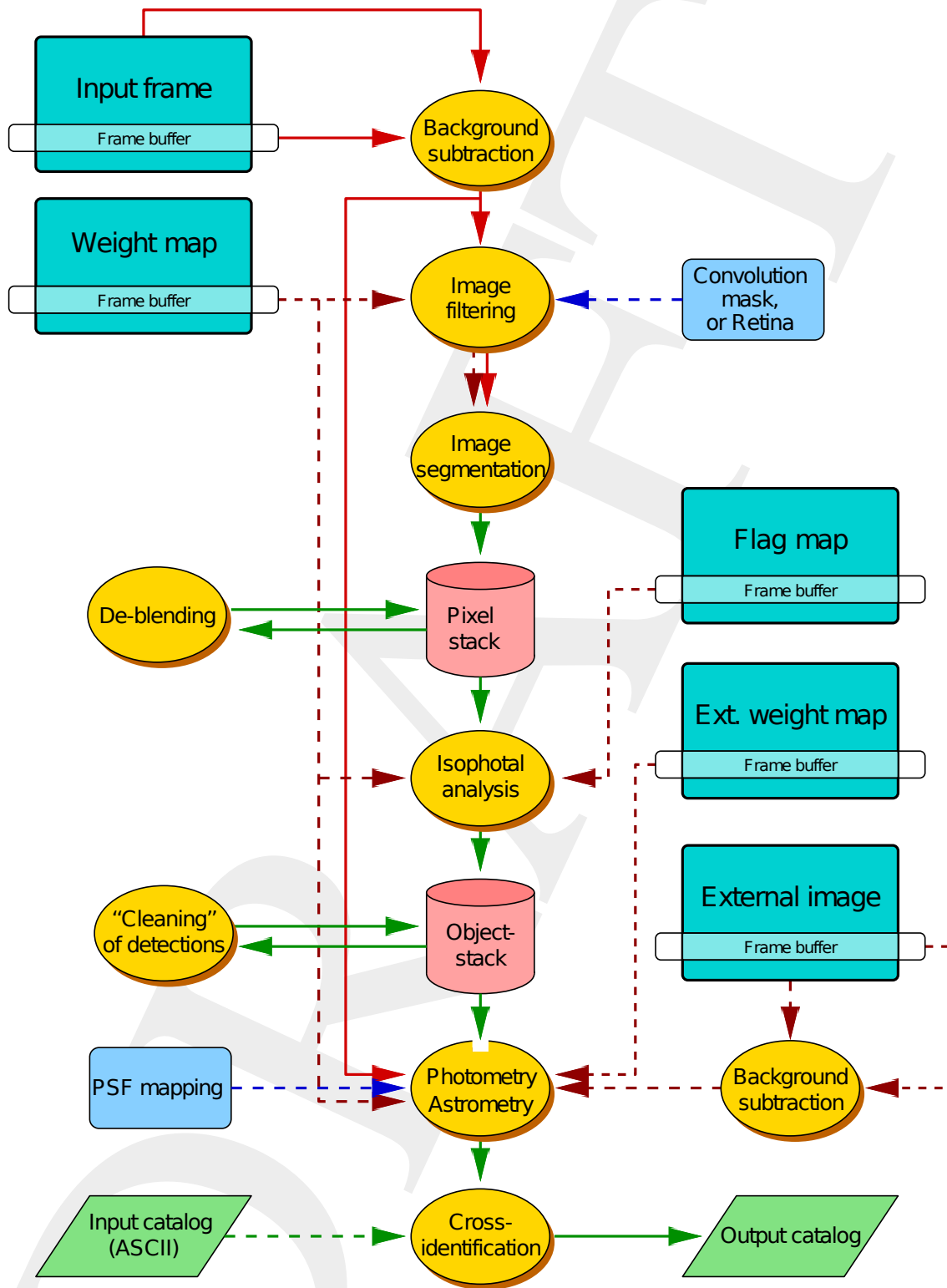


Figure 4.1: Layout of the main SEXTRACTOR procedures. *Dashed arrows* represent optional inputs.

Chapter 5

Handling of image data

5.1 Image format

SEXTRACTOR accepts images stored in FITS¹ format (Wells et al. 1981, see also <http://fits.gsfc.nasa.gov>). Both “Basic FITS” (one single header and one single body) and “Multi-Extension-FITS” (MEF) images are recognized. Binary SEXTRACTOR catalogues produced from MEF images are MEF files themselves. If catalogue output is in ASCII format, all catalogues from the individual extensions are concatenated in one big file; the `EXT_NUMBER` catalogue parameter must be used to tell which extension the source belongs to.

For images with `NAXIS > 2`, only the first data-plane is loaded. If WCS² information (Greisen & Calabretta 1995, <http://www.cv.nrao.edu/fits/documents/wcs/wcs.all.ps>) is available in the header, it is automatically used by SEXTRACTOR to compute astrometric parameters. Other astrometric descriptions like AST (*Starlink* format) or the solution coefficients of the DSS³ plates are not recognized by the software.

In SEXTRACTOR, as in all similar programs, FITS axis “1” is traditionally referred as the X axis, and FITS axis “2” as the Y axis.

5.2 Double image mode

If data are available for the same field in several photometric bands, it is usually desirable to measure object characteristics such as magnitudes exactly at the same positions and through the same apertures for the different bands, to derive precise colour indices for example. SEXTRACTOR makes this possible by providing a special mode dubbed “double image mode” where detections are made on one image while measurements are carried out on another (both images must have exactly the same number of pixels). By repeatedly running SEXTRACTOR with various “measurement images” while keeping the same “detection image”, one ends up with a set of catalogues having the same sources measured through different channels. The detection image will generally be chosen in the band where the data are the deepest. Alternatively, using a “ χ^2 image” (Szalay et al., 1999) (produced e.g., by SWARP) as a detection image, will allow most of the sources present in at least one channel to be detected and measured.

Double image mode is automatically engaged by providing SEXTRACTOR with two images in-

¹*Flexible Image Transport System*

²*World Coordinate System*

³*Digital Sky Survey*

stead of one:

```
% sex detection.fits,measurement.fits
```

A space may be used instead of a coma. In the example above, `sex detection.fits` is used as a detection image, while measurements are carried out on `measurement.fits`.

You then obtain two different catalogues for the two images, which have exactly the same numbers of lines and can be put together.

If you have images in other bands covering the same regions, you can measure them in double image mode as well, always taking as a reference image 1, and then combine all the catalogues obtained.

Chapter 6

Detection and segmentation

In SEXTRACTOR, the detection of sources is part of a process called *segmentation* in the image-processing vocabulary. Segmentation normally consists of identifying and separating image regions that have different properties (brightness, colour, texture...) or are delineated by edges. In the astronomical context, the segmentation process consists of separating objects from the sky background. This is however a somewhat imprecise definition, as astronomical sources have, on the images — and even often physically —, no clear boundaries, and may overlap. We shall therefore use the following working definition of an object in SEXTRACTOR: a group of pixels selected through some detection process and for which the flux contribution of an astronomical source is believed to be dominant over that of other objects. Note that this means that a simple x, y position vector alone cannot be handled by SEXTRACTOR as a detection: most measurement routines require some rough shape information about the objects.

Segmentation in SEXTRACTOR is achieved through a very simple thresholding process: a group of connected pixels that exceed some threshold above the background is identified as a detection. But things are a little bit more complicated in practice. First, on most astronomical images, the background is not constant over the frame, and its determination can be ambiguous in crowded regions. Second, the software has to operate on noisy data, and some filtering adapted to the characteristics of the image has to be applied prior to detection, to reduce the contamination by noise peaks. Third, many sources that overlap on the image are unlikely to be detected separately with a single detection threshold, and require a de-blending procedure, which is actually multi-thresholding in SEXTRACTOR. Each of these points will now be described in greater detail below. It is worth mentioning here that these 3 difficulties could, to a large extent, be bypassed using a wavelet decomposition (e.g. Bijaoui et al. 1998). Although such an algorithm might be implemented in a future version of SEXTRACTOR, current constraints in processing speed, available memory (processing of gigantic images) often make the “pedestrian approach” still more interesting in the case of large scale surveys.

6.1 Background estimation

The value measured at each pixel is a function of the sum of a “background” signal and light coming from the objects of interest. To be able to detect the faintest of these objects and also to measure accurately their fluxes, one needs to have an accurate estimate of the background level in any place of the image, a *background map*. Strictly speaking, there should be one background map per object, that is, what would the image look like if that object was absent. But, at least for detection, we may start by assuming that most discrete sources do not overlap too severely,

which is generally the case for high galactic latitude fields.

To construct the background map, SExtractor 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:

$$\text{Mode} = 2.5 \times \text{Median} - 1.5 \times \text{Mean} \quad (6.1)$$

This expression is different from the usual approximation

$$\text{Mode} = 3 \times \text{Median} - 2 \times \text{Mean} \quad (6.2)$$

(e.g. Kendall and Stuart 1977), but was found to be more accurate with our clipped distributions, from the simulations we made. Fig. 6.1 shows that the expression of the mode above is considerably less affected¹ 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. This is why we revert to the clipped mean in non-crowded fields.

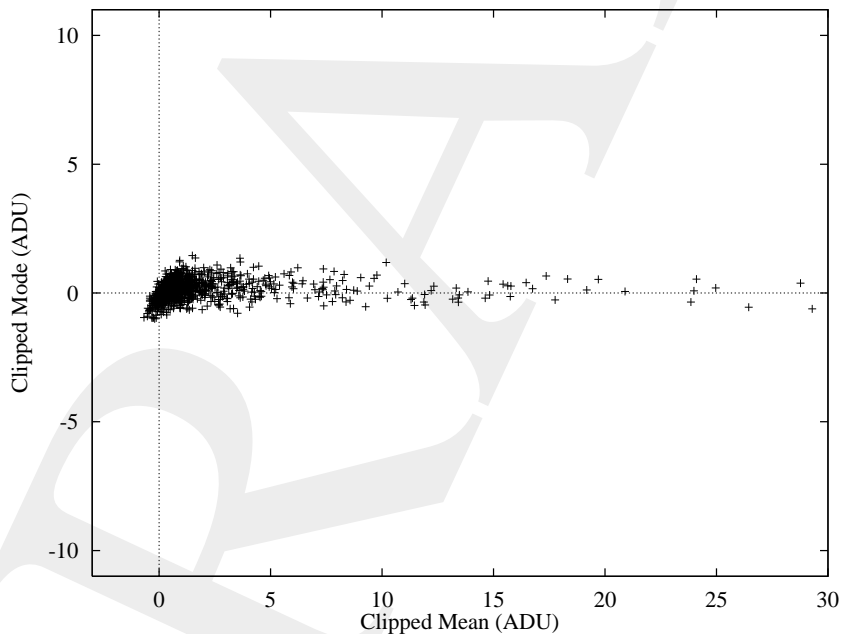


Figure 6.1: Simulations of 32×32 pixels background meshes polluted by random Gaussian profiles. The true background lies at 0 ADU. While being slightly noisier, the clipped “Mode” gives a more robust estimate than a clipped Mean in crowded regions.

Once the grid is set up, a median filter can be applied to suppress possible local overestimations due to bright stars. The resulting background map is then simply a (natural) bicubic-spline interpolation between the meshes of the grid. In parallel with the making of the background map, an *RMS background map*, that is, a map of the background noise in the image is produced. It will be used if the WEIGHT_TYPE parameter is set different from NONE (see §7.1).

¹Obviously in some very unfavorable cases (like small meshes falling on bright stars), it leads to totally inaccurate results.

6.1.1 Configuration parameters and tuning

. The choice of the mesh size (`BACK_SIZE`) is very important. If it is too small, the background estimation is affected by the presence of objects and random noise. Most importantly, part of the flux of the most extended objects can be absorbed into the background map. If the mesh size is too large, it cannot reproduce the small scale variations of the background. Therefore a good compromise has to be found by the user. Typically, for reasonably sampled images, a width² of 32 to 256 pixels works well. The user has some control over the background map by specifying the size of the median filter (`BACK_FILTERSIZE`). A width and height of 1 means that no filtering will be applied to the background grid. Usually a size of 3×3 is enough, 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 artefacts 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. It is important to note that all `BACK_` configuration parameters also affect the background-RMS map.

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_VALUE` parameter to be subtracted from the input image. The default value is 0.

6.1.2 CPU cost

. The background estimation operation can take a considerable time on the largest images, e.g. a few minutes for a 32000×32000 frame on a 2GHz processor.

6.2 Filtering

6.2.1 Convolution

Detectability is generally limited at the faintest flux levels by the background noise. The power-spectrum of the noise and that of the superimposed signal can be significantly different. Some gain in the ability to detect sources may therefore be obtained simply through appropriate linear filtering of the data, prior to segmentation. In low density fields, an optimal convolution kernel h (“matched filter”) can be found that maximizes detectability. An estimator of detectability is for instance the signal-to-noise ratio at source position $(x_0, y_0) \equiv (0, 0)$:

$$\left[\frac{S}{N} \right]^2 \equiv \frac{((s * h)(x_0, y_0))^2}{(n * h)^2}, \quad (6.3)$$

where s is the signal to be detected, n the noise, and ‘*’ the convolution operator. Moving to Fourier space, we get:

$$\left(\frac{S}{N} \right)^2 = \frac{(\int S \mathcal{H} d\omega)^2}{\int |\mathcal{N}|^2 |\mathcal{H}|^2 d\omega}, \quad (6.4)$$

²SEXTRACTOR offers the possibility of rectangular background meshes; but it is advised to use square ones, except in some very special cases (rapidly varying background in one direction for example).

where \mathcal{S} and \mathcal{H} are the Fourier-transforms of s and h , respectively, and $|\mathcal{N}|^2$ is the power-spectrum of the noise. Remarking, using Schwartz inequality, that

$$\left| \int \mathcal{S}\mathcal{H} d\omega \right|^2 \leq \int \frac{|\mathcal{S}|^2}{|\mathcal{N}|^2} d\omega \int |\mathcal{N}|^2 |\mathcal{H}|^2 d\omega, \quad (6.5)$$

we see that

$$\left(\frac{\mathcal{S}}{\mathcal{N}} \right)^2 \leq \int \frac{|\mathcal{S}|^2}{|\mathcal{N}|^2} d\omega. \quad (6.6)$$

Equality (maximum S/N) in (6.5) and (6.6) is achieved for

$$\frac{\mathcal{S}}{|\mathcal{N}|} \propto |\mathcal{N}|\mathcal{H}^*, \text{ that is} \quad (6.7)$$

$$\mathcal{H} \propto \frac{\mathcal{S}^*}{|\mathcal{N}|^2}. \quad (6.8)$$

In the case of white noise (a valid approximation for many astronomical images, especially CCD ones), $|\mathcal{N}|^2 = \text{cst}$; the optimal convolution kernel for detecting stars is then the *point spread function*³ (PSF) flipped over the x and y directions. It may also be described as the cross-correlation with the template of the sources to be detected (for more details see e.g., Bijaoui & Dantel (1970)).

There are of course a few problems with this method. First of all, many sources of unquestionable interest, like galaxies, appear in a variety of shapes and scales on astronomical images. A perfectly optimized detection routine should ultimately apply all relevant convolution kernels one after the other in order to make a complete catalogue. Approximations to this approach are the (isotropic) wavelet analysis mentioned earlier, or the more empirical ImCat algorithm (Kaiser et al. 1995), both of which assume that the sources are reasonably round. The impact on memory usage and processing speed of such refinements is currently judged too severe to be applied in SExtractor. Simple filtering does a good job in general: the topological constraints added by the segmentation process make the detection somewhat tolerant towards larger objects. Extended, very Low-Surface-Brightness (LSB) features found in astronomical images are often artifacts (flat-fielding errors, optical “ghosts” or halos). However, it is true that some of them can be genuine objects, like LSB galaxies, or distant galaxy clusters buried in the background noise. For detecting those with software like SExtractor, a specific processing is needed (see for instance Dalcanton et al. 1997 and references therein). The simplest way to achieve the detection of extended LSB objects in SExtractor is to work on MINIBACK check-images (see §6.1.1).

A second problem may occur because of overlaps with other objects. Convolution with a low-pass filter (the PSF has no negative side-lobes) diminishes the contrast between objects, and makes segmentation less effective in isolating individual sources. This can to some extent be recovered by deblending (see §6.4). In severely crowded fields however, confusion noise becomes the limiting factor for detection, and it is then advisable not to filter at all, or to use a bandpass-filter (compensated filter).

Finally, the PSF can vary across the field. The convolution mask should ideally follow these variations in order to allow for optimal detection everywhere in the image. However, considering approximately-Gaussian PSF cores and convolution kernels, detectability is a rather slow function of their FWHMs⁴: a mismatch as large as 50% between the kernel FWHM and that of

³The PSF is the convolution of the instrumental PSF and the atmospheric seeing.

⁴Full-Width at Half-Maximum

the PSF will lead to no more than a 10% loss in peak S/N (Irwin 1985). Considering that PSF variations are generally much smaller than this, filtering in SEXTRACTOR is limited to constant kernels.

6.2.2 Non-linear filtering

There are many situations in which convolution is of little help: filtering of (strongly) non-Gaussian noise, extraction of specific image patterns,... In those cases, one would like to extend the concept of a convolution kernel to that of a more general stationary filter, able for instance to mimic boolean-like operations on pixels. What one wants is thus a mapping from \mathbf{R}^n to \mathbf{R} around each pixel. But the more general the filter, the more difficult it is to design “by hand” for each case, specifying how input pixel #i should be taken into account with respect to input pixel #j to form the output, etc.. The solution to this is machine-learning. Given a training set containing input and output pixels, a machine-learning software will adapt its internal parameters in order to minimize a “cost function” (generally a χ^2 error) and converge toward the desired mapping-function. These parameters can then for example be reloaded by a “read-only” routine to provide the actual filtering.

SEXTRACTOR implements this kind of “read-only” functionality in the form of the so-called “retina filtering”. The EYE⁵ software (Bertin 1997) performs neural-network-learning on input and output images to produce “retina files”. These files contain weights that describe the behaviour of the neural network. The neural network can thus be seen as an “artificial retina” that takes its stimuli from a small rectangular array of pixels and produces a response according to prior learning (for more details, see the EYE documentation). Typical applications of the retina are the identification of glitches.

6.2.3 What is filtered, and what isn’t

Although filtering is a benefit for detection, it distorts profiles and correlates the noise; it is therefore detrimental for most measurement tasks. Because of this, filtering is applied “on the fly” to the image, and *directly* affects only the detection process and the isophotal parameters described in §11.1. Other catalogue parameters are indirectly affected — through the exact position of the barycenter and typical object extent —, but the effect is considerably less. Obviously, in double-image mode, filtering is only applied to the *detection* image.

6.2.4 Image boundaries and bad pixels

“Virtual” pixels that lie outside image boundaries are arbitrarily set to zero. This makes sense since filtering occurs on a background-subtracted image. When weighting is applied (§7), bad pixels (pixels with weight < WEIGHT_THRESH) are interpolated by default (§7.5) and should therefore not cause much trouble. It is recommended not to turn-off interpolation of bad pixels when filtering is on.

6.2.5 Configuration parameters

Filtering is triggered when the FILTER keyword is set to Y. If active, a file with name specified by FILTER_NAME is searched for and loaded. Filtering with large retinas can be extremely time

⁵Enhance Your Extraction

consuming. In many cases, one is only interested in filtering pixels whose values stand out from the background noise. The `FILTER_THRESH` keyword can be given to specify the range of pixel values within which retina-filtering will be applied, in units of background noise standard deviation. If one value is given, it is interpreted as a lower threshold. For instance:

```
FILTER_THRESH 3.0
```

will allow filtering for pixel values exceeding $+3\sigma$ above the local background, whereas

```
FILTER_THRESH -10.0,3.0
```

will only allow filtering for pixel values between -10σ and $+3\sigma$. `FILTER_THRESH` has no effect on convolution.

The result of the filtering process can be verified through a `FILTERED` check-image: see §13.

6.2.6 CPU cost

The `SEXTRACTOR` filtering routine is particularly optimized for small kernels. It thus provides a convenient way of filtering large image data. On a 2GHz machine, a convolution by a 5×5 kernel will contribute less than 1 second to the processing time of a 2048×4096 image. The numbers for non-linear (retina) filtering depend on the complexity of the neural network, but can be a hundred times larger.

6.2.7 Filter file formats

As described above, two kinds of filter files are recognized by `SEXTRACTOR`: convolution files (traditionally suffixed with “.conv”), and “retina” files (“.ret” extensions⁶).

Retina files are written exclusively by the `EYE` software, as FITS binary-tables.

Convolution files are in ASCII format. The following example shows the content of the `gauss_2.0_5x5.conv` file which can be found in the `config/` sub-directory of the `SEXTRACTOR` distribution:

```
CONV NORM
# 5x5 \index{convolution} convolution mask of a gaussian \index{PSF} PSF with \index{FWHM} FWHM
0.006319 0.040599 0.075183 0.040599 0.006319
0.040599 0.260856 0.483068 0.260856 0.040599
0.075183 0.483068 0.894573 0.483068 0.075183
0.040599 0.260856 0.483068 0.260856 0.040599
0.006319 0.040599 0.075183 0.040599 0.006319
```

The `CONV` keyword appearing at the beginning of the first line tells `SEXTRACTOR` that the file contains the description of a convolution mask (kernel). It can be followed by `NORM` if the mask is to be normalized to 1 before being applied, or `NONORM` otherwise⁷. The following lines should contain an equal number of kernel coefficients, separated by `<space>` or `<TAB>` characters. Coefficients in the example above are read from left to right and top to bottom, corresponding to increasing `NAXIS1` (x) and `NAXIS2` (y) in the image. Formatting is free, and number representations like `-0.14`, `-0.1400`, `-1.4e-1` or `-1.4E-01` are equivalent. The width of the kernel is set by the number of values per line, and its height is given by the number of lines. Lines beginning with “#” are treated as comments.

⁶In `SEXTRACTOR`, file name extensions are just conventions; they are not used by the software to distinguish between different file formats.

⁷If the sum of the kernel coefficients happens to be exactly zero, the kernel is normalized to variance unity.

6.3 Thresholding

Thresholding is applied to the background-subtracted, filtered image to isolate connected groups of pixels. Each group defines the approximate position and shape of a basic SExtractor detection that will be processed further in the pipeline. Groups are made of pixels whose values exceed the local threshold and which touch each other at their sides or angles (“8-connectivity”).

6.3.1 Configuration parameters.

Thresholding is mostly controlled through the `DETECT_THRESH`, `DETECT_MINAREA` and `DETECT_MAXAREA` keywords.

`DETECT_THRESH` sets the threshold value. If one single value is given, it is interpreted as a threshold in units of the (unfiltered) background’s standard deviation. For example:

```
DETECT_THRESH 1.5
```

will set the detection threshold at 1.5σ above the local background. It is important to note that *the standard deviation quoted here is that of the unFILTERed image, at the pixel scale*. Hence, on images with white Gaussian background noise for instance, a `DETECT_THRESH` of 3.0 will be close to optimum if low-pass FILTERing is turned off, but sub-optimum (too high) if it is on. On the contrary, if the background noise of the image is intrinsically correlated from pixel-to-pixel, a `DETECT_THRESH` of 3.0 (with no FILTERing) will be too low and will result in a poor reliability of the extracted catalogue.

Two numbers can be given as arguments to `DETECT_THRESH`, in which case the first one is interpreted as an absolute threshold in units of “magnitudes per square-arcsecond”, and the second as a zero-point in the same units.

```
DETECT_THRESH 27.2,30.0
```

will for example set the threshold at $10^{-0.4(27.2-30)} = 13.18$ ADUs above the local background.

`DETECT_MINAREA` sets the minimum number of pixels a group should have to trigger a detection. Obviously this parameter can be used just like `DETECT_THRESH` to detect only bright and “big” sources, or to increase detection reliability. It is however more tricky to manipulate at low detection thresholds because of the complex interplay of object topology, noise correlations (including those induced by filtering), and sampling. In most cases it is therefore recommended to keep `DETECT_MINAREA` at a small value, typically 1 to 5 pixels, and let `DETECT_THRESH` and the filter define SExtractor’s sensitivity.

`DETECT_MAXAREA`, on the other hand, sets the maximum number of pixels a group must have in order to trigger a detection. Thus, this parameter may be used in conjunction with `DETECT_MINAREA` in order to detect only objects whose size is within a certain range. Note that, although large objects may be removed from the catalogue by filtering out those with `ISOAREA_IMAGE` larger than some threshold, these detections would still appear in the check-image. If it is required that large objects be not present in it, `DETECT_MAXAREA` should be used in order to effectively exclude them from the check-image. See fig. 6.2 for an example.

6.4 Deblending

Each time an object extraction is completed, the connected set of pixels passes through a filter that tries to split it into eventual overlapping components. This case appears more frequently

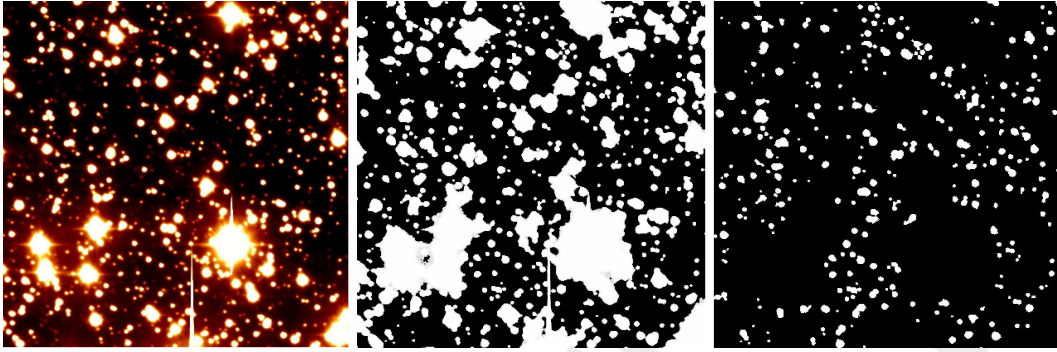


Figure 6.2: Example of how the `DETECT_MAXAREA` parameter can be used in order not to detect objects larger than a determined number of pixels. *Left*: close-up of the original image. *Center*: `OBJECTS` check-image generated without `DETECT_MAXAREA`. *Right*: the same `OBJECTS` check-image, when generated with `DETECT_MAXAREA = 100`.

when the field is crowded or when the detection threshold is set very low. The deblending method adopted in `SExtractor`, is based on *multi-thresholding*, and works on any kind of object; but it is unable to deblend components that are so close that no saddle is present in their profile. However, as no assumption has to be made on the shape of the objects, it is perfectly suited for galaxies as well as for high galactic latitude stellar fields.

Typical problematic cases for deblending include patchy, extended **Sc** galaxies (which must be considered as single entities), and close or interacting pairs of optically faint galaxies (which should be considered as separate objects). Basically, the multi-thresholding algorithm employs a multiple isophotal analysis technique similar to those in use at the APM and the COSMOS machines Beard et al. (1990); in a first pass, each extracted set of connected pixels is re-thresholded at N levels linearly or exponentially spaced between its primary extraction threshold and its peak value. This gives us a 2-dimensional “model” of the light distribution within the object(s), which is stored in the form of a tree structure (fig. 6.3). Then the algorithm goes downwards, from the tips of branches to the trunk, and decides at each junction whether it shall extract two (or more) objects or continue its way down. To meet the conditions described earlier, the following simple decision criteria are adopted: at any junction threshold t_i , any branch will be considered as a separate component if

- (1) the integrated pixel intensity (above t_i) of the branch is greater than a certain fraction δ_c of the total intensity of the composite object;
- (2) condition (1) is verified for at least one more branch at the same level i .

Note that ideally, condition (1) is both flux- and scale-invariant. However for faint, poorly resolved objects, the efficiency of the deblending is limited mostly by seeing and sampling. From the analysis of both small and extended galaxy images, a compromise value for the contrast parameter $\delta_c \sim 0.005$ proved to be optimum. This should normally separate objects with a difference in magnitude greater than ≈ 6 .

The outlying pixels with flux lower than the separation thresholds have to be reallocated to the proper components of the merger. To do so, we have opted for a *statistical* approach: at each faint pixel, we compute the contribution expected from each sub-object, using a bivariate Gaussian fit to its profile, and then derive the probability for that pixel to belong to the sub-object. For instance, a faint pixel lying halfway between two close bright stars having the same magnitude will be appended to one of these with equal probabilities. One important advantage

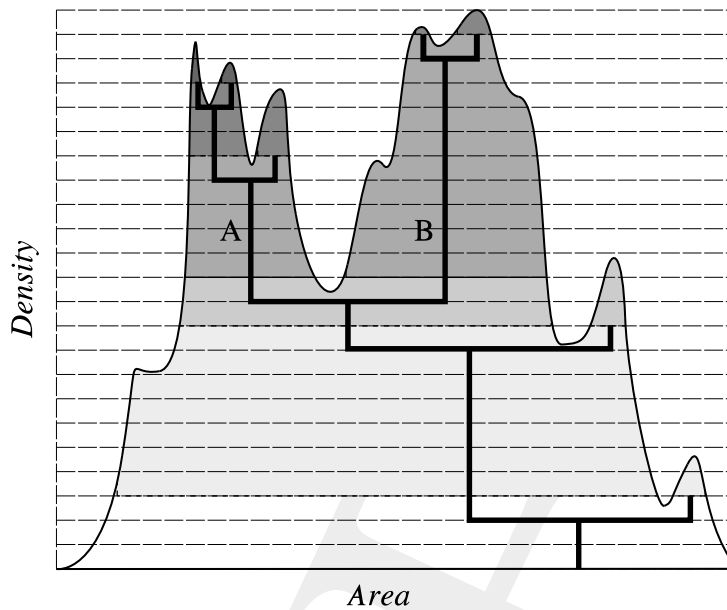


Figure 6.3: A schematic diagram of the method used to deblend a composite object. The area profile of the object (*smooth curve*) can be described in a tree-structured way (*thick lines*). The decision to regard or not a branch as a distinct object is determined according to its relative integrated intensity (*tinted area*). In that case above, the original object shall split into two components A and B. Remaining pixels are assigned to their most credible “progenitors” afterwards.

of this technique is that the morphology of any object is completely defined simply through its list of pixels.

To test the effects of deblending on photometry and astrometry measurements, we made several simulations of photographic images of double stars with different separations and magnitudes under typical observational conditions (fig. 6.4). It is obvious that multiple isophotal techniques fail when there is no saddle point present in profiles (i.e. for distance between stars $< 2\sigma$ in the case of Gaussian images). We measured a magnitude error ≤ 0.2 mag and a shift of the centroid (≤ 0.4 pixels) for the fainter star in the very worst cases, but no other systematic effects were noticeable.

The user can control the multi-thresholding operation through 3 parameters. The first one is the number of deblending thresholds (`DEBLEND_NTHRESH`). A good value is 32. Higher values are generally useless, except perhaps for images having an unusually high dynamic range. In case of memory problems, decreasing the number of thresholds to say, 8 or even less may be a solution. But then, of course, a degradation of the deblending performances may occur. The second parameter is the contrast parameter (`DEBLEND_MINCONT`). As described above, values from 0.001 to 0.01 give the best results. Putting `DEBLEND_MINCONT` to 0 means that even the faintest local peaks in the profile will be considered as separate objects. Putting it to 1 means that no deblending will be authorized. The last parameter concerns the kind of scale used for the thresholds. If the image comes from photographic material, then a linear scale has to be used (`DETECTION_TYPE PHOTO`). Otherwise, for an image obtained with a linear device like a CCD, an exponential scale is more appropriate (`DETECTION_TYPE CCD`).

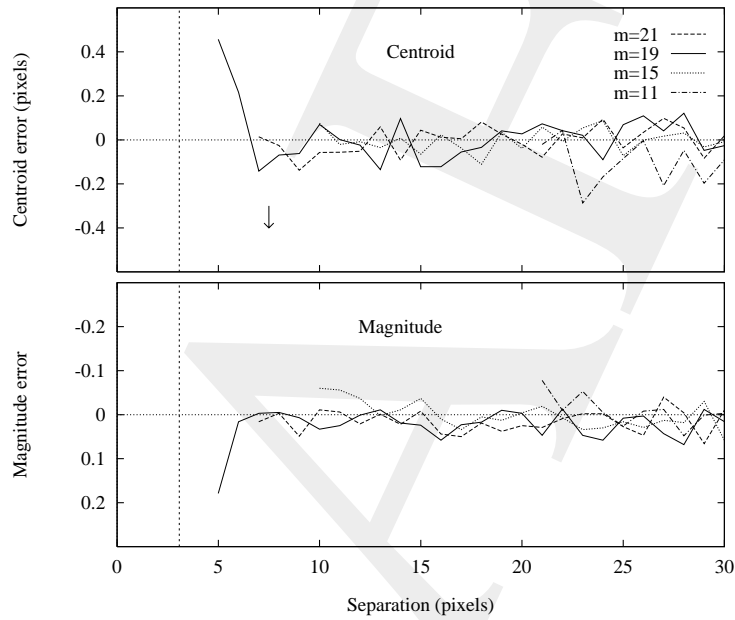


Figure 6.4: Centroid and corrected isophotal magnitude errors for a simulated 19^{th} magnitude star blended with a 11, 15, 19 and 21^{th} mag. companion as a function of distance (expressed in pixels). Lines stop at the left when the objects are too close to be deblended. The *dashed vertical line* is the theoretical limit for unsaturated stars with equal magnitudes. In the centroid plot, the *arrow* indicates the direction of the neighbour. The simulation assumes a 1 hour exposure with the CERGA telescope on a IIIaJ plate and Moffat profiles with a seeing FWHM of 3 pixels ($2''$).

Chapter 7

Weighting

The noise level in astronomical images is often fairly constant, that is, constant values for the gain, the background noise and the detection thresholds can be used over the whole frame. Unfortunately, in some cases, like strongly vignetted or composited images, this approximation is no longer good enough. This leads to detecting clusters of detected noise peaks in the noisiest parts of the image, or missing obvious objects in the most sensitive ones. SExtractor is able to handle images with variable noise. It does so through *weight maps*, which are frames having the same size as the images where objects are detected or measured, and which describe the noise intensity at each pixel. These maps are internally stored in units of *absolute variance* (in ADU²). We employ the generic term “weight map” because these maps can also be interpreted as quality index maps: infinite variance ($\geq 10^{30}$ by definition in SExtractor) means that the related pixel in the science frame is totally unreliable and should be ignored. The variance format was adopted as it linearizes most of the operations done over weight maps (see below).

This means that the noise covariances between pixels are ignored. Although raw CCD images have essentially white noise, this is not the case for warped images, for which resampling may induce a strong correlation between neighbouring pixels. In theory, all non-zero covariances within the geometrical limits of the analysed patterns should be taken into account to derive thresholds or error estimates. Fortunately, the correlation length of the noise is often smaller than the patterns to be detected or measured, and constant over the image. In that case one can apply a simple “fudge factor” to the estimated variance to account for correlations on small scales. This proves to be a good approximation in general, although it certainly leads to underestimations for the smallest patterns.

7.1 Weight map formats

SExtractor accepts in input, and converts to its internal variance format, several types of weight maps. This is controlled through the WEIGHT_TYPE configuration keyword. These weight maps can either be read from a FITS file, whose name is specified by the WEIGHT_IMAGE keyword, or computed internally. Valid WEIGHT_TYPES are:

- **NONE:** No weighting is applied. The related WEIGHT_IMAGE and WEIGHT_THRESH (see below) parameters are ignored.
- **BACKGROUND:** the science image itself is used to compute internally a variance map (the related WEIGHT_IMAGE parameter is ignored). Robust (3σ -clipped) variance estimates are

first computed within the same background meshes as those described in §6.1¹. The resulting low-resolution variance map is then bicubic-spline-interpolated on the fly to produce the actual full-size variance map. A check-image with CHECKIMAGE_TYPE MINIBACK_RMS can be requested to examine the low-resolution variance map.

- **MAP_RMS**: the FITS image specified by the WEIGHT_IMAGE file name must contain a weight map in units of absolute standard deviations (in ADUs per pixel).
- **MAP_VAR**: the FITS image specified by the WEIGHT_IMAGE file name must contain a weight map in units of relative variance. A robust scaling to the appropriate absolute level is then performed by comparing this variance map to an internal, low-resolution, absolute variance map built from the science image itself.
- **MAP_WEIGHT**: the FITS image specified by the WEIGHT_IMAGE file name must contain a weight map in units of relative weights. The data are converted to variance units (by definition variance $\propto 1/\text{weight}$), and scaled as for MAP_VAR. MAP_WEIGHT is the most commonly used type of weight map: a flat-field, for example, is generally a good approximation to a perfect weight map.

7.2 Weight threshold

It may happen, that some weights are too low (or variances too high) to be of any interest: it is then more appropriate to discard such pixels than to include them in unweighted measurements such as FLUX_APER. To allow discarding these very bad pixels, a threshold can be set with the WEIGHT_THRESH parameter. The unit in which this threshold should be expressed is that of input data: ADUs for BACKGROUND and MAP_RMS maps, uncalibrated ADUs² for MAP_VAR, and uncalibrated weight-values for MAP_WEIGHT maps. Depending on the weight map type, the threshold will set a lower or a higher limit for “bad pixel” values: higher for weights, and lower for variances and standard deviations. The default value is 0 for weights, and 10^{30} for variance and standard deviation maps.

7.3 Effects of weighting

Weight maps modify the working of SExtractor in the following respects:

1. Bad pixels are discarded from the background statistics. If more than 50% of the pixels in a background mesh are bad, the local background value and its standard deviation are replaced by interpolation of the nearest valid meshes.
2. The detection threshold t above the local sky background is adjusted for each pixel i with variance σ_i^2 : $t_i = \text{DETECT_THRESH} \times \sqrt{\sigma_i^2}$, where DETECT_THRESH is expressed in units of standard deviations of the background noise. Pixels with variance above the threshold set with the WEIGHT_THRESH parameter are therefore simply not detected. This may result in splitting objects crossed by a group of bad pixels. Interpolation (see §7.5) should be used to avoid this problem. If convolution filtering (§6.2.1) is applied for detection, the variance map is convolved too. This yields optimum scaling of the detection threshold in the case where noise is uncorrelated from pixel to pixel. Non-linear filtering operations (like those offered by artificial retinæ, §6.2.2) are not affected.

¹The mesh-filtering procedures act on the variance map, too.

3. The CLEANing process (§8) takes into account the exact individual thresholds assigned to each pixel for deciding about the fate of faint detections.
4. Error estimates like `FLUXISO_ERR`, `ERRA_IMAGE`, ... make use of individual variances too. The local background noise standard deviation is simply set to $\sqrt{\sigma_i^2}$. In addition, if the `WEIGHT_GAIN` parameter is set to `Y` — which is the default —, it is assumed that the local pixel gain (i.e., the conversion factor from photo-electrons to ADUs) is inversely proportional to σ_i^2 , its median value over the image being set by the `GAIN` configuration parameter. In other words, the changes in noise intensities seen over the images are assumed to be caused by spatial variations in the gain. This is the most common case: correction for vignetting, or coverage depth. When this is not the case, for instance when changes are purely dominated by those of the read-out noise, `WEIGHT_GAIN` shall be set to `N`.
5. Finally, pixels with weights beyond `WEIGHT_THRESH` are treated just like pixels discarded by the MASKing process (§9).

7.4 Combining weight maps

All the weighting options listed in §7.1 can be applied separately to detection and measurement images (§3), — even if some combinations may not always make sense. For instance, the following set of configuration lines:

```
WEIGHT_IMAGE rms.fits,weight.fits
WEIGHT_TYPE MAP_RMS,MAP_WEIGHT
```

will load the FITS file `rms.fits` and use it as an RMS map for adjusting the detection threshold and CLEANing, while the `weight.fits` weight map will only be used for scaling the error estimates on measurements. This can be done in single- as well as in dual-image mode (§3). `WEIGHT_IMAGES` can be ignored for BACKGROUND `WEIGHT_TYPES`. It is of course possible to use weight maps for detection or for measurement only. The following configuration:

```
WEIGHT_IMAGE weight.fits
WEIGHT_TYPE NONE,MAP_WEIGHT
```

will apply weighting only for measurements; detection and CLEANing operations will remain unaffected.

7.5 Interpolation

TO BE WRITTEN

Chapter 8

Cleaning

TO BE WRITTEN

DRAFT

Chapter 9

Masking

TO BE WRITTEN

DRAFT

Chapter 10

Flags

A set of both *internal* and *external* flags is accessible for each object. Internal flags are produced by the various detection and measurement processes within SEXTRACTOR; they tell for instance if an object is saturated or has been truncated at the edge of the image. External flags come from *flag maps*: these are images with the same size as the one where objects are detected, where integer numbers can be used to flag some pixels (for instance, “bad” or noisy pixels). Different combinations of flags can be applied within the isophotal area that defines each object, to produce a unique value that will be written to the catalogue.

10.1 Internal flags

Internal flags contain, coded in decimal, various flag bits as a sum of powers of 2.

10.1.1 Extraction flags: FLAGS

This 16-bit flag parameter contains basic warnings about the source extraction process, in order of increasing concern.

- 1 The object has neighbours, bright and close enough to significantly bias MAG_AUTO photometry¹, or bad pixels (if more than 10% of the integrated area is affected).
- 2 The object was originally blended with another one.
- 4 At least one object pixel is saturated (or very close to).
- 8 The isophotal footprint of the detected object is truncated (too close to an image boundary).
- 16 Object's aperture data are incomplete or corrupted.
- 32 Object's isophotal data are incomplete or corrupted².
- 64 A memory overflow occurred during deblending.
- 128 A memory overflow occurred during extraction.

For example, an object close to an image border may have `FLAGS = 16`, and perhaps `FLAGS = 8+16+32 = 56`.

¹This flag can be activated only when `MAG_AUTO` magnitudes (§11.4) are requested.

²This flag is inherited from SEXTRACTOR V1.0, and has been kept for compatibility reasons. With SEXTRACTOR V2.0+, having this flag activated doesn't have any consequence for the extracted parameters.

10.1.2 Weight-map flags: FLAGS_WEIGHT

This 16-bit flag parameter contains warnings related to the pixel weighting process.

- 1 The isophotal footprint of the detected object overlaps at least one low weight³ in the measurement image.
- 2 The isophotal footprint of the detected object contains or touches at least one low weight in the filtered detection image.

10.1.3 Weight-map flags: FLAGS_WIN

This 16-bit flag parameter contains warnings about *_WIN measurements (§11.2).

- 1 Second-order moments measured through the Gaussian window are inconsistent ($\overline{x^2y^2} - \bar{x}\bar{y} \leq 0$).
- 2 Second-order moments measured through the Gaussian window are negative or zero.
- 4 Flux integrated within the Gaussian window is negative or zero.

10.2 External flags

SEXTRACTOR understands that it must process external flags when IMAFLAGS_ISO or NIMAFLAGS_ISO are present in the catalogue parameter file. It then looks for a FITS image specified by the FLAG_IMAGE keyword in the configuration file. The FITS image must contain the flag map, in the form of a 2-dimensional array of 8, 16 or 32 bits integers. It must have the same size as the image used for detection. Such flag maps can be created using for example the **WeightWatcher** software (Bertin 1997).

The flag map values for pixels that coincide with the isophotal area of a given detected object are then combined, and stored in the catalogue as the long integer IMAFLAGS_ISO. 5 kinds of combination can be selected using the FLAG_TYPE configuration keyword:

- OR: the result is an arithmetic (bit-to-bit) **OR** of flag map pixels.
- AND: the result is an arithmetic (bit-to-bit) **AND** of non-zero flag map pixels.
- MIN: the result is the minimum of the (signed) flag map pixels.
- MAX: the result is the maximum of the (signed) flag map pixels.
- MOST: the result is the most frequent non-zero flag map pixel-value.

The NIMAFLAGS_ISO catalogue parameter contains a number of relevant flag map pixels: the number of non-zero flag map pixels in the case of an OR or AND FLAG_TYPE, or the number of pixels with value IMAFLAGS_ISO if the FLAG_TYPE is MIN,MAX or MOST.

³the term "low weight" refers to a weight that falls below the threshold set by the WEIGHT_THRESH configuration parameter.

Chapter 11

Measurements

Once sources have been detected and deblended, they enter the measurement phase. SEXTRACTOR performs two categories of measurements. Measurements from the first category are made on the isophotal object profiles. Only pixels above the detection threshold are considered. Many of these isophotal measurements (like X_IMAGE, Y_IMAGE, etc.) are necessary for the internal operations of SEXTRACTOR and are therefore executed even if they are not requested. Measurements from the second category have access to all pixels of the image. These measurements are generally more sophisticated and are done at a later stage of the processing (after CLEANing and MASKing).

11.1 Positional parameters derived from the isophotal profile

The following parameters are derived from the spatial distribution \mathcal{S} of pixels detected above the extraction threshold. *The pixel values I_i are taken from the (filtered) detection image.*

Note that, unless otherwise noted, all parameter names given below are only prefixes. They must be followed by "_IMAGE" if the results shall be expressed in pixel units (see §.), or "_WORLD" for World Coordinate System (WCS) units (see §11.3). For example: THETA \rightarrow THETA_IMAGE. In all cases, parameters are first computed in the image coordinate system, and then converted to WCS if requested.

11.1.1 Limits: XMIN, YMIN, XMAX, YMAX

These coordinates define two corners of a rectangle which encloses the detected object:

$$\text{XMIN} = \min_{i \in \mathcal{S}} x_i, \quad (11.1)$$

$$\text{YMIN} = \min_{i \in \mathcal{S}} y_i, \quad (11.2)$$

$$\text{XMAX} = \max_{i \in \mathcal{S}} x_i, \quad (11.3)$$

$$\text{YMAX} = \max_{i \in \mathcal{S}} y_i, \quad (11.4)$$

where x_i and y_i are respectively the x-coordinate and y-coordinate of pixel i .

11.1.2 Barycenter: X, Y

Barycenter coordinates generally define the position of the “center” of a source, although this definition can be inadequate or inaccurate if its spatial profile shows a strong skewness or very large wings. X and Y are simply computed as the first order moments of the profile:

$$X = \bar{x} = \frac{\sum_{i \in S} I_i x_i}{\sum_{i \in S} I_i}, \quad (11.5)$$

$$Y = \bar{y} = \frac{\sum_{i \in S} I_i y_i}{\sum_{i \in S} I_i}. \quad (11.6)$$

In practice, x_i and y_i are summed relative to XMIN and YMIN in order to reduce roundoff errors in the summing.

11.1.3 Position of the peak: XPEAK, YPEAK

It is sometimes useful to have the position XPEAK, YPEAK of the pixel with maximum intensity in a detected object, for instance when working with likelihood maps, or when searching for artifacts. For better robustness, PEAK coordinates are computed on *filtered* profiles if available. On symmetrical profiles, PEAK positions and barycenters coincide within a fraction of pixel (XPEAK and YPEAK coordinates are quantized by steps of 1 pixel, thus XPEAK_IMAGE and YPEAK_IMAGE are integers). This is no longer true for skewed profiles, therefore a simple comparison between PEAK and barycenter coordinates can be used to identify asymmetrical objects on well-sampled images.

11.1.4 2nd order moments: X2, Y2, XY

(Centered) second-order moments are convenient for measuring the spatial spread of a source profile. In SExtractor they are computed with:

$$X2 = \overline{x^2} = \frac{\sum_{i \in S} I_i x_i^2}{\sum_{i \in S} I_i} - \bar{x}^2, \quad (11.7)$$

$$Y2 = \overline{y^2} = \frac{\sum_{i \in S} I_i y_i^2}{\sum_{i \in S} I_i} - \bar{y}^2, \quad (11.8)$$

$$XY = \overline{xy} = \frac{\sum_{i \in S} I_i x_i y_i}{\sum_{i \in S} I_i} - \bar{x} \bar{y}, \quad (11.9)$$

These expressions are more subject to roundoff errors than if the 1st-order moments were subtracted before summing, but allow both 1st and 2nd order moments to be computed in one pass.

Roundoff errors are however kept to a negligible value by measuring all positions relative here again to XMIN and YMIN.

11.1.5 Basic shape parameters: A, B, THETA

These parameters are intended to describe the detected object as an elliptical shape. A and B are its semi-major and semi-minor axis lengths, respectively. More precisely, they represent the maximum and minimum spatial *rms* dispersion of the object profile along any direction. THETA is the position-angle of the A axis relative to the first image axis. It is counted positive in the direction of the second axis. Here is how they are computed:

2nd-order moments can easily be expressed in a referential rotated from the x, y image coordinate system by an angle $+\theta$:

$$\begin{aligned} \overline{x_\theta^2} &= \cos^2 \theta \overline{x^2} + \sin^2 \theta \overline{y^2} - 2 \cos \theta \sin \theta \overline{xy}, \\ \overline{y_\theta^2} &= \sin^2 \theta \overline{x^2} + \cos^2 \theta \overline{y^2} + 2 \cos \theta \sin \theta \overline{xy}, \\ \overline{xy_\theta} &= \cos \theta \sin \theta \overline{x^2} - \cos \theta \sin \theta \overline{y^2} + (\cos^2 \theta - \sin^2 \theta) \overline{xy}. \end{aligned} \quad (11.10)$$

One can find interesting angles θ_0 for which the variance is minimized (or maximized) along x_θ :

$$\left. \frac{\partial \overline{x_\theta^2}}{\partial \theta} \right|_{\theta_0} = 0, \quad (11.11)$$

which leads to

$$2 \cos \theta \sin \theta_0 (\overline{y^2} - \overline{x^2}) + 2(\cos^2 \theta_0 - \sin^2 \theta_0) \overline{xy} = 0. \quad (11.12)$$

If $\overline{y^2} \neq \overline{x^2}$, this implies:

$$\tan 2\theta_0 = 2 \frac{\overline{xy}}{\overline{x^2} - \overline{y^2}}, \quad (11.13)$$

a result which can also be obtained by requiring the covariance $\overline{xy_{\theta_0}}$ to be null. Over the domain $[-\pi/2, +\pi/2[$, two different angles — with opposite signs — satisfy (11.13). By definition, THETA is the position angle for which $\overline{x_\theta^2}$ is *maximized*. THETA is therefore the solution to (11.13) that has the same sign as the covariance \overline{xy} . A and B can now simply be expressed as:

$$A^2 = \overline{x_{\text{THETA}}^2}, \quad \text{and} \quad (11.14)$$

$$B^2 = \overline{y_{\text{THETA}}^2}. \quad (11.15)$$

A and B can be computed directly from the 2nd-order moments, using the following equations derived from (11.10) after some algebra:

$$A^2 = \frac{\overline{x^2} + \overline{y^2}}{2} + \sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}, \quad (11.16)$$

$$B^2 = \frac{\overline{x^2} + \overline{y^2}}{2} - \sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}. \quad (11.17)$$

Note that A and B are exactly halves the a and b parameters computed by the COSMOS image analyser (Stobie 1980,1986). Actually, a and b are defined by Stobie as the semi-major and semi-minor axes of an elliptical shape with constant surface brightness, which would have the same 2nd-order moments as the analysed object.

11.1.6 Ellipse parameters: CXX, CYY, CXY

A, B and THETA are not very convenient to use when, for instance, one wants to know if a particular SExtractor detection extends over some position. For this kind of application, three other ellipse parameters are provided; CXX, CYY and CXY. They do nothing more than describing the same ellipse, but in a different way: the elliptical shape associated to a detection is now parameterized as

$$\text{CXX}(x - \bar{x})^2 + \text{CYY}(y - \bar{y})^2 + \text{CXY}(x - \bar{x})(y - \bar{y}) = R^2, \quad (11.18)$$

where R is a parameter which scales the ellipse, in units of A (or B). Generally, the isophotal limit of a detected object is well represented by $R \approx 3$ (Fig. 11.1). Ellipse parameters can be derived from the 2nd order moments:

$$\text{CXX} = \frac{\cos^2 \text{THETA}}{A^2} + \frac{\sin^2 \text{THETA}}{B^2} = \frac{\overline{y^2}}{\sqrt{\left(\frac{x^2 - y^2}{2}\right)^2 + \overline{xy}^2}} \quad (11.19)$$

$$\text{CYY} = \frac{\sin^2 \text{THETA}}{A^2} + \frac{\cos^2 \text{THETA}}{B^2} = \frac{\overline{x^2}}{\sqrt{\left(\frac{x^2 - y^2}{2}\right)^2 + \overline{xy}^2}} \quad (11.20)$$

$$\text{CXY} = 2 \cos \text{THETA} \sin \text{THETA} \left(\frac{1}{A^2} - \frac{1}{B^2} \right) = -2 \frac{\overline{xy}}{\sqrt{\left(\frac{x^2 - y^2}{2}\right)^2 + \overline{xy}^2}} \quad (11.21)$$

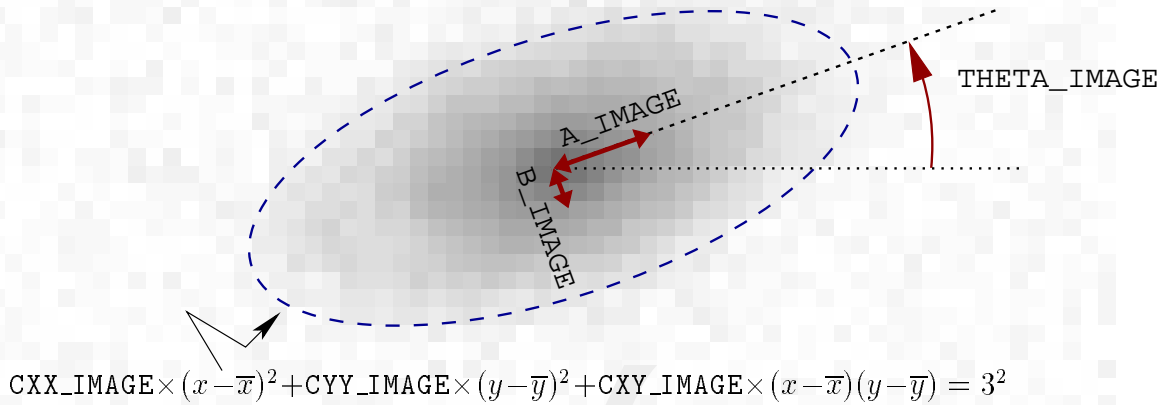


Figure 11.1: The meaning of basic shape parameters.

11.1.7 By-products of shape parameters: ELONGATION and ELLIPTICITY

1

¹Such parameters are dimensionless and therefore do not accept any _IMAGE or _WORLD suffix

These parameters are directly derived from A and B:

$$\text{ELONGATION} = \frac{A}{B} \quad \text{and} \quad (11.22)$$

$$\text{ELLIPTICITY} = 1 - \frac{B}{A}. \quad (11.23)$$

11.1.8 Position errors: ERRX2, ERRY2, ERRXY, ERRA, ERRB, ERRTHETA, ERRCXX, ERRCYY, ERRCXY

Uncertainties on the position of the barycenter can be estimated using photon statistics. Of course, this kind of estimate has to be considered as a lower-value of the real error since it does not include, for instance, the contribution of detection biases or the contamination by neighbours. As SExtractor does not currently take into account possible correlations between pixels, the variances simply write:

$$\text{ERRX2} = \text{var}(\bar{x}) = \frac{\sum_{i \in \mathcal{S}} \sigma_i^2 (x_i - \bar{x})^2}{\left(\sum_{i \in \mathcal{S}} I_i \right)^2}, \quad (11.24)$$

$$\text{ERRY2} = \text{var}(\bar{y}) = \frac{\sum_{i \in \mathcal{S}} \sigma_i^2 (y_i - \bar{y})^2}{\left(\sum_{i \in \mathcal{S}} I_i \right)^2}, \quad (11.25)$$

$$\text{ERRXY} = \text{cov}(\bar{x}, \bar{y}) = \frac{\sum_{i \in \mathcal{S}} \sigma_i^2 (x_i - \bar{x})(y_i - \bar{y})}{\left(\sum_{i \in \mathcal{S}} I_i \right)^2}. \quad (11.26)$$

σ_i is the flux uncertainty estimated for pixel i :

$$\sigma_i^2 = \sigma_{B_i}^2 + \frac{I_i}{g_i}, \quad (11.27)$$

where σ_{B_i} is the local background noise and g_i the local gain — conversion factor — for pixel i (see §7 for more details). Semi-major axis ERRA, semi-minor axis ERRB, and position angle ERRTHETA of the 1σ position error ellipse are computed from the covariance matrix exactly like in 11.1.5 for shape parameters:

$$\text{ERRA}^2 = \frac{\text{var}(\bar{x}) + \text{var}(\bar{y})}{2} + \sqrt{\left(\frac{\text{var}(\bar{x}) - \text{var}(\bar{y})}{2} \right)^2 + \text{cov}^2(\bar{x}, \bar{y})}, \quad (11.28)$$

$$\text{ERRB}^2 = \frac{\text{var}(\bar{x}) + \text{var}(\bar{y})}{2} - \sqrt{\left(\frac{\text{var}(\bar{x}) - \text{var}(\bar{y})}{2} \right)^2 + \text{cov}^2(\bar{x}, \bar{y})}, \quad (11.29)$$

$$\tan(2\text{ERRTHETA}) = 2 \frac{\text{cov}(\bar{x}, \bar{y})}{\text{var}(\bar{x}) - \text{var}(\bar{y})}. \quad (11.30)$$

And the ellipse parameters are:

$$\text{ERRCXX} = \frac{\cos^2 \text{ERRTHETA}}{\text{ERRA}^2} + \frac{\sin^2 \text{ERRTHETA}}{\text{ERRB}^2} = \frac{\text{var}(\bar{y})}{\sqrt{\left(\frac{\text{var}(\bar{x}) - \text{var}(\bar{y})}{2} \right)^2 + \text{cov}^2(\bar{x}, \bar{y})}}, \quad (11.31)$$

$$\text{ERRCY} = \frac{\sin^2 \text{ERRTHETA}}{\text{ERRA}^2} + \frac{\cos^2 \text{ERRTHETA}}{\text{ERRB}^2} = \frac{\text{var}(\bar{x})}{\sqrt{\left(\frac{\text{var}(\bar{x}) - \text{var}(\bar{y})}{2}\right)^2 + \text{cov}^2(\bar{x}, \bar{y})}}, \quad (11.32)$$

$$\text{ERRCY} = 2 \cos \text{ERRTHETA} \sin \text{ERRTHETA} \left(\frac{1}{\text{ERRA}^2} - \frac{1}{\text{ERRB}^2} \right) \quad (11.33)$$

$$= -2 \frac{\text{cov}(\bar{x}, \bar{y})}{\sqrt{\left(\frac{\text{var}(\bar{x}) - \text{var}(\bar{y})}{2}\right)^2 + \text{cov}^2(\bar{x}, \bar{y})}}. \quad (11.34)$$

11.1.9 Handling of “infinitely thin” detections

Apart from the mathematical singularities that can be found in some of the above equations describing shape parameters (and which SExtractor handles, of course), some detections with very specific shapes may yield quite unphysical parameters, namely null values for B, ERRB, or even A and ERRA. Such detections include single-pixel objects and horizontal, vertical or diagonal lines which are 1-pixel wide. They will generally originate from glitches; but very undersampled and/or low S/N genuine sources may also produce such shapes.

For basic shape parameters, the following convention was adopted: if the light distribution of the object falls on one single pixel, or lies on a sufficiently thin line of pixels, which we translate mathematically by

$$\overline{x^2 y^2} - \overline{xy}^2 < \rho^2, \quad (11.35)$$

then $\overline{x^2}$ and $\overline{y^2}$ are incremented by ρ . SExtractor sets $\rho = 1/12$, which is the variance of a 1-dimensional top-hat distribution with unit width. Therefore $1/\sqrt{12}$ represents the typical minor-axis values assigned (in pixels units) to undersampled sources in SExtractor.

Positional errors are more difficult to handle, as objects with very high signal-to-noise can yield extremely small position uncertainties, just like singular profiles do. Therefore SExtractor first checks that (11.35) is true. If this is the case, a new test is conducted:

$$\text{var}(\bar{x}) \text{var}(\bar{y}) - \text{covar}^2(\bar{x}, \bar{y}) < \rho_e^2, \quad (11.36)$$

where ρ_e is arbitrarily set to $(\sum_{i \in S} \sigma_i^2) / (\sum_{i \in S} I_i)^2$. If (11.36) is true, then $\overline{x^2}$ and $\overline{y^2}$ are incremented by ρ_e .

11.2 Windowed positional parameters

Parameters measured within an object’s isophotal limit are sensitive to two main factors: 1) changes in the detection threshold, which create a variable bias and 2) irregularities in the object’s isophotal boundaries, which act as additional “noise” in the measurements.

Measurements performed through a *window* function (an *envelope*) do not have such drawbacks. SExtractor versions 2.4 and above implement “windowed” versions for most of the measurements described in 11.1:

| Isophotal parameters | Equivalent windowed parameters |
|--|---|
| X_IMAGE, Y_IMAGE | XWIN_IMAGE, YWIN_IMAGE |
| ERRA_IMAGE, ERRB_IMAGE, ERRTHETA_IMAGE | ERRAWIN_IMAGE, ERRBWIN_IMAGE, ERRTHETAWIN_IMAGE |
| A_IMAGE, B_IMAGE, THETA_IMAGE | AWIN_IMAGE, BWIN_IMAGE, THETAWIN_IMAGE |
| X2_IMAGE, Y2_IMAGE, XY_IMAGE | X2WIN_IMAGE, Y2WIN_IMAGE, XYWIN_IMAGE |
| CXX_IMAGE, CYY_IMAGE, CXY_IMAGE | CXXWIN_IMAGE, CYYWIN_IMAGE, CXYWIN_IMAGE |

The computations involved are roughly the same except that the pixel values are integrated within a circular Gaussian window as opposed to the object's isophotal footprint. The Gaussian window is scaled to each object; its FWHM is the diameter of the disk that contains half of the object flux (d_{50}). Note that in double-image mode (§3) the window is scaled based on the *measurement* image.

11.2.1 Windowed centroid: XWIN, YWIN

This is an iterative process. The computation starts by initializing the windowed centroid coordinates $\overline{x_{\text{WIN}}}^{(0)}$ and $\overline{y_{\text{WIN}}}^{(0)}$ to their basic \bar{x} and \bar{y} isophotal equivalents, respectively. Then at each iteration t , $\overline{x_{\text{WIN}}}$ and $\overline{y_{\text{WIN}}}$ are refined using:

$$\text{XWIN}^{(t+1)} = \overline{x_{\text{WIN}}}^{(t+1)} = \overline{x_{\text{WIN}}}^{(t)} + 2 \frac{\sum_{r_i^{(t)} < r_{\text{max}}} w_i^{(t)} I_i (x_i - \overline{x_{\text{WIN}}}^{(t)})}{\sum_{r_i^{(t)} < r_{\text{max}}} w_i^{(t)} I_i}, \quad (11.37)$$

$$\text{YWIN}^{(t+1)} = \overline{y_{\text{WIN}}}^{(t+1)} = \overline{y_{\text{WIN}}}^{(t)} + 2 \frac{\sum_{r_i^{(t)} < r_{\text{max}}} w_i^{(t)} I_i (y_i - \overline{y_{\text{WIN}}}^{(t)})}{\sum_{r_i^{(t)} < r_{\text{max}}} w_i^{(t)} I_i}, \quad (11.38)$$

where

$$w_i^{(t)} = \exp\left(-\frac{r_i^{(t)2}}{2s_{\text{WIN}}^2}\right), \quad (11.39)$$

with

$$r_i^{(t)} = \sqrt{(x_i - \overline{x_{\text{WIN}}}^{(t)})^2 + (y_i - \overline{y_{\text{WIN}}}^{(t)})^2} \quad (11.40)$$

and $s_{\text{WIN}} = d_{50}/\sqrt{8 \ln 2}$. The process stops when the change in position between two iterations is less than 2×10^{-4} pixel, a condition which is generally achieved in about 3 to 5 iterations.

Although the iterative nature of the processing slows down the processing, it is recommended to use whenever possible windowed parameters instead of their isophotal equivalents, since the measurements they provide are much more precise (Fig. 11.2). The precision in centroiding offered by XWIN_IMAGE and YWIN_IMAGE is actually very close to that of PSF-fitting on focused and properly sampled star images, and can also be applied to galaxies. It has been verified that for isolated, Gaussian-like PSFs, its accuracy is close to the theoretical limit set by image noise².

11.2.2 Windowed 2nd order moments: X2, Y2, XY

Windowed second-order moments are computed on the image data once the centering process from §11.2.1 has converged:

$$\text{X2WIN} = \overline{x_{\text{WIN}}^2} = \frac{\sum_{r_i < r_{\text{max}}} w_i I_i (x_i - \overline{x_{\text{WIN}}})^2}{\sum_{r_i < r_{\text{max}}} w_i I_i}, \quad (11.41)$$

$$\text{Y2WIN} = \overline{y_{\text{WIN}}^2} = \frac{\sum_{r_i < r_{\text{max}}} w_i I_i (y_i - \overline{y_{\text{WIN}}})^2}{\sum_{r_i < r_{\text{max}}} w_i I_i}, \quad (11.42)$$

$$\text{XYWIN} = \overline{xy_{\text{WIN}}} = \frac{\sum_{r_i < r_{\text{max}}} w_i I_i (x_i - \overline{x_{\text{WIN}}})(y_i - \overline{y_{\text{WIN}}})}{\sum_{r_i < r_{\text{max}}} w_i I_i}. \quad (11.43)$$

Windowed second-order moments are typically twice smaller than their isophotal equivalent.

²see <http://www.astromatic.net/forum/showthread.php?tid=581>

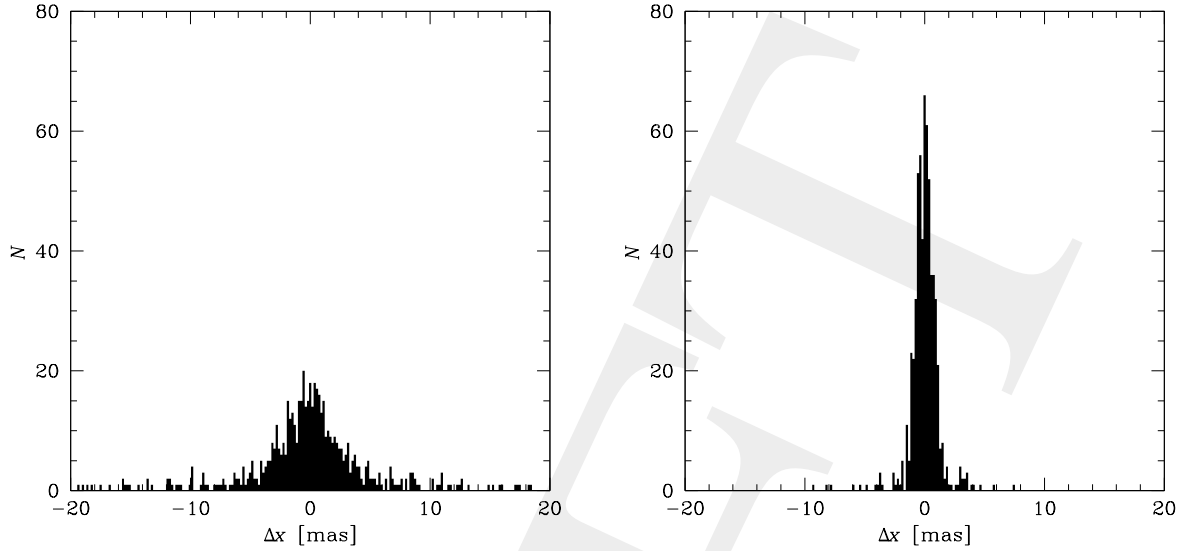


Figure 11.2: Comparison between isophotal and windowed centroid measurement accuracies on simulated, background noise-limited images. *Left*: histogram of the difference between X_IMAGE and the simulation centroid in x. *Right*: histogram of the difference between XWIN_IMAGE and the simulation centroid in x.

11.2.3 Windowed ellipse parameters: CXXWIN, CYYWIN, CXYWIN

They are computed from the windowed 2nd order moments exactly the same way as in §11.1.6.

11.2.4 Windowed position errors: ERRX2WIN, ERRY2WIN, ERRXYWIN, ERRRAWIN, ERRBWIN, ERRTHETAWIN, ERRCXXWIN, ERRCYYWIN, ERRCXYWIN

Windowed position errors are computed on the image data once the centering process from §11.2.1 has converged. Assuming that noise is uncorrelated among pixels, standard error propagation applied to (11.37) and (11.37) gives us:

$$\text{ERRX2WIN} = \text{var}(\overline{x_{\text{WIN}}}) = 4 \frac{\sum_{r_i < r_{\text{max}}} w_i^2 \sigma_i^2 (x_i - \bar{x})^2}{(\sum_{r_i < r_{\text{max}}} w_i I_i)^2}, \quad (11.44)$$

$$\text{ERRY2WIN} = \text{var}(\overline{y_{\text{WIN}}}) = 4 \frac{\sum_{r_i < r_{\text{max}}} w_i^2 \sigma_i^2 (y_i - \bar{y})^2}{(\sum_{r_i < r_{\text{max}}} w_i I_i)^2}, \quad (11.45)$$

$$\text{ERRXYWIN} = \text{cov}(\overline{x_{\text{WIN}}}, \overline{y_{\text{WIN}}}) = 4 \frac{\sum_{r_i < r_{\text{max}}} w_i^2 \sigma_i^2 (x_i - \overline{x_{\text{WIN}}})(y_i - \overline{y_{\text{WIN}}})}{(\sum_{r_i < r_{\text{max}}} w_i I_i)^2}. \quad (11.46)$$

The semi-major axis ERRRAWIN, semi-minor axis ERRBWIN, and position angle ERRTHETAWIN of the 1σ position error ellipse are computed from the covariance matrix elements $\text{var}(\overline{x_{\text{WIN}}})$, $\text{var}(\overline{y_{\text{WIN}}})$, $\text{cov}(\overline{x_{\text{WIN}}}, \overline{y_{\text{WIN}}})$, exactly as in §11.1.8: see eqs. (11.28), (11.29), (11.30), (11.31), (11.32) and (11.33).

11.3 Astrometry and WORLD coordinates

All SExtractor measurements related to positions, distances and areas in the image, like those described above can also be expressed in WORLD coordinates in the output catalogue. These parameters simply have the `_WORLD` suffix instead of the `_IMAGE` appended to them. The conversion from `IMAGE` to `WORLD` coordinates is presently performed by using information found in the FITS header of the *measurement* image, even if the parameter is originally computed from the *detection* image (like the basic shape parameters for instance).

To understand how this is done in practice, let's have a general look at the way the mapping from `IMAGE` to `WORLD` coordinates is currently described in a FITS image header. First, a linear transformation (involving most of the time only scaling and possibly rotation, and more rarely shear) allows one to convert integer pixel positions (1,2,...) for each axis to some "projected" coordinate system. This is where you might want to stop if your `WORLD` system is just some kind of simple focal-plane coordinate-system (in meters for instance), or for a calibrated wavelength axis (spectrum). Now, the FITS WCS (World Coordinate System) convention allows you to apply to these "projected coordinates" a non-linear transformation, which is in fact a de-projection back to "local" spherical (celestial) coordinates. Many types of projections are allowed by the WCS convention, but the traditional tangential (gnomonic) projection is the most commonly used. The last step of the transformation is to convert these local coordinates, still relative to a projection reference point, to an absolute position in celestial longitude and latitude, for instance right-ascension and declination. For this one needs to know the reference frame of the coordinate system, which often requires some information about the equinox or the observation date. At this level, all transformations are matters of spherical trigonometry.

11.3.1 Celestial coordinates

We will not describe here the transformations $(\alpha, \delta) = f(x, y)$ themselves. SExtractor de-projections rely on the WCSlib 2.4 written by Mark Calabretta, and all the details concerning those can be found in Greisen & Calabretta (1995). In addition to the `_WORLD` parameters, 3 purely angular "world" coordinates are available in SExtractor, expressed in decimal degrees:

1. `_SKY` coordinates: strictly identical to `_WORLD` coordinates, except that the units are explicitly degrees. They correspond to sky coordinates in the "native" system without any precession correction, conversion, etc.
2. `_J2000` coordinates: precession corrections are applied in the FK5 system to convert to J2000 coordinates if necessary.
3. `_B1950` coordinates: precession corrections are computed in the FK5 system and transformation to B1950 is applied.

Transformation to J2000 or B1950 is done without taking into account proper motions, which are obviously unknown for the detected objects. In both cases, epoch 2000.0 is assumed.

Here is a list of catalogue parameters currently supporting angular coordinates:

| Image parameters | World parameters | Angular parameters |
|--|--|--|
| X_IMAGE, Y_IMAGE | X_WORLD, Y_WORLD | ALPHA_SKY, DELTA_SKY ALPHA_J2000, DELTA_J2000 ALPHA_B1950, DELTA_B1950 |
| XWIN_IMAGE, YWIN_IMAGE | XWIN_WORLD, YWIN_WORLD | ALPHAWIN_SKY, DELTAWIN_SKY ALPHAWIN_J2000, DELTAWIN_J2000 ALPHAWIN_B1950, DELTAWIN_B1950 |
| XPEAK_IMAGE, YPEAK_IMAGE | XPEAK_WORLD, YPEAK_WORLD | ALPHAPEAK_SKY, DELTAPEAK_SKY ALPHAPEAK_J2000, DELTAPEAK_J2000 ALPHAPEAK_B1950, DELTAPEAK_B1950 |
| X2_IMAGE, Y2_IMAGE, XY_IMAGE X2WIN_IMAGE, Y2WIN_IMAGE, XYWIN_IMAGE CXX_IMAGE, CYY_IMAGE, CXY_IMAGE CXXWIN_IMAGE, CYYWIN_IMAGE, CXYWIN_IMAGE | X2_WORLD, Y2_WORLD, XY_WORLD X2WIN_WORLD, Y2WIN_WORLD, XYWIN_WORLD CXX_WORLD, CYY_WORLD, CXY_WORLD CXXWIN_WORLD, CYYWIN_WORLD, CXYWIN_WORLD | |

TO BE WRITTEN

11.3.2 Use of the FITS keywords for astrometry

TO BE WRITTEN

11.4 Photometry

SEXTRACTOR has currently the possibility to compute four types of magnitude: *isophotal*, *corrected-isophotal*, *fixed-aperture* and *adaptive-aperture*. For all magnitudes, an additive “zero-point” correction can be applied with the `MAG_ZEROPOINT` keyword. Note that for each `MAG_XXXX`, a magnitude error estimate `MAGERR_XXXX`, a linear `FLUX_XXXX` measurement and its error estimate `FLUXERR_XXXX` are also available.

Isophotal magnitudes (`MAG_ISO`) are computed simply, using the detection threshold as the lowest isophote.

Corrected isophotal magnitudes (`MAG_ISOCOR`) can be considered as a quick-and-dirty way for retrieving the fraction of flux lost by isophotal magnitudes. Although their use is now deprecated, they have been kept in SEXTRACTOR 2.x and above for compatibility with SEXTRACTOR 1. If we make the assumption that the intensity profiles of the faint objects recorded on the plate are roughly Gaussian because of atmospheric blurring, then the fraction $\eta = \frac{I_{\text{iso}}}{I_{\text{tot}}}$ of the total flux enclosed within a particular isophote reads (see Maddox et al. 1990):

$$\left(1 - \frac{1}{\eta}\right) \ln(1 - \eta) = \frac{At}{I_{\text{iso}}} \quad (11.47)$$

where A is the area and t the threshold related to this isophote. Eq. (11.47) is not analytically invertible, but a good approximation to η (error $< 10^{-2}$ for $\eta > 0.4$) can be done with the second-order polynomial fit:

$$\eta \approx 1 - 0.1961 \frac{At}{I_{\text{iso}}} - 0.7512 \left(\frac{At}{I_{\text{iso}}}\right)^2 \quad (11.48)$$

A “total” magnitude m_{tot} estimate is then

$$m_{\text{tot}} = m_{\text{iso}} + 2.5 \log \eta \quad (11.49)$$

Clearly this cheap correction works best with stars; and although it is shown to give tolerably accurate results with most disk galaxies, it fails with ellipticals because of the broader wings of their profiles.

Fixed-aperture magnitudes (`MAG_APER`) estimate the flux above the background within a circular aperture. The diameter of the aperture in pixels (`PHOTOM_APERTURES`) is supplied by the user (in fact it does not need to be an integer since each “normal” pixel is subdivided in 5×5 sub-pixels before measuring the flux within the aperture). If `MAG_APER` is provided as a vector `MAG_APER[n]`, at least n apertures must be specified with `PHOTOM_APERTURES`.

Automatic aperture magnitudes (`MAG_AUTO`) provides an estimate of the “total magnitude” by integrating the source flux within an adaptively scaled aperture. `SEXTRACTOR`’s automatic aperture photometry routine is inspired by Kron’s “first moment” algorithm (1980). (1) We define an elliptical aperture whose elongation ϵ and position angle θ are defined by second order moments of the object’s light distribution. The ellipse is scaled to $R_{\max} \cdot \sigma_{\text{iso}}$ ($6\sigma_{\text{iso}}$, which corresponds roughly to 2 isophotal “radii”). (2) Within this aperture we compute the “first moment”:

$$r_1 = \frac{\sum r I(r)}{\sum I(r)} \quad (11.50)$$

Kron (1980) and Infante (1987) have shown that for stars and galaxy profiles convolved with Gaussian seeing, $\geq 90\%$ of the flux is expected to lie within a circular aperture of radius kr_1 if $k = 2$, almost independently of their magnitude. This picture remains unchanged if we consider an ellipse with ϵkr_1 and kr_1/ϵ as principal axes. $k = 2$ defines a sort of balance between systematic and random errors. By choosing a larger $k = 2.5$, the mean fraction of flux lost drops from about 10% to 6%. When Signal to Noise is low, it may appear that an erroneously small aperture is taken by the algorithm. That’s why we have to bound the smallest accessible aperture to R_{\min} (typically $R_{\min} = 3 - 4 \sigma_{\text{iso}}$). The user has full control over the parameters k and R_{\min} through the configuration parameters `PHOT_AUTOPARAMS`; by default, `PHOT_AUTOPARAMS` is set to 2.5,3.5.

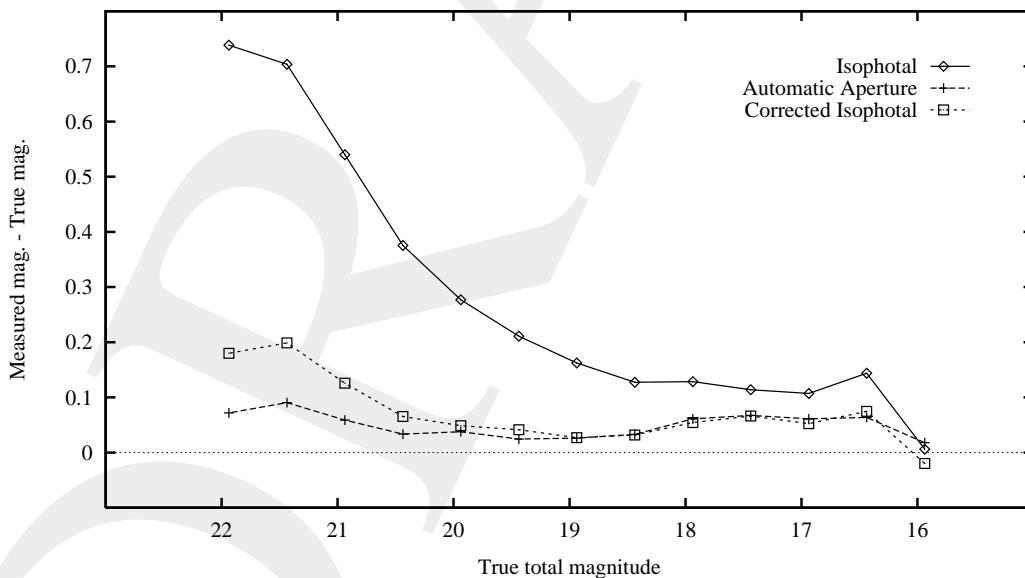


Figure 11.3: Flux lost (expressed as a mean magnitude difference) with different faint-object photometry techniques as a function of total magnitude (see text). Only isolated galaxies (no blends) of the simulations have been considered.

Aperture magnitudes are sensitive to crowding. In `SEXTRACTOR 1`, `MAG_AUTO` measurements were not very robust in that respect. It was therefore suggested to replace the aperture magnitude by the corrected-isophotal one when an object is too close to its neighbours (2 isopho-

tal radii for instance). This was done automatically when using the `MAG_BEST` magnitude: `MAG_BEST = MAG_AUTO` when it is sure that no neighbour can bias `MAG_AUTO` by more than 10%, or `MAG_BEST = MAG_ISOCOR` otherwise. Experience showed that the `MAG_ISOCOR` and `MAG_AUTO` magnitude would loose about the same fraction of flux on stars or compact galaxy profiles: around 0.06 % for default extraction parameters. The use of `MAG_BEST` is now deprecated as `MAG_AUTO` measurements are much more robust in versions 2.x of `SEXTRACTOR`. The first improvement is a crude subtraction of all the neighbours which have been detected around the measured source (the `MASK_TYPE BLANK` option). The second improvement is an automatic correction of parts of the aperture that are suspected to be contaminated by a neighbour. This is done by mirroring the opposite, cleaner side of the measurement ellipse if available (the `MASK_TYPE CORRECT` option, which is also the default). Figure 11.3 shows the mean loss of flux measured with isophotal (threshold = 24.4 magnitude arcsec⁻²), corrected isophotal and automatic aperture photometries for simulated galaxy B_J on a typical Schmidt-survey plate image. The automatic adaptive aperture photometry leads to the lowest loss of flux.

Photographic photometry In `DETECT_TYPE PHOTO` mode, `SEXTRACTOR` assumes that the response of the detector, over the dynamic range of the image, is logarithmic. This is generally a good approximation for photographic density on deep exposures. Photometric procedures described above remain unchanged, except that for each pixel we apply first the transformation

$$I = I_0 10^{D/\gamma} , \quad (11.51)$$

where γ (`MAG_GAMMA`) is the contrast index of the emulsion, D the original pixel value from the background-subtracted image, and I_0 is computed from the magnitude zero-point m_0 :

$$I_0 = \frac{\gamma}{\ln 10} 10^{-0.4 m_0} . \quad (11.52)$$

One advantage of using a density-to-intensity transformation relative to the local sky background is that it corrects (to some extent) large-scale inhomogeneities in sensitivity (see Bertin 1996 for details).

Errors on magnitude An estimate of the error³ is available for each type of magnitude. It is computed through

$$\Delta m = 1.0857 \frac{\sqrt{A \sigma^2 + F/g}}{F} \quad (11.53)$$

where A is the area (in pixels) over which the total flux F (in ADU) is summed, σ the standard deviation of noise (in ADU) estimated from the background, and g the detector gain (`GAIN` parameter⁴, in e^-/ADU). For corrected-isophotal magnitudes, a term, derived from Eq. 11.48 is quadratically added to take into account the error on the correction itself.

In `DETECT_TYPE PHOTO` mode, things are slightly more complex. Making the assumption that plate-noise is the major contributor to photometric errors, and that it is roughly constant in density, we can write:

$$\Delta m = 1.0857 \ln 10 \frac{\sigma}{\gamma} \frac{\sqrt{\sum_{x,y} I^2(x,y)}}{\sum_{x,y} I(x,y)} = 2.5 \frac{\sigma}{\gamma} \frac{\sqrt{\sum_{x,y} I^2(x,y)}}{\sum_{x,y} I(x,y)} \quad (11.54)$$

where $I(x,y)$ is the contribution of pixel (x,y) to the total flux (Eq. 11.51). The `GAIN` is ignored in `PHOTO` mode.

³It is important to note that this error provides a lower limit, since it does not take into account the (complex) uncertainty on the local background estimate.

⁴Setting `GAIN` to 0 in the configuration file is equivalent to $g = +\infty$

Background is the last point relative to photometry. The assumption made in §6.1 — that the “local” background associated to an object can be interpolated from the global background map — is no longer valid in crowded regions. An example is a globular cluster superimposed on a bulge of galaxy. `SEXTRACTOR` offers the possibility to estimate locally the background used to compute magnitudes. When this option is switched on (`BACKPHOTO_TYPE LOCAL` instead of `GLOBAL`), the “photometric” background is estimated within a “rectangular annulus” around the isophotal limits of the object. The thickness of the annulus (in pixels) can be specified by the user with `BACKPHOTO_SIZE`. A typical value is `BACKPHOTO_SIZE=24`.

Chapter 12

Model fitting

Since version 2.8, SEXTRACTOR can fit models to the objects in the images. The fit is performed with Levenberg-Marquardt minimization, inside a disk which diameter is scaled to include the isophotal footprint plus a 20 % margin, plus the size of the PSF model image.

The models that can be fit are:

- Exponential disk (eq. [12.1])

$$\Sigma_{\text{ExpDisk}}(R) = \Sigma(0) \exp\left(-\frac{R}{h}\right) \quad (12.1)$$

- Sérsic ($R^{1/n}$) spheroid (bulge, eq. [12.2])

$$\Sigma_{\text{Sersic}}(R) = \Sigma(0) \exp\left[-b(n) \left(\frac{R}{R_e}\right)^{1/n}\right], \quad (12.2)$$

where, for the Sérsic (1968) model, $b(n)$ is the solution of

$$2\gamma[2n, b(n)] = \Gamma(2n) \quad (12.3)$$

An accurate approximation for the solution for $b(n)$ of equation (12.3) is (Ciotti & Bertin, 1999)

$$b(n) = 2n - \frac{1}{3} + \frac{4}{405n} + \frac{46}{25515n^2} + \frac{131}{1148175n^3}$$

- de Vaucouleurs (1948) spheroid (bulge, eq. [12.2], with $n = 4$)
- Exponential disk + Sérsic ($R^{1/n}$) spheroid (bulge)
- Point source
- Background (constant)

For these models, SEXTRACTOR can compute fluxes and magnitudes, as well as sizes (disk scale length for the disks and effective — projected half-light — radii for the spheroids), characteristic surface magnitudes, and Sérsic index, as well as their uncertainties.

The models are concentric (they assume the same center) and are all convolved with the PSF, given by the .psf file, which must be determined by first running PSFEX (see below).

Unfortunately, the Sérsic profile is very cuspy in the center for $n > 2$. To avoid huge wings in the FFTs when convolving the profile with the PSF, the profile is split between a 3rd order polynomial, analytically fit to match, in intensity and its 1st and 2nd spatial derivatives, the Sérsic profile at $R = 4$ pixels, $I(r) = I_0 + (r/a)^3$, which has zero first and 2nd derivative at the center, i.e. a homogeneous core on one hand, and a residual with finite extent on the other.

For the fit of the spheroid component, the apparent ellipticity allowed is taken in the range $[0.5, 2]$. This obviously forbids very flat spheroids to avoid confusion with a flattened disk. By allowing ellipticities greater than unity, SExtractor avoids dichotomies of position angle when the ellipticity is very low. The Sérsic index is allowed values between 1 and 10.

Models are measured according to the following table. Table 12.1 should be interpreted as mean-

Table 12.1: Which parameters trigger which models in fits?

| | | |
|--|---|---|
| FLUX_BACKOFFSET or FLUXERR_BACKOFFSET | → | background |
| DISK_XXX | → | exponential disk |
| SPHEROID_SERSICN or SPHEROID_SERSICNERR | → | Sérsic |
| SPHEROID_XXX without SPHEROID_SERSICN[ERR] | → | de Vaucouleurs ($n = 4$ Sérsic) |
| MODEL_XXX only | → | Sérsic [??? |
| SPHEROID_XXX and DISK_XXX | → | Sérsic spheroid + exponential disk [??? |

ing that if one of the parameters given in the parameter file (e.g. `default.param`) includes the string on the left of the arrow, the model to the right of the arrow is triggered. For example, when including parameters that contain the string ‘MODEL’, both galaxies and stars are fit with convolutions of Sérsic models with the PSF. If no SPHEROID_XXX or DISK_XXX parameter is present, but the model-fitting process is nevertheless triggered by the presence of other measurement parameters or relevant CHECKIMAGE_TYPES, a single component with Sérsic profile and adjustable Sérsic index n is fitted.

The number of parameters that are fit are 2 for the global center, 4 per model for the scale, normalization, aspect ratio and position angle, plus the index for the Sérsic model. For example, fitting a Sérsic + exponential disk involves a fitting 11 parameters.

Experience shows that the de Vaucouleurs spheroid + exponential disk combination provides fairly accurate and robust fits for moderately resolved faint galaxies. An adjustable Sérsic index may offer lower residuals on spheroids and/or well-resolved galaxies, but makes the fit less robust and more sensitive to PSF model errors. One might think of adding some mechanism to lock or unlock the Sérsic index automatically in future versions of SExtractor.

The measurement parameters related to model-fitting follow the usual SExtractor rules:

Flux measurements are available in ADUs (FLUX_XXX parameters) or magnitudes (MAG_XXX parameters), Coordinates and radii are available in pixels or celestial units (provided that the FITS image header contains the appropriate WCS information).

xxxMODEL_yyy measurement parameters deal with the global fitted model, i.e. the sum of all components (e.g. chi-square per d.o.f. CHI2_MODEL, PSF-corrected ellipticities E1/2MODEL_IMAGE, EPS1/2MODEL_IMAGE).

1σ error estimates xxxERR_yyy are provided for most measurement parameters; they are obtained by marginalizing the full covariance matrix of the fit.

Since the model fitting involves convolution with the PSF, it is imperative to launch PSFEX before launching SExtractor. In practice, the sequence of operations is:

1. Run SExtractor to prepare PSFEX;
2. Run PSFEX to prepare model fits in SExtractor;
3. Run SExtractor with model fit parameters.

DRAFT

Chapter 13

Checking the output

A visual inspection is always useful after running SExtractor on the first image to check if all the objects seen on the image are actually detected, and also to check that there are no spurious detections.

For this, the easiest method is to display the image with ds9 and to superimpose the positions of the objects detected by using...

Chapter 14

Cross-identification within SEXTRACTOR

SEXTRACTOR allows one to perform on-the-fly cross-identification of detections with an ASCII list provided by the user. Cross-identification can be done both in pixel or world coordinates. Configuration parameters related to cross-identification are prefixed with ASSOC_.

14.1 The ASSOC list

The ASSOC process is initiated by requesting in the parameter file at least one of the ASSOC catalogue parameters: VECTOR_ASSOC or NUMBER_ASSOC. Then SEXTRACTOR looks for an ASCII file (the so-called ASSOC list), which must be specified using the ASSOC_NAME configuration keyword. The ASSOC list should contain columns of numbers separated by spaces or tabs. Each line describes a source that will enter the cross-identification process. Lines with zero characters, or beginning with “#” (for comments) are ignored. You may therefore use any ASCII catalogue generated by a previous SEXTRACTOR run as an ASSOC list.

In order to perform the cross-identification, SEXTRACTOR needs to know which are the columns that contain the coordinates in the ASSOC list. These can be designated using the ASSOC_PARAMS configuration parameter. The syntax is: “ASSOC_PARAMS $c_x, c_y [, c_z]$ ”, where c_x and c_y are the positions of the columns containing the x and y coordinates, or world coordinates such as α and δ . By definition, the position of the first column is 1. c_z (optional) specifies an extra column containing some “Z” parameter that may be used for controlling or weighting the ASSOC process. Z will typically be a flux estimate. c_z is required if ASSOC_TYPE is MIN, MAX, MEAN or MAG_MEAN (see §14.2 below).

14.2 Controlling the ASSOC process

Three configuration parameters control the ASSOC process. The first one, ASSOCCOORD_TYPE, should be set to PIXEL or WORLD, depending on what type of coordinates is in the ASSOC list. If ASSOCCOORD_TYPE is set to PIXEL, SEXTRACTOR expects the x and y coordinates to comply with the FITS convention: by definition, the center of the first pixel in the image array has pixel-coordinates (1.0,1.0). If ASSOCCOORD_TYPE is set to WORLD, SEXTRACTOR uses the WCS information found in the FITS image header to convert to pixel coordinates the coordinates read in the ASSOC list prior to making the cross-identification.

`ASSOC_RADIUS`, accepts a decimal number which represents the maximum distance (in pixels) one should have between the barycenter of the current `SExtractor` detection and an `ASSOC`-list member to consider a match. This number must of course account for positional uncertainties in both catalogues. In most cases, a value of a few pixels will do just fine.

The last configuration parameter, `ASSOC_TYPE`, accepts a keyword as argument and selects the kind of identification procedure one wants to operate:

- **FIRST**: this is the simplest way of performing a cross-identification. It does not require the c_Z column in `ASSOC_PARAMS`. The first geometrical match encountered while scanning the `ASSOC` list is retained as the actual match. This can be used for catalogues with low spatial density.
- **NEAREST**: this option does not require the c_Z column in `ASSOC_PARAMS`. The match is performed with the `ASSOC`-list member the closest (in position) to the current detection, provided that it lies within the `ASSOC_RADIUS`.
- **SUM**: all parameters issued from `ASSOC`-list members which geometrically match the current detection are summed. c_Z is not required.
- **MAG_SUM**: all parameters c_i issued from `ASSOC`-list members which geometrically match the current detection are combined using the following law: $-2.5 \log(\sum_i 10^{-0.4c_i})$. This option allows one to sum flux contributions from magnitude data. c_Z is not required.
- **MIN**: among all geometrical matches, retains the `ASSOC`-list member which has the smallest Z parameter.
- **MAX**: among all geometrical matches, retains the `ASSOC`-list member which has the largest Z parameter.
- **MEAN**: all parameters issued from `ASSOC`-list members which geometrically match the current detection are weighted-averaged, using the Z parameter as the weight.
- **MAG_MEAN**: all parameters issued from `ASSOC`-list members which geometrically match the current detection are weighted-averaged, using $10^{-0.4Z}$ as the weight. This option is useful for weighting catalogue sources with magnitudes.

14.3 Output from ASSOC

Now that we have described the cross-identification process, let's see how information coming from the matching with the `ASSOC` list are propagated to the output `SExtractor` catalogue.

The output of `ASSOC` data in `SExtractor` catalogue is done through the `VECTOR_ASSOC()` catalogue parameter. `VECTOR_ASSOC()` is a vector, each element of which refers to a column from the input `ASSOC` list. `VECTOR_ASSOC()` contains either `ASSOC`-list member data from the best match (if `ASSOC_TYPE` is `FIRST`, `NEAREST`, `MIN` or `MAX`), or a combination of `ASSOC`-list member data (if `ASSOC_TYPE` is `MEAN`, `MAG_MEAN`, `SUM` or `MAG_SUM`). If no match has been found, it just contains zeros. The `NUMBER_ASSOC` contains the number of `ASSOC`-list members that geometrically match the current `SExtractor` detection, and obviously, if different from zero, indicates that `VECTOR_ASSOC()` has a meaningful content.

The `ASSOC_DATA` configuration parameter is used to tell `SExtractor` to which column refers each element of `VECTOR_ASSOC()`. The syntax of `ASSOC_DATA` is similar to that of `ASSOC_PARAMS`:

“ASSOC_DATA c_1, c_2, c_3, \dots ” where the c_i are the column positions in the ASSOC list. The special case “ASSOC_DATA 0” tells SExtractor to propagate all columns from the ASSOC file to the output catalogue.

There are situations where it might be desirable to keep in the output SExtractor catalogue only those detections that were matched with some ASSOC-list member. Such a feature is controlled by the ASSOCSELEC_TYPE configuration parameter, which accepts one of the three following keywords:

- ALL: keep all SExtractor detections, regardless of matching. This is the default.
- MATCHED: keep only SExtractor detections that were matched with at least one ASSOC-list member.
- -MATCHED: keep only SExtractor detections that were not matched with any ASSOC-list member.

Acknowledgements

DRAFT

Bibliography

- Beard S. M., MacGillivray H. T., Thanisch P. F., The Cosmos System for Crowded-Field Analysis of Digitized Photographic Plate Scans, 1990, MNRAS, 247, 311
- Bijaoui A., Dantel M., Shut's Method, 1970, A&A, 6, 51
- Ciotti L., Bertin G., Analytical properties of the $R^{1/m}$ law, 1999, A&A, 352, 447
- de Vaucouleurs G., Recherches sur les Nebuleuses Extragalactiques, 1948, Annales d'Astrophysique, 11, 247
- Greisen E. W., Calabretta M. R., Representations of world coordinates in FITS, 2002, A&A, 395, 1061
- Pogson N., Magnitudes of Thirty-six of the Minor Planets for the first day of each month of the year 1857, 1856, MNRAS, 17, 12
- Sersic J. L., 1968, Atlas de galaxias australes
- Szalay A. S., Connolly A. J., Szokoly G. P., Simultaneous Multicolor Detection of Faint Galaxies in the Hubble Deep Field, 1999, AJ, 117, 68

Appendix A

FAQs (Frequently Asked Questions)

Q.: I heard that SExtractor does not do as good a job at extracting and measuring sources in image *YYY* as does package *XXX*.

A.: The purpose of SExtractor is to find a compromise between refinement in detection/measurement accuracy and computational speed. Although efforts are continuously being made in implementing more sophisticated algorithms as computer speed increases, SExtractor remains a general purpose program.

Q.: Can SExtractor work on X-ray data?

A.: X-ray data tends to be low count and source photon noise limited. In contrast with optical images, the low counts in X-ray images make the Poisson background noise highly skewed. Since many features in the current version of SExtractor assume that noise distribution is symmetrical around its mean or rely on χ^2 minimisation, using SExtractor to analyze X-ray data is not recommended at this time.

Q: Why isn't the detection threshold expressed in units of the background noise standard deviation in the *FILTERed* image ?

A: There are two reasons for this. First, it makes the threshold independent of the choice of a *FILTER*, which is a good thing. Second, having σ measured on the *FILTERed* image may have given un-informed users the wrong impression that increasing filtering systematically improves the detectability of any source, whereas it depends on scale.

Q: Can SExtractor compute asymmetry and concentration parameters?

Index

- adaptive-aperture, 43
- aperture magnitudes, 44
- apertures, 44
- APM, 24
- area, 25, 32, 33, 42, 43, 45
- asymmetry, 56
- atmospheric blurring, 43

- B1950, 42
- background map, 7, 8, 17–19, 46
- background RMS, 10
- bad pixel, 21, 28, 32
- bad pixels, 21, 28, 32
- barycenter, 21, 35, 38, 52
- boundaries, 17, 21, 39
- boundary, 32
- bulge, 46

- CCD, 8, 9, 20, 25, 27
- celestial coordinates, 12
- centroid, 25, 26, 40, 41
- check-image, 8, 20, 22–24, 28
- check-images, 20
- colour, 17
- concentration, 56
- configuration file, 3, 5, 6, 11, 13, 33, 45
- convolution, 19–22, 28
- corrected-isophotal, 43–45
- COSMOS, 24, 36
- covariance, 27, 36, 38, 41
- covariances, 27
- cross-identification, 2, 51, 52
- crowding, 18, 44

- DAOPHOT, 18
- deblending, 2, 8, 20, 24, 25, 32
- deep exposures, 45
- double-image mode, 21
- download, 3

- epoch, 42
- equinox, 42
- error ellipse, 38, 41

- external flags, 33
- Eye, 21, 22

- FITS binary-tables, 22
- FITS header, 12, 42
- flags, 9, 32, 33, 57
- flat-field, 28
- FOCAS, 18
- Fourier-transforms, 20
- FWHM, 10, 20, 26, 40

- gain, 11, 19, 27, 29, 38, 45
- gain map, 11
- gain maps, 11
- galaxy clusters, 20
- glitch, 21, 39
- glitches, 21, 39
- globular cluster, 46

- image, 2, 3, 7–13, 15, 17–25, 27–29, 32–36, 40–42, 45, 51, 56
- ImCat, 20
- installation, 4
- interpolation, 9, 18, 21, 28
- isophotal magnitudes, 43

- J2000, 42

- linear filtering, 19
- local background, 18, 22, 23, 28, 29, 38, 45
- LSB, 20

- machine-learning, 21
- magnitude error, 25, 26, 43
- magnitude errors, 26
- masking, 9
- mean, 6, 17–19, 25, 27, 37, 44, 45, 52, 56
- median, 18, 19, 29
- memory, 3, 10, 17, 20, 25, 32
- mode, 2, 3, 7, 9, 13, 18, 29, 40, 45, 57
- moments, 33, 35–37, 40, 41, 44
- multi-thresholding, 17, 24, 25

- neighbour, 9, 26, 27, 32, 38, 44, 45

neighbours, 9, 32, 38, 44, 45
neural network, 21
NICMOS, 8

parameter file, 11, 13, 33
precession, 42
PSF, 20, 21, 40

requirements, 3
ringing, 19

Schwartz inequality, 20
segmentation, 17, 19, 20
semi-major, 36, 41
semi-minor axis, 36
signal-to-noise ratio, 19
standard deviation, 9, 22, 23, 28, 45, 56
stars, 18, 20, 24–26, 43–45
syntax, 5, 51, 52

threshold, 8, 9, 13, 17, 19, 22–25, 27–29, 33, 34,
39, 43, 56

variance map, 9, 11, 27, 28
vignetting, 29
VOTable, 2

wavelet, 17, 20
WCS, 15, 34, 42
weight map, 2, 9, 11, 27–29
weight maps, 2, 11, 27, 29
window, 33, 39–41
windowed, 39–41
world coordinates, 12

XML, 2, 11

zero-point, 12, 23, 45