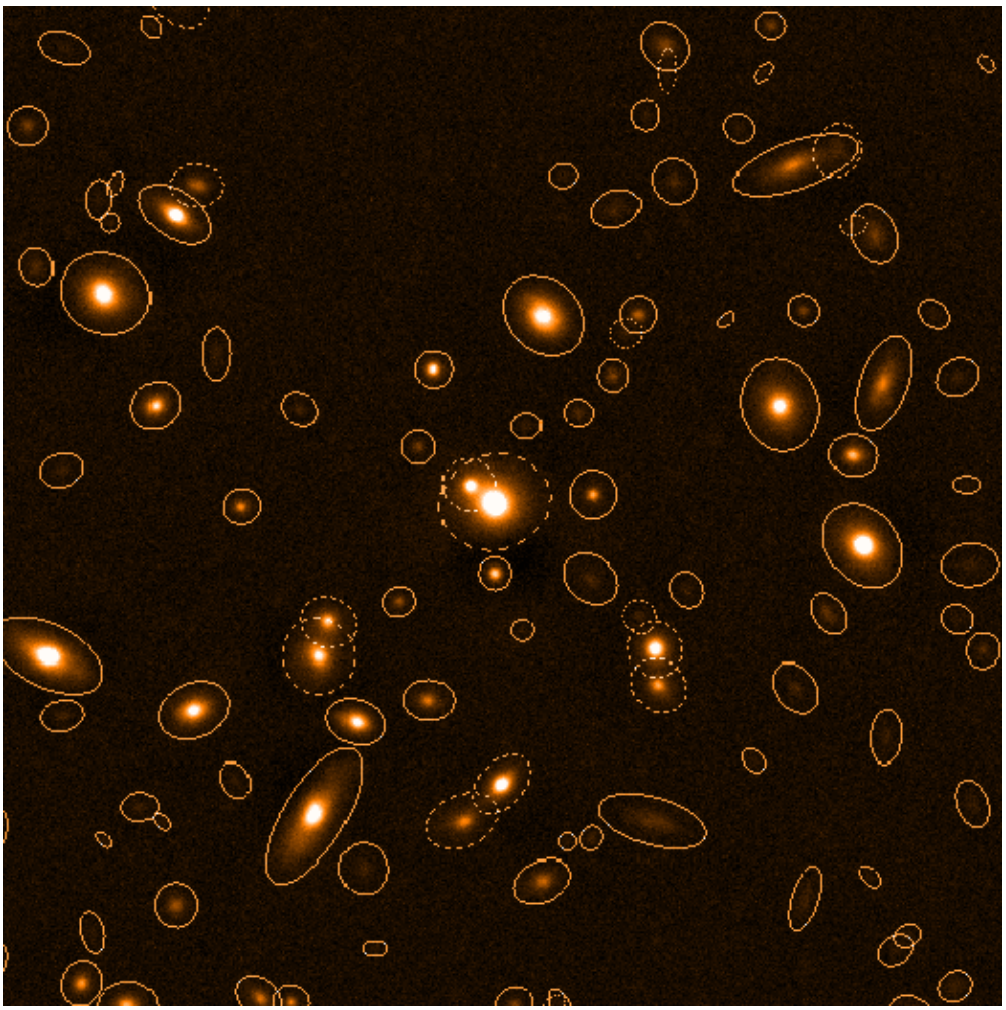


SExtractor Documentation

Release 2.28.0

E. Bertin

Fri Mar 10 2023

CONTENTS

CHAPTER

ONE

CONTENTS

1.1 Introduction

SExtractor (Source-Extractor) is a program that builds a catalog 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 (MEF (Multi-Extension FITS))
- Speed: up to about 50 Mpixel/s or 10,000 sources/s with a 3 GHz processor
- Ability to work with very large images (up to 2Gx2G 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 catalog output of desired parameters only
- Pixel-to-pixel photometry in dual-image mode
- Fast and accurate Point-Spread-Function and galaxy model fitting.
- Handling of weight maps and flag maps.
- Optimum handling of images with variable SNR.
- Built-in catalog cross-identification.
- Special mode for photographic scans.
- XML (eXtensible Markup Language) VOTable-compliant catalog output.
- XSLT (eXtensible Stylesheet Language Transformations) filter sheet provided for convenient access to metadata from a regular web browser.

1.2 License

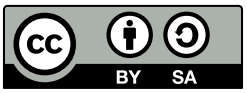
1.2.1 Code

The SExtractor code is licensed under a GPL v3 license:

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.*

1.2.2 Documentation



1.3 Installing the software

1.3.1 Hardware requirements

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

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.

Obtaining SExtractor

For Linux users, the simplest way to have **SExtractor** up and running is to install the standard binary package the comes with your Linux distribution. Run, e.g., `apt-get sextractor` (on Debian) or `dnf sextractor` (Fedora) as root and **SExtractor**, as well as all its dependencies, will automatically be installed. If you decided to install the package this way you may skip the following and move straight to the next section.

However if **SExtractor** is not available in your distribution, or to obtain the most recent version, the **SExtractor** source package can be downloaded from the official GitHub repository . One may choose one of the stable releases, or for the fearless, a copy of the current master development branch.

Software requirements

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

- **ATLAS** V3.6 and above[2],
- **FFTW** V3.0 and above[3],

On Fedora/Redhat distributions for instance, the development packages above are available as `atlas-devel` and `fftw-devel`. Note that **ATLAS** and **FFTW** are not necessary if **SExtractor** is linked with Intel®'s MKL (Math Kernel Library) library.

[2] Use the `--with-atlas` and/or `--with-atlas-incdir` options of the **SExtractor** **configure** script to specify the **ATLAS** library and include paths if **ATLAS** files are installed at unusual locations.

[3] Make sure that **FFTW** has been compiled with **configure** options `--enable-threads --enable-float`.

Installation

To install from the GitHub source package, you must first uncompress the archive:

```
$ unzip sextractor-<version>.zip
```

A new directory called `sextractor-<version>` should now appear at the current location on your disk. Enter the directory and generate the files required by the autotools, which the package relies on:

```
$ cd sextractor-<version>
$ sh autogen.sh
```

A **`configure`** script is created. This script has many options, which may be listed with the `--help` option:

```
$ ./configure --help
```

No options are required for compiling with the default GNU C compiler (**`gcc`**) if all the required libraries are installed at their default locations:

```
$ ./configure
```

Compared to **`gcc`** and the librairies above, the combination of the Intel® compiler (**`icc`**) and the MKL libraries can give the **`SExtractor`** executable a strong boost in performance, thanks to better vectorized code. If **`icc`** and the MKL are installed on your system[4] , you can take advantage of them using

```
$ ./configure --enable-mkl
```

Additionally, if the **`SExtractor`** binary is to be run on a different machine that does not have **`icc`** and the MKL installed (e.g., a cluster computing node), you must configure a partially statically linked executable using

```
$ ./configure --enable-mkl --enable-auto-flags --enable-best-link
```

In all cases, **`SExtractor`** can now be compiled with

```
$ make -j
```

An `src/sex` executable is created. For system-wide installation, run the usual

```
$ sudo make install
```

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

```
$ sex
```

which will return the version number and other basic information (note that some shells require the **`rehash`** command to be run before making a freshly installed executable accessible in the execution path).

[4] The Linux versions of the Intel® compiler and MKL are available for free to academic researchers, students, educators and open source contributors.

1.4 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. Any *-Parameter Value* statement in the command-line overrides the corresponding definition in the configuration file or any default value (see *configuration section*).

1.4.1 Input files

SExtractor accepts images stored in FITS (Flexible Image Transport System) [1]. Both "Basic FITS" (one single header and one single body) and MEF files are recognized. Binary **SExtractor** catalogs produced from MEF images are MEF files themselves. If the catalog output format is set to ASCII, all catalogs from the individual extensions are concatenated in one big file; the EXT_NUMBER catalog parameter can be used to tell which extension the detection belongs to.

Important: Contrary to most other astronomy packages, **SExtractor** does not rely on the FITSIO library and instead uses its own library for managing FITS files. As a consequence, some features of FITS such as image compression/tiling are not supported at this time.

For images with NAXIS > 2, only the first data-plane is loaded. In **SExtractor**, as in all similar programs, FITS axis #1 is traditionally referred to as the *x* axis, and FITS axis #2 as the *y* axis.

Double image mode

When the pixel data for a given field are available using the same pixel grid in several photometric channels, it is often best to measure object characteristics, like magnitudes, exactly at the same positions and through the same apertures for the different channels. This to derive precise color 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 the exact same dimensions, obviously. By repeatedly running **SExtractor** with various "measurement images" while keeping the same "detection image", one ends up with a set of catalogs 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 "χ2 image" [2][1] 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 when providing **SExtractor** with two images instead of one:

```
$ sex detection.fits,measurement.fits
```

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

[1] χ2 images can easily be generated using the **SWarp** package [3].

1.4.2 Output files

Diagnostic files

Two types of files can be generated by **SExtractor**, providing diagnostics about the source extraction process:

- "Check-images" are FITS files containing raster images such as maps of the background model, apertures, etc.. The configuration parameters `CHECKIMAGE_TYPE` and `CHECKIMAGE_NAME` allow the user to provide a list of check-image types and file names, respectively, to be produced by **SExtractor**. A complete list of available check-image types is given in §[chap:paramlist].
- An XML file providing a processing summary and various statistics in VOTable format is written if the `WRITE_XML` switch is set to `Y` (the default). The `XML_NAME` parameter can be used to change the default file name `sex.xml`. The XML file can be displayed with any recent web browser; the XSLT stylesheet installed together with **SExtractor** will automatically translate it into a dynamic, user-friendly web-page. For more advanced usages (e.g., access from a remote web server), alternative XSLT translation URLs may be specified using the `XSL_URL` configuration parameter.

1.4.3 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.

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.

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 O), or in hexadecimal (preceded by *0x*). The hexadecimal format is particularly convenient for writing multiplexed bit values such as binary masks. Environment variables, written as `$HOME` or `${HOME}` are expanded.

Configuration parameter list

Here is a complete list of all the configuration parameters known to **SExtractor**. Please refer to the next sections for a detailed description of their meaning.

1.4.4 The measurement (or catalog) parameter file

In addition to the configuration file detailed above, **SExtractor** requires a file containing the list of measurements ("catalog parameters") that will be listed in the output catalog for every detection. This allows the software to compute only the measurements that are needed. The name of this catalog 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
```

Format

The format of the catalog parameter list is ASCII, and there must be *a single keyword per line*. Presently two kinds of keywords are recognized by **SExtractor**: scalars and vectors. Scalars, like X_IMAGE, produce single numbers in the output catalog. Vectors, like MAG_APER(4) or VIGNET(15,15), produce arrays of numbers. The ordering of measurements in the output catalog is identical to that of the keywords in the parameter list. Comments are allowed, they must begin with a #.

Variants

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

Fluxes and magnitudes

Fluxes may be expressed in counts (ADU (Analog-to-Digital Unit)s) or as Pogson [4] 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, ... The `MAG_ZEROPOINT` configuration parameter sets the magnitude zero-point of magnitudes:

$$\texttt{MAG} = \begin{cases} \text{MAG\_ZEROPOINT} - 2.5 \log_{10} \texttt{FLUX} & \text{if } \texttt{FLUX} > 0 \\ 99.0 & \text{otherwise.} \end{cases} \tag{1.1}$$

Flux and magnitude uncertainties

Flux uncertainties (error estimates) follow a scheme similar to that of fluxes. They are prefixed with FLUXERR_, as in FLUXERR_AUTO or FLUXERR_ISO. Magnitude uncertainties start with MAGERR_, for instance: MAGERR_AUTO, MAGERR_ISO,... They are computed using

$$\texttt{MAGERR} = \begin{cases} \frac{2.5}{\ln 10} (\texttt{FLUXERR}/\texttt{FLUX}) & \text{if } \texttt{FLUX} > 0 \\ 99.0 & \text{otherwise.} \end{cases} \tag{1.2}$$

Pixel values and Surface brightnesses

Pixel values (averaged or not) p are expressed in counts (ADUs). There is no specific prefix (THRESHOLD, BACKGROUND, FLUX_MAX, etc.). Surface brightnesses are given in magnitudes per square arcsecond, and prefixed with MU_ e.g., MU_THRESHOLD, MU_MAX, ... The conversion to surface brightness relies on the `MAG_ZEROPOINT` and the `PIXEL_SCALE` configuration parameters:

$$\texttt{MU} = \begin{cases} \text{MAG\_ZEROPOINT} - 2.5\log_{10}(p \times \text{PIXEL\_SCALE}^{\,2}) & \text{if } p > 0 \\ 99.0 & \text{otherwise.} \end{cases} \tag{1.3}$$

Setting `PIXEL_SCALE` to 0 instructs **`SExtractor`** to compute the pixel scale from the local Jacobian of the astrometric deprojection, based on the celestial WCS info [5] in the FITS image header, if available.

Positions and shapes

Positions, distances and position angles are computed in pixel coordinates. They may be expressed in image pixels, world coordinates, or in celestial coordinates, depending on the suffix:

_IMAGE
: Measurements are given in pixel coordinates, in units of pixels. For example: Y_IMAGE, ERRAWIN_IMAGE, THETA_IMAGE etc. Following the FITS convention, in **`SExtractor`** the center of the first image pixel has coordinates (1.0,1.0). Position angles are counted from the x axis (axis 1), positive towards the y axis (axis 2)

_WORLD
: Measurements are given in so-called "world coordinates", converted from pixel coordinates using the local Jacobian of the transformation between both systems. This requires WCS metadata [6] to be present in the FITS image header(s). Position angles are counted from the first world axis, positive towards the second world axis.

_SKY, _J2000, _B1950
: Measurements are given in celestial (equatorial) coordinates, converted from pixel coordinates using the local Jacobian of the transformation between both systems. Positions and distances are in units of degrees. This requires celestial WCS metadata [5] to be present in the FITS image header(s). _SKY measurements are given in the native world coordinate system. _J2000 and _B1950 measurements are automatically converted from the native WCS, taking into account the change of reference frame. In all cases, positions angles are counted East-of-North.

_FOCAL
: Measurements are given in "focal plane coordinates", which are actually projected coordinates, in degrees. This requires WCS metadata [6] to be present in the FITS image header(s). The computation of focal plane coordinates from pixel coordinates is similar to that of _SKY coordinates except that they are not de-projected and remain Cartesian. The main purpose of focal plane coordinates is to provide a common system for all the chips in a mosaic camera.

Note: Conversion to _FOCAL coordinates is available only for a limited subset of measurements.

Important: The WCS library currently implemented in **`SExtractor`** is a customized version of an early implementation (v1.1.1) by Calabretta. Several projections from later versions and alternative astrometric descriptions such as AST or that of original DSS plates are not supported at this time.

Measurement parameter list

Below is an exhaustive list of all the measurement parameters known to **SExtractor**. Please refer to the next sections for a detailed description of their meaning.

Table 1.1: **SExtractor** measurement parameters

Name	Unit	Description
NUMBER	...	Running object number
ID_PARENT	...	Parent ID (before deblending)
EXT_NUMBER	...	FITS extension number
FLAGS	...	*Source extraction flags*
FLAGS_WEIGHT	...	*Weighting flags*
IMAFLAGS_ISO	...	*External flags combined within the isophotal footprint*
NIMAFLAGS_ISO	...	*Number of combined external flags*
FLUX_ISO	count	*Isophotal flux*
FLUXERR_ISO	count	*RMS error estimate for the isophotal flux*
MAG_ISO	magnitude	*Isophotal magnitude*
MAGERR_ISO	magnitude	*RMS error estimate for the isophotal magnitude*
FLUX_ISOCOR	count	*Corrected isophotal flux*
FLUXERR_ISOCOR	count	*RMS error estimate for the corrected isophotal flux*
MAG_ISOCOR	magnitude	*Corrected isophotal magnitude*
MAGERR_ISOCOR	magnitude	*RMS error estimate for the corrected isophotal magnitude*
FLUX_APER	count	*Flux(es) within fixed circular aperture(s)*
FLUXERR_APER	count	*RMS error estimate(s) for the flux(es) within fixed circular aperture(s)*
MAG_APER	magnitude	*Circular aperture magnitude(s)*
MAGERR_APER	magnitude	*RMS error estimate(s) for circular aperture magnitude(s)*
FLUX_AUTO	count	*Kron-like automated aperture flux*
FLUXERR_AUTO	count	*RMS error estimate for Kron-like automated aperture flux*
MAG_AUTO	magnitude	*Kron-like automated aperture magnitude*
MAGERR_AUTO	magnitude	*RMS error estimate for Kron-like automated aperture magnitude*
KRON_RADIUS	...	*Kron radius in units of A or B*
FLUX_PETRO	count	*Petrosian-like aperture flux*
FLUXERR_PETRO	count	*RMS error estimate for Petrosian-like aperture flux*
MAG_PETRO	magnitude	*Petrosian-like aperture magnitude*
MAGERR_PETRO	magnitude	*RMS error estimate for Petrosian-like aperture magnitude*
PETRO_RADIUS	...	*Petrosian radius in units of A or B*
BACKGROUND	count	*Background level at the position of the centroid*
X_IMAGE	pixel	*Pixel x coordinate* of the *isophotal image centroid*
Y_IMAGE	pixel	*Pixel y coordinate* of the *isophotal image centroid*
X_FOCAL	degree	*Focal plane x coordinate* of *isophotal image centroid*
Y_FOCAL	degree	*Focal plane y coordinate* of *isophotal image centroid*
X_WORLD	...	*World x coordinate* of the *isophotal image centroid*
Y_WORLD	...	*World y coordinate* of the *isophotal image centroid*
ALPHA_SKY	degree	*Native right ascension* of the *isophotal image centroid*
DELTA_SKY	degree	*Native declination* of the *isophotal image centroid*
ALPHA_J2000	degree	*J2000 right ascension* of the *isophotal image centroid*
DELTA_J2000	degree	*J2000 declination* of the *isophotal image centroid*
ALPHA_B1950	degree	*B1950 right ascension* of the *isophotal image centroid*
DELTA_B1950	degree	*B1950 declination* of the *isophotal image centroid*
ERRX2_IMAGE	$pixel^2$	*Estimated variance of isophotal image centroid x coordinate*
ERRY2_IMAGE	$pixel^2$	*Estimated variance of isophotal image centroid y coordinate*

continues on next page

Table 1.1 – continued from previous page

Name	Unit	Description
ERRXY_IMAGE	pixel^2	*Estimated covariance of isophotal image centroid x and y coordinates*
ERRA_IMAGE	pixel	*Major axis of the isophotal image centroid error ellipse*
ERRB_IMAGE	pixel	*Minor axis of the isophotal image centroid error ellipse*
ERRTHETA_IMAGE	degree	*Position angle of the isophotal image centroid ellipse*
ERRCXX_IMAGE	pixel^{-2}	*Isophotal image centroid Cxx error ellipse parameter*
ERRCYY_IMAGE	pixel^{-2}	*Isophotal image centroid Cyy error ellipse parameter*
ERRCXY_IMAGE	pixel^{-2}	*Isophotal image centroid Cxy error ellipse parameter*
XPEAK_IMAGE	pixel	*Pixel x coordinate* of the *brightest pixel*
YPEAK_IMAGE	pixel	*Pixel y coordinate* of the *brightest pixel*
XPEAK_FOCAL	degree	*Focal plane x coordinate* of the *brightest pixel*
YPEAK_FOCAL	degree	*Focal plane y coordinate* of the *brightest pixel*
XPEAK_WORLD	...	*World x coordinate* of the *brightest pixel*
YPEAK_WORLD	...	*World y coordinate* of the *brightest pixel*
ALPHAPEAK_SKY	degree	*Native right ascension* of the *brightest pixel*
DELTAPEAK_SKY	degree	*Native declination* of the *brightest pixel*
ALPHAPEAK_J2000	degree	*J2000 right ascension* of the *brightest pixel*
DELTAPEAK_J2000	degree	*J2000 declination* of the *brightest pixel*
ALPHAPEAK_B1950	degree	*J2000 right ascension* of the *brightest pixel*
DELTAPEAK_B1950	degree	*J2000 declination* of the *brightest pixel*
XMIN_IMAGE	pixel	*Minimum x coordinate among detected pixels*
YMIN_IMAGE	pixel	*Minimum y coordinate among detected pixels*
XMAX_IMAGE	pixel	*Maximum x coordinate among detected pixels*
YMAX_IMAGE	pixel	*Maximum y coordinate among detected pixels*
XWIN_IMAGE	pixel	*x coordinate of windowed image centroid*
YWIN_IMAGE	pixel	*y coordinate of windowed image centroid*
ERRX2WIN_IMAGE	pixel^2	*Estimated variance of windowed image centroid x coordinate*
ERRY2WIN_IMAGE	pixel^2	*Estimated variance of windowed image centroid y coordinate*
ERRXYWIN_IMAGE	pixel^2	*Estimated covariance of windowed image centroid x and y coordinates*
ERRAWIN_IMAGE	pixel	*Major axis of the windowed image centroid error ellipse*
ERRBWIN_IMAGE	pixel	*Minor axis of the windowed image centroid error ellipse*
ERRTHETAWIN_IMAGE	degree	*Position angle of the windowed image centroid ellipse*
ERRCXXWIN_IMAGE	pixel^{-2}	*Windowed image centroid Cxx error ellipse parameter*
ERRCYYWIN_IMAGE	pixel^{-2}	*Windowed image centroid Cyy error ellipse parameter*
ERRCXYWIN_IMAGE	pixel^{-2}	*Windowed image centroid Cxy error ellipse parameter*
FLAGS_WIN	...	*Windowed measurement flags*
X2_IMAGE	pixel^2	*Isophotal image 2nd order central moment in x*
Y2_IMAGE	pixel^2	*Isophotal image 2nd order central moment in y*
XY_IMAGE	pixel^2	*Isophotal image 2nd order central cross-moment in xy*
A_IMAGE	pixel	*Isophotal image major axis*
B_IMAGE	pixel	*Isophotal image minor axis*
THETA_IMAGE	degree	*Isophotal image position angle*
ELONGATION	...	*A_IMAGE / B_IMAGE*
ELLIPTICITY	...	*1 - B_IMAGE / A_IMAGE*
CXX_IMAGE	pixel^{-2}	*Isophotal image Cxx ellipse parameter*
CYY_IMAGE	pixel^{-2}	*Isophotal image Cyy ellipse parameter*
CXY_IMAGE	pixel^{-2}	*Isophotal image Cxy ellipse parameter*
ISOAREAF_IMAGE	pixel^2	*Isophotal area (filtered) above Detection threshold*
ISOAREA_IMAGE	pixel^2	*Isophotal area above Analysis threshold*
X2WIN_IMAGE	pixel^2	*Windowed image 2nd order central moment in x*
Y2WIN_IMAGE	pixel^2	*Windowed image 2nd order central moment in y*
XYWIN_IMAGE	pixel^2	*Windowed image 2nd order central cross-moment in xy*

continues on next page

Table 1.1 – continued from previous page

Name	Unit	Description
CXXWIN_IMAGE	pixel^{-2}	*Windowed image Cxx ellipse parameter*
CYYWIN_IMAGE	pixel^{-2}	*Windowed image Cyy ellipse parameter*
CXYWIN_IMAGE	pixel^{-2}	*Windowed image Cxy ellipse parameter*
AWIN_IMAGE	pixel	*Windowed image major axis*
BWIN_IMAGE	pixel	*Windowed image minor axis*
THETAWIN_IMAGE	degree	*Windowed image position angle*
CLASS_STAR	...	*Star/galaxy classifier*
VECTOR_MODEL	...	*Model-fitting coefficients*
VECTOR_MODELERR	...	*Model-fitting coefficient uncertainties*
MATRIX_MODELERR	...	*Model-fitting covariance matrix*
CHI2_MODEL	...	*Reduced modified Chi2 of the fit*
FLAGS_MODEL	...	*Model-fitting flags*
NITER_MODEL	...	*Number of model-fitting iterations*
FLUX_MODEL	count	*Flux from model-fitting*
FLUXERR_MODEL	count	*RMS error estimate for the model-fitting flux*
MAG_MODEL	magnitude	*Magnitude from model-fitting*
MAGERR_MODEL	count	*RMS error estimate for the model-fitting magnitude*
FLUX_MAX_MODEL	count	*Peak model flux above the background*
FLUX_EFF_MODEL	count	*Effective model flux above the background*
FLUX_MEAN_MODEL	count	*Mean effective model flux above the background*
MU_MAX_MODEL	mag.arcsec^{-2}	*Peak model surface brightness above the background*
MU_EFF_MODEL	mag.arcsec^{-2}	*Effective model surface brightness above the background*
MU_MEAN_MODEL	mag.arcsec^{-2}	*Mean effective model surface brightness above the background*
XMODEL_IMAGE	pixel	*x coordinate from model-fitting*
YMODEL_IMAGE	pixel	*y coordinate from model-fitting*
SPREAD_MODEL	...	*Spread parameter from model-fitting*
SPREADERR_MODEL	...	*RMS error estimate on spread parameter from model-fitting*

1.5 Processing

The complete analysis of an image is fully automated (Fig. 1.1). Two passes are made through the data. During the first pass, a model of the sky background is built, and several global statistics are computed. During the second pass, image pixels are background-subtracted, filtered and segmented on-the-fly. Detections are then deblended, pruned ("CLEANed"), and enter the measurement phase. Finally, the measured quantities are written to the output catalog, after cross-matching with an optional input list.

The following sections describe each of these operations in more detail.

1.5.1 Modeling the background

On linear detectors, the value measured at each pixel is the sum of a "background" signal and light coming from the sources of interest. To be able to detect the faintest objects and make accurate measurements, **SExtractor** needs first computing a precise estimate of the background level at any position of the image: a *background map*. Strictly speaking, there should be one background map per source, that is, what would the image look like if that very source was missing. But, at least for detection, one can start by assuming that most discrete sources do not overlap too severely — which is generally the case for high galactic latitude fields —, and that the background varies smoothly across the field.

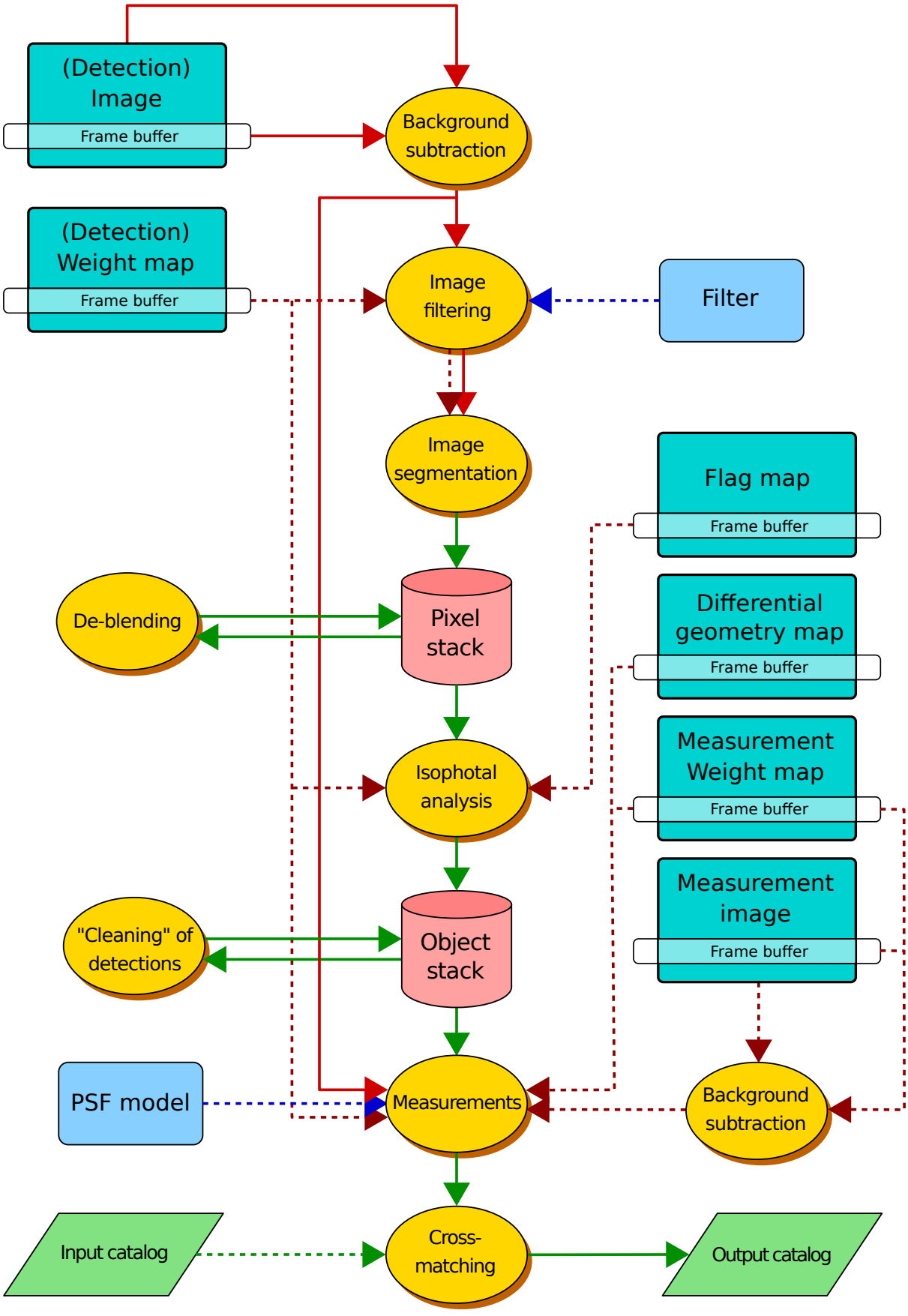


Fig. 1.1: Layout of the main **SExtractor** procedures. *Dashed arrows* represent optional inputs.

Background estimation

To compute the background map, **SExtractor** makes a first pass through the pixel data, estimating the local background in each mesh of a rectangular grid that covers the whole frame. The background estimator is a combination of $\kappa\,\sigma$ clipping and mode estimation, similar to Stetson's **DAOPHOT** program [7, 8].

Briefly, the local background histogram is clipped iteratively until convergence at $\pm 3\sigma$ around its median. The mode of the histogram is estimated using the semi-empirical relation

$$\text{Mode} - \text{Mean} \approx \alpha_{\text{Pearson}} \times (\text{Median} - \text{Mean}), \tag{1.4}$$

and therefore

$$\text{Mode} \approx \alpha_{\text{Pearson}} \times \text{Median} - (\alpha_{\text{Pearson}} - 1) \times \text{Mean}, \tag{1.5}$$

with $\alpha_{\text{Pearson}} \approx 3$ [9, 10, 11].

Using simulated images, [12] found $\alpha_{\text{Pearson}} \approx 2.5$ more appropriate for determining the mode of the background histogram in **SExtractor**. However, extensive photometric assessments carried out in the context of the Dark Energy Survey later showed that this value leads to a slight overestimation of the local background [13], and a higher $\alpha_{\text{Pearson}} \approx 3.5$ was adopted in Dark Energy Survey. Since **SExtractor** v2.28, α_{Pearson} can be changed using the `BACK_PEARSON` configuration parameter. However, for the sake of compatibility with previous results, the default value of `BACK_PEARSON` remains `2.5`.

Fig. 1.2 shows that the mode estimation in (1.5) is considerably less affected by source crowding than a simple clipped mean [14, 15] but it is $\approx 30\%$ noisier. Obviously (1.5) is not valid for any distribution; **SExtractor** falls back to a simple median for estimating the local background value if the mode and the median disagree by more than 30%.

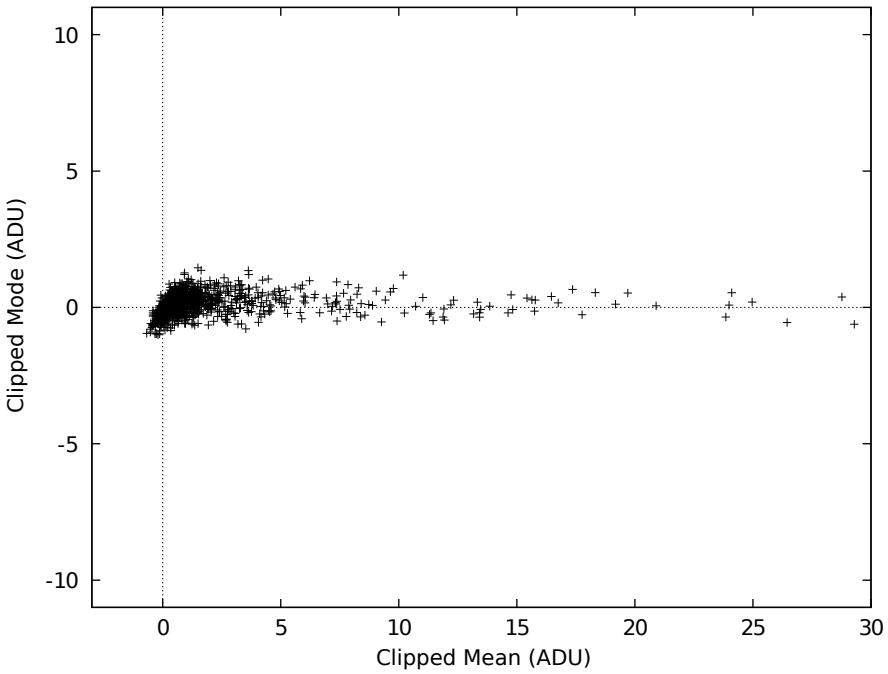


Fig. 1.2: Simulations of 32×32 pixels background meshes contamined by random Gaussian profiles. The true background lies at 0 ADUs. While being a bit noisier, the clipped "mode" gives a more robust estimate than the clipped mean in crowded regions.

Background map

Once the grid is set up, a median filter can be applied to suppress possible local overestimations due to bright stars. The final background map is simply a (natural) bicubic-spline interpolation between the meshes of the grid. Median filtering helps reducing possible ringing effects of the bicubic-spline around bright features.

In parallel with the making of the background map, an *RMS background map*, that is, a map of the background noise standard deviation in the image is produced. It may be used as an internal weight map if the `WEIGHT_TYPE` configuration parameter is to `BACKGROUND` (see *Weight-map formats*).

Configuration and tuning

Note: All background configuration parameters also affect background-RMS maps.

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 must be found by the user. Typically, for reasonably sampled images, a width[1] of 32 to 512 pixels works well.

The user has some control over background map filtering by specifying the size of the median filter `BACK_FILTERSIZE`. A width and height of 1 means that no filtering is applied to the background grid. A 3×3 filtering is sufficient in most cases. Larger dimensions may occasionally be used to compensate for small background mesh sizes, or in the presence of large image artifacts. In some specific cases it is desirable to median-filter only background meshes that have values exceeding some threshold above the filtered value. This differential threshold is set by the `BACK_FILTERTHRESH` parameter, in ADUs.

By default, the computed background maps are automatically subtracted from input images. However there are some situations where it is more appropriate to subtract a *constant* from the image (e.g., images with strongly non-Gaussian background noise PDF (Probability Density Function)s). The `BACK_TYPE` configuration parameter (set to `AUTO` by default) may be switched to `MANUAL` to force the value specified by the `BACK_VALUE` parameter to be subtracted from the input image, instead of the background map. `BACK_VALUE` is `0` by default.

Computing cost

The background estimation operation is generally I/O (Input/Output)-bound, unless the image file already resides in the disk cache.

1.5.2 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 it 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 ADU^2) 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 neighboring pixels. In theory, all non-zero covariances within the geometrical limits of the analysed patterns should

[1] It is possible to specify rectangular background meshes, although it is advised to use square ones, except in special cases (background varying rapidly along the x or y axis).

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.

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. Those weight-maps can either be read from a FITS file, whose name is specified by the WEIGHT_IMAGE keyword, or computed internally. Valid WEIGHT_TYPE values 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 the *background model*[1]. 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 $\propto \frac{1}{weight}$), 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.

Effect 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 the background 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 =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 should be used to avoid this problem. If convolution filtering 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 retinae) are not affected.
3. The cleaning process takes into account the exact individual thresholds assigned to each pixel for deciding about the fate of faint detections.

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

4. Error estimates like FLUXISO_ERR , ERRA_IMAGE , etc. also make use of individual variances. 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, the median value over the image being set by the GAIN configuration parameter. In other words, it is then supposed that the changes in noise intensities seen over the images are due to gain changes. 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.

Combining weight maps

All the weighting options listed in *Weight-map formats* can be applied separately to detection and measurement images (*Using SExtractor*), 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 (see *Using SExtractor*). WEIGHT_IMAGE can be ignored for BACKGROUND WEIGHT_TYPE. 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.

Weight-map flags: FLAGS_WEIGHT

The FLAGS_WEIGHT catalog parameter flags various issues which may happen during the weighting process (see the *Flagging* section for details on how flags are managed in **SExtractor**):

Table 1.2: FLAGS_WEIGHT description

Vali	Meaning
1	at least one measurement-image weight with a value below WEIGHT_THRESH *overlaps* the isophotal footprint of the detected object
2	at least one measurement-image weight with a value below WEIGHT_THRESH *touches* the isophotal footprint of the detected object

1.5.3 Flagging

Both *internal* and *external* flags are available for each detection as catalog parameters:

- Several sets of internal flags are produced by the various processes within **SExtractor**:
  - *Extraction flags* (FLAGS)
  - *Weighting flags* (FLAGS_WEIGHT)
  - *Windowed measurement flags* (FLAGS_WIN)
  - *Model-fitting flags* (FLAGS_MODEL);

- *External flags* are derived from *flag maps*. Flag maps are images in integer format having the same dimensions as the science images, with pixel values that 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 catalog.

Each catalog parameter comprises several flag bits as a sum of powers of 2 coded in decimal. For example, a saturated detection close to an image boundary will have FLAGS = 4+8 = 12 (see below).

Extraction flags: FLAGS

FLAGS contains 8 flag bits with basic warnings about the source extraction process, in order of increasing concern.

Table 1.3: FLAGS description

Vali	Meaning
1	*aperture photometry* is likely to be biased by neighboring sources or by more than 10% of bad pixels in any aperture
2	the object has been deblended
4	at least one object pixel is saturated
8	the isophotal footprint of the detected object is truncated (too close to an image boundary)
16	at least one photometric aperture is incomplete or corrupted (hitting buffer or memory limits)
32	the isophotal footprint is incomplete or corrupted (hitting buffer or memory limits)
64	a memory overflow occurred during deblending
128	a memory overflow occurred during extraction

External flags: IMAFLAGS_ISO

The processing of external flags is triggered whenever IMAFLAGS_ISO or NIMAFLAGS_ISO are present in the catalog parameter file. The file names of the images containing flag maps in FITS format are set with the `FLAG_IMAGE` configuration keyword. Flag maps must be stored as 2-dimensional arrays of 8, 16 or 32 bits integers. The integers may be signed or unsigned, although for Boolean operations the sign bit will be interpreted as bit 7, 15 or 31, depending on integer size. Obviously all flag maps must have the same dimensions as the image used for detection. Flag maps can be created using, e.g., the `WeightWatcher` software [16].

The values of flag map pixels that overlap the isophotal area of a given detected object are combined and stored in the catalog as the long integer IMAFLAGS_ISO. The `FLAG_TYPE` configuration keyword sets the type of combination applied individually to each flag map. 5 types of combination are currently supported:

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

The NIMAFLAGS_ISO catalog parameter contains the number(s) of relevant flag map pixels: the number of non-zero flag map pixels if `FLAG_TYPE` is set to `OR` or `AND`, or the number of pixels with value IMAFLAGS_ISO if `FLAG_TYPE` is set to `MIN` , `MAX`, or `MOST`.

1.6 Measurements

Once sources have been detected and deblended, they enter the measurement phase. **SExtractor** performs three categories of measurements: isophotal, full, and model-fitting.

Isophotal
: Measurements are made on the isophotal object footprints, which are defined on the filtered detection image. Only pixels with values above the detection threshold (set with DETECT_THRESH) are considered[1], which makes the analysis extremely fast, but obviously strongly dependent on the threshold itself. This is an issue particularly when the amplitude of the bakground noise varies over the image. Many of the isophotal measurements (e.g., X_IMAGE, Y_IMAGE, FLUX_ISO) are necessary for the internal operations of **SExtractor** and are therefore executed even if they are not requested.

Full
: Measurements have access to all pixels of the image. These measurements are generally more sophisticated, less affected by variable biases induced by the detection threshold, and still reasonably fast. They are done at a later stage of the processing, after CLEANing and MASKing.

Model-fitting
: Measurements require PSF models[2]. They are often the most accurate and can recover the flux of saturated objects. They are also much slower, allowing typically only a few tens of objects to be processed every second.

1.6.1 Isophotal measurements

Position and shape

The following quantities are derived from the spatial distribution $\mathcal{S}$ of pixels detected above the detection threshold (see *description*).

Important: Unless otherwise noted, the pixel values used for computing "isophotal" positions and shapes are taken from the filtered, background-subtracted detection image.

Note: 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 coordinates or _WORLD, _SKY, _J2000 or _B1950 for WCS coordinates (see *Positions and shapes*).

Limits: XMIN, YMIN, XMAX, YMAX

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

$$\texttt{XMIN} = \min_{i \in \mathcal{S}} x_i, \tag{1.6}$$

$$\texttt{YMIN} = \min_{i \in \mathcal{S}} y_i, \tag{1.7}$$

$$\texttt{XMAX} = \max_{i \in \mathcal{S}} x_i, \tag{1.8}$$

$$\texttt{YMAX} = \max_{i \in \mathcal{S}} y_i, \tag{1.9}$$

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

[1] For some isophotal measurements pixel values also have to exceed the local analysis threshold set with ANALYSIS_THRESH.
[2] PSF models be computed using the **PSFEx** package.

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:

$$\texttt{X} \quad = \quad \overline{x} = \frac{\sum_{i\in\mathcal{S}} p_i^{(f)} x_i}{\sum_{i\in\mathcal{S}} p_i^{(f)}}, \tag{1.10}$$

$$\texttt{Y} \quad = \quad \overline{y} = \frac{\sum_{i\in\mathcal{S}} p_i^{(f)} y_i}{\sum_{i\in\mathcal{S}} p_i^{(f)}}, \tag{1.11}$$

where $p_i^{(f)}$ is the value of the pixel i in the (filtered) detection image. In practice, x_i and y_i are summed relative to XMIN and YMIN in order to reduce roundoff errors in the summing.

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, hence 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.

Second-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:

$$\texttt{X2} \quad = \overline{x^2} = \quad \frac{\sum_{i\in\mathcal{S}} p_i^{(f)} x_i^2}{\sum_{i\in\mathcal{S}} p_i^{(f)}} - \overline{x}^2, \tag{1.12}$$

$$\texttt{Y2} \quad = \overline{y^2} = \quad \frac{\sum_{i\in\mathcal{S}} p_i^{(f)} y_i^2}{\sum_{i\in\mathcal{S}} p_i^{(f)}} - \overline{y}^2, \tag{1.13}$$

$$\texttt{XY} \quad = \overline{xy} = \quad \frac{\sum_{i\in\mathcal{S}} p_i^{(f)} x_i y_i}{\sum_{i\in\mathcal{S}} p_i^{(f)}} - \overline{x}\,\overline{y}, \tag{1.14}$$

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.

Basic shape parameters: A, B, THETA

These parameters are intended to describe the detected object as an elliptical shape. A and B are the lengths of the semi-major and semi-minor axes, respectively. More precisely, they represent the maximum and minimum spatial 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 shape parameters are computed. 2nd-order moments can easily be expressed in a referential rotated from the x, y image coordinate system by an angle $+\theta$:

$$\begin{array}{rcrcrcl} \overline{x_\theta^2} & = & \cos^2\theta\,\overline{x^2} & + \sin^2\theta\,\overline{y^2} & - 2\cos\theta\sin\theta\,\overline{xy}, \\ \overline{y_\theta^2} & = & \sin^2\theta\,\overline{x^2} & + \cos^2\theta\,\overline{y^2} & + 2\cos\theta\sin\theta\,\overline{xy}, \\ \overline{xy_\theta} & = & \cos\theta\sin\theta\,\overline{x^2} & - \cos\theta\sin\theta\,\overline{y^2} & + (\cos^2\theta - \sin^2\theta)\,\overline{xy}. \end{array} \tag{1.15}$$

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

$$\left.\frac{\partial\overline{x_\theta^2}}{\partial\theta}\right|_{\theta_0} = 0, \tag{1.16}$$

which leads to

$$2\cos\theta\sin\theta_0\,(\overline{y^2} - \overline{x^2}) + 2(\cos^2\theta_0 - \sin^2\theta_0)\,\overline{xy} = 0. \tag{1.17}$$

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

$$\tan 2\theta_0 = 2\frac{\overline{xy}}{\overline{x^2} - \overline{y^2}}, \tag{1.18}$$

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 (1.18). By definition, THETA is the position angle for which $\overline{x_\theta^2}$ is maximized. THETA is therefore the solution to (1.18) that has the same sign as the covariance $\overline{xy}$. A and B can now simply be expressed as:

$$\mathtt{A}^2 = \overline{x^2}_{\mathtt{THETA}}, \quad \text{and} \tag{1.19}$$

$$\mathtt{B}^2 = \overline{y^2}_{\mathtt{THETA}}. \tag{1.20}$$

A and B can be computed directly from the 2nd-order moments, using the following equations derived from (1.15) after some algebra:

$$\mathtt{A}^2 = \frac{\overline{x^2} + \overline{y^2}}{2} + \sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}, \tag{1.21}$$

$$\mathtt{B}^2 = \frac{\overline{x^2} + \overline{y^2}}{2}, -\sqrt{\left(\frac{\overline{x^2} - \overline{y^2}}{2}\right)^2 + \overline{xy}^2}. \tag{1.22}$$

Note that A and B are exactly halves the a and b parameters computed by the COSMOS image analyser [17]. Actually, a and b are defined in [17] 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 analyzed object.

By-products of shape parameters: ELONGATION and ELLIPTICITY

These parameters[1] are directly derived from A and B:

$$\mathtt{ELONGATION} = \frac{\mathtt{A}}{\mathtt{B}} \quad \text{and} \tag{1.23}$$

$$\mathtt{ELLIPTICITY} = 1 - \frac{\mathtt{B}}{\mathtt{A}}. \tag{1.24}$$

[1] These parameters are dimensionless, and for historical reasons do not accept suffixes such as _IMAGE or _WORLD.

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

$$\mathtt{CXX}(x-\overline{x})^2 + \mathtt{CYY}(y-\overline{y})^2 + \mathtt{CXY}(x-\overline{x})(y-\overline{y}) = R^2, \tag{1.25}$$

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. 1.3). Ellipse parameters can be derived from the 2nd order moments:

$$\mathtt{CXX} = \frac{\cos^2 \mathtt{THETA}}{\mathtt{A}^2} + \frac{\sin^2 \mathtt{THETA}}{\mathtt{B}^2} = \frac{\overline{y^2}}{\overline{x^2}\,\overline{y^2} - \overline{xy}^2} \tag{1.26}$$

$$\mathtt{CYY} = \frac{\sin^2 \mathtt{THETA}}{\mathtt{A}^2} + \frac{\cos^2 \mathtt{THETA}}{\mathtt{B}^2} = \frac{\overline{x^2}}{\overline{x^2}\,\overline{y^2} - \overline{xy}^2} \tag{1.27}$$

$$\mathtt{CXY} = 2\cos \mathtt{THETA} \sin \mathtt{THETA} \left(\frac{1}{\mathtt{A}^2} - \frac{1}{\mathtt{B}^2}\right) = -2\,\frac{\overline{xy}}{\overline{x^2}\,\overline{y^2} - \overline{xy}^2}. \tag{1.28}$$

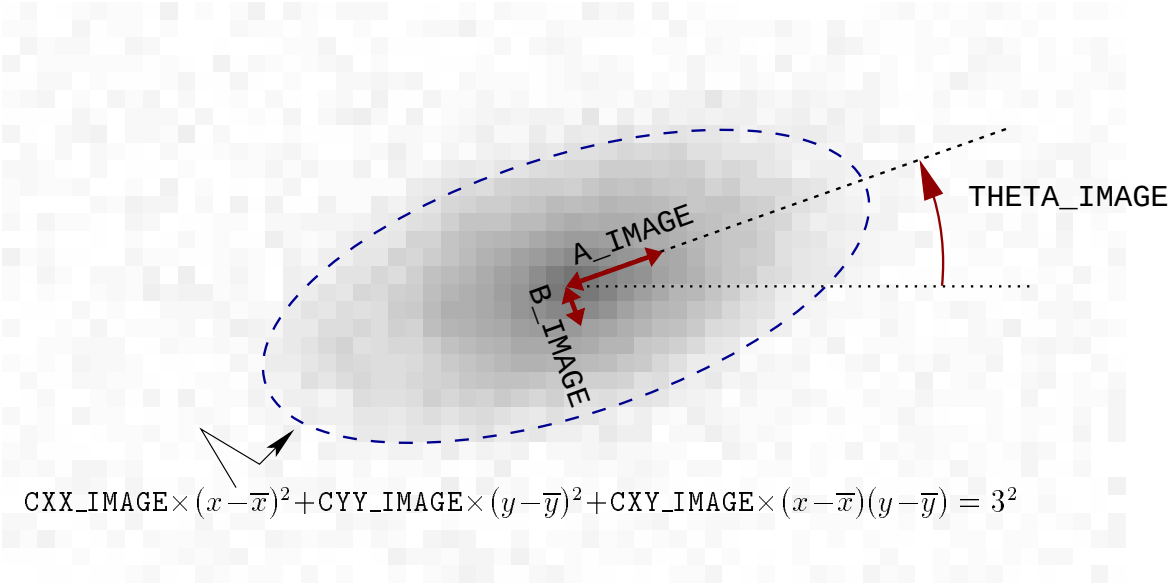


Fig. 1.3: Meaning of basic shape parameters.

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

Uncertainties on the position of the barycenter can be estimated using photon statistics. In practice, such estimates are a lower-value of the full uncertainties because they do not include, for instance, the contribution of detection biases or contamination by neighbors. Furthermore, **SExtractor** does not currently take into account possible

correlations of the noise between adjacent pixels. Hence variances simply write:

$$\texttt{ERRX2} \quad = \mathrm{var}(\overline{x}) = \quad \frac{\sum_{i\in\mathcal{S}} \sigma_i^2 (x_i - \overline{x})^2}{\left(\sum_{i\in\mathcal{S}} p_i^{(f)}\right)^2}, \tag{1.29}$$

$$\texttt{ERRY2} \quad = \mathrm{var}(\overline{y}) = \quad \frac{\sum_{i\in\mathcal{S}} \sigma_i^2 (y_i - \overline{y})^2}{\left(\sum_{i\in\mathcal{S}} p_i^{(f)}\right)^2}, \tag{1.30}$$

$$\texttt{ERRXY} \quad = \mathrm{cov}(\overline{x}, \overline{y}) = \quad \frac{\sum_{i\in\mathcal{S}} \sigma_i^2 (x_i - \overline{x})(y_i - \overline{y})}{\left(\sum_{i\in\mathcal{S}} p_i^{(f)}\right)^2}. \tag{1.31}$$

σ_i is the flux uncertainty estimated for pixel i:

$$\sigma_i^2 = {\sigma_B}_i^2 + \frac{p_i^{(f)}}{g_i},$$

where σ_{Bi} is the local background noise and g_i the local gain — conversion factor — for pixel i (see *Effect of weighting* 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 for *basic shape parameters*:

$$\texttt{ERRA}^2 \quad = \quad \frac{\mathrm{var}(\overline{x}) + \mathrm{var}(\overline{y})}{2} + \sqrt{\left(\frac{\mathrm{var}(\overline{x}) - \mathrm{var}(\overline{y})}{2}\right)^2 + \mathrm{cov}^2(\overline{x}, \overline{y})}, \tag{1.32}$$

$$\texttt{ERRB}^2 \quad = \quad \frac{\mathrm{var}(\overline{x}) + \mathrm{var}(\overline{y})}{2} - \sqrt{\left(\frac{\mathrm{var}(\overline{x}) - \mathrm{var}(\overline{y})}{2}\right)^2 + \mathrm{cov}^2(\overline{x}, \overline{y})}, \tag{1.33}$$

$$\tan(2\texttt{ERRTHETA}) \quad = \quad 2\,\frac{\mathrm{cov}(\overline{x}, \overline{y})}{\mathrm{var}(\overline{x}) - \mathrm{var}(\overline{y})}. \tag{1.34}$$

And the error ellipse parameters are:

$$\texttt{ERRCXX} \quad = \quad \frac{\cos^2 \texttt{ERRTHETA}}{\texttt{ERRA}^2} + \frac{\sin^2 \texttt{ERRTHETA}}{\texttt{ERRB}^2} = \frac{\mathrm{var}(\overline{y})}{\mathrm{var}(\overline{x})\mathrm{var}(\overline{y}) - \mathrm{cov}^2(\overline{x}, \overline{y})}, \tag{1.35}$$

$$\texttt{ERRCYY} \quad = \quad \frac{\sin^2 \texttt{ERRTHETA}}{\texttt{ERRA}^2} + \frac{\cos^2 \texttt{ERRTHETA}}{\texttt{ERRB}^2} = \frac{\mathrm{var}(\overline{x})}{\mathrm{var}(\overline{x})\mathrm{var}(\overline{y}) - \mathrm{cov}^2(\overline{x}, \overline{y})}, \tag{1.36}$$

$$\texttt{ERRCXY} \quad = \quad 2 \cos \texttt{ERRTHETA} \sin \texttt{ERRTHETA} \left(\frac{1}{\texttt{ERRA}^2} - \frac{1}{\texttt{ERRB}^2}\right) \tag{1.37}$$

$$= \quad -2 \frac{\mathrm{cov}(\overline{x}, \overline{y})}{\mathrm{var}(\overline{x})\mathrm{var}(\overline{y}) - \mathrm{cov}^2(\overline{x}, \overline{y})}. \tag{1.38}$$

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** is able to handle, of course), some detections with very specific shapes may yield 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 strongly 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

$$\overline{x^2}\,\overline{y^2} - \overline{xy}^2 < \rho^2, \tag{1.39}$$

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 (1.39) is true. If this is the case, a new test is conducted:

$$\mathrm{var}(\overline{x})\,\mathrm{var}(\overline{y}) - \mathrm{covar}^2(\overline{x},\overline{y}) < \rho_e^2, \tag{1.40}$$

where ρ_e is arbitrarily set to $\left(\sum_{i\in\mathcal{S}} \sigma_i^2\right) / \left(\sum_{i\in\mathcal{S}} p_i\right)^2$. If (1.40) is true, then $\overline{x^2}$ and $\overline{y^2}$ are incremented by ρ_e.

Isophotal areas: ISOAREA, ISOAREAF

An isophotal area is defined as the number of pixels with values exceeding some threshold above the background. Isophotal areas played an important role in the 80's and the 90's by providing, at a small computing cost, morphometric quantities complementary to second-order moments. **SExtractor** computes two isophotal areas inside the detection footprint:

- ISOAREAF applies to the (filtered) detection image, above the detection threshold.
- ISOAREA applies to the measurement image, with the additional constraint that the background-subtracted value of measurement image pixels must exceed the threshold set with the `ANALYSIS_THRESH` configuration parameter.

Photometry

Important: Unless otherwise noted, the pixel values used for computing "isophotal" fluxes and surface brightnesses are taken from the background-subtracted measurement image.

Isophotal flux: FLUX_ISO

FLUX_ISO is computed simply by integrating the background-subracted pixels values p_i from the *measurement* image within the detection footprint

$$\texttt{FLUX\_ISO} = F_{\mathrm{iso}} = \sum_{i\in\mathcal{S}} p_i. \tag{1.41}$$

SExtractor also provides an estimation of the uncertainty FLUXERR_ISO, a magnitude MAG_ISO and a magnitude error estimate MAGERR_ISO: see *Fluxes and magnitudes*.

Corrected isophotal magnitude: MAG_ISOCOR

Note: Corrected isophotal magnitudes are now deprecated; they remain in **SExtractor** v2.x for compatibility with **SExtractor** v1.

MAG_ISOCOR magnitudes are a quick-and-dirty way of retrieving the fraction of flux lost by isophotal magnitudes.

If one makes the assumption that the intensity profiles of faint objects recorded in the frame are roughly Gaussian because of atmospheric blurring, then the fraction $\eta = \frac{F_{\mathrm{iso}}}{F_{\mathrm{tot}}}$ of the total flux enclosed within a particular isophote reads [18]:

$$\left(1 - \frac{1}{\eta}\right)\ln(1-\eta) = \frac{A\,t}{F_{\mathrm{iso}}}, \tag{1.42}$$

where A is the area and t the threshold related to this isophote. :eq:isocor 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:

$$\eta \approx 1 - 0.1961 \frac{A\,t}{F_{\rm iso}} - 0.7512 \left(\frac{A\,t}{F_{\rm iso}} \right)^2 . \tag{1.43}$$

A "total" magnitude MAG_ISOCOR estimate is then

$$\texttt{MAG\_ISOCOR} = \texttt{MAG\_ISO} + 2.5 \log_{10} \eta. \tag{1.44}$$

Clearly the MAG_ISOCOR correction works best with stars; and although it gives reasonably accurate results with most disk galaxies, it breaks down for ellipticals because of the broader wings in the profiles.

Peak value: FLUX_MAX

FLUX_MAX is the peak pixel value (above the local background) in the measurement image. Note that it may not correspond to the pixel with coordinates given by XPEAK and YPEAK (see *Position of the peak*) if `FILTER` is set to `Y` or if the measurement image differs from the detection image.

CLASS_STAR classifier

Note: The CLASS_STAR classifier has been superseded by the SPREAD_MODEL estimator (see *Model-based star-galaxy separation: SPREAD_MODEL*), which offers better performance by making explicit use of the full, variable PSF (Point Spread Function) model.

A good discrimination between stars and galaxies is essential for both galactic and extragalactic statistical studies. The common assumption is that galaxy images look more extended or fuzzier than those of stars (or QSO (Quasi-Stellar Object)s). **SExtractor** provides the CLASS_STAR catalog parameter for separating both types of sources. The CLASS_STAR classifier relies on a multilayer feed-forward neural network trained using supervised learning to estimate the *a posteriori* probability [19, 20] of a **SExtractor** detection to be a point source or an extended object. Below is a shortened description of the estimator, see [12] for more details.

Inputs and outputs

The neural network is a multilayer Perceptron with a single fully connected, hidden layers. Of all neural networks it is probably the best-studied, and it has been intensively applied with success for many classification tasks.

The classifier (Fig. 1.4) has 10 inputs:

- 8 isophotal areas $A_0..A_7$, measured at isophotes exponentially spaced between the analysis threshold (which may be modified with the `ANALYSIS_THRESH` configuration parameter) and the object's peak pixel value
- The object's peak pixel value above the local background I_{max}
- A *seeing* input, which acts as a tuning button.

The output layer contains only one neuron, as "star" and "galaxy" are two classes mutually exclusive. The output value is a "stellarity index", which for images that reasonably match those of the training sample is an estimation of the *a posteriori* probability for the classified object to be a point-source. Hence a CLASS_STAR close to 0 means that the object is very likely a galaxy, and 1 that it is a star. In practice, real data always differ at least slightly from the training sample, and the CLASS_STAR output is often a poor approximation of the expected *a posteriori* probabilities. Nevertheless, a CLASS_STAR value closer to 0 or 1 normally indicates a higher confidence in the classification, and the balance between sample completeness and purity may still be adjusted by tweaking the decision threshold .

The *seeing* input must be set by the user with the `SEEING_FWHM` configuration parameter. If `SEEING_FWHM` is set to 0, it is automatically measured on the PSF model which must be provided (using the `PSF_NAME` configuration parameter).

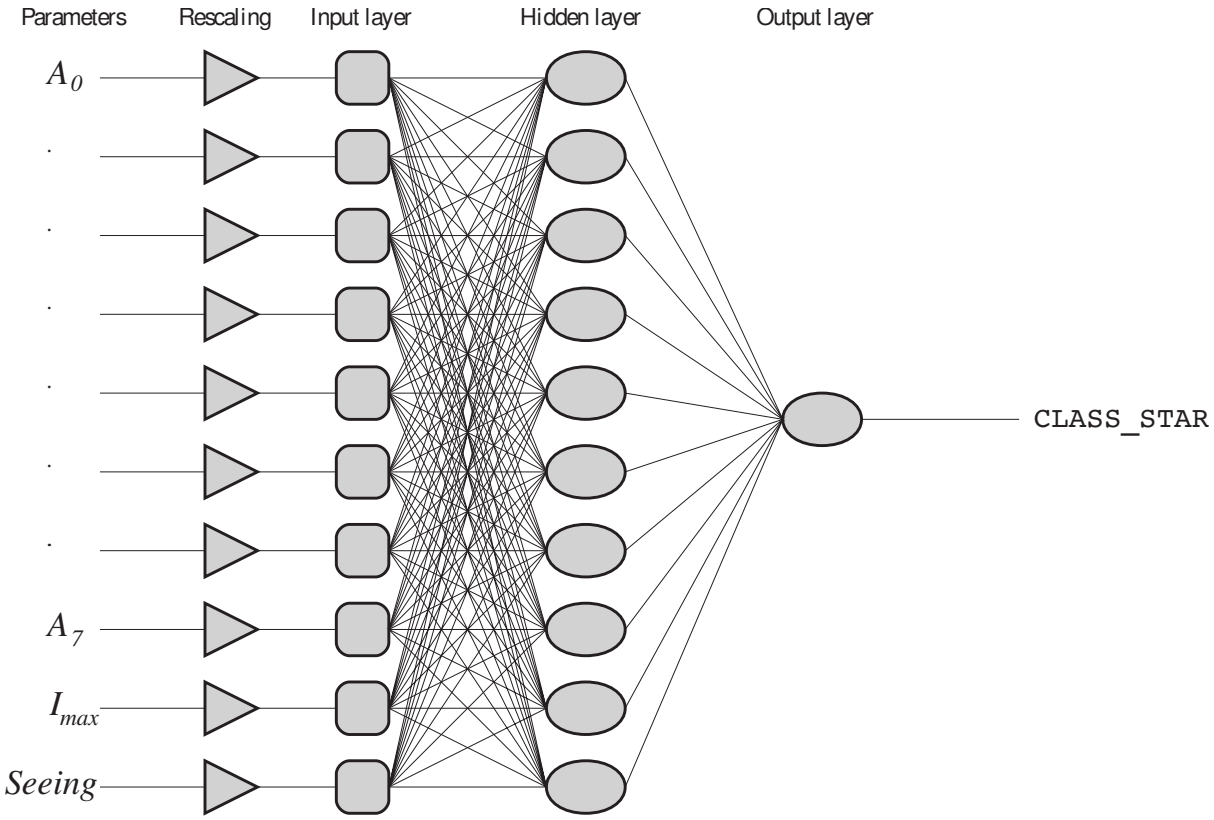


Fig. 1.4: Architecture of **SExtractor**'s CLASS_STAR classifier

If no PSF model is available, the `SEEING_FWHM` configuration parameter must be adjusted by the user to match the actual average PSF FWHM (Full Width at Half Maximum) on the image. The accuracy with which `SEEING_FWHM` must be set for optimal results ranges from $\pm 20\%$ for bright sources to about $\pm 5\%$ for the faintest (Fig. 1.5). `SEEING_FWHM` is expressed in arcseconds. The `PIXEL_SCALE` configuration parameter must therefore also be set by the user if WCS information is missing from the FITS image header. There are several ways to measure, directly or indirectly, the size of point sources in **SExtractor**; they may lead to slightly discordant results, depending on the exact shape of the PSF. The measurement FWHM_IMAGE (although not the most reliable as an image quality estimator) sets the reference when it comes to setting `SEEING_FWHM`.

One may check that the `SEEING_FWHM` is set correctly by making sure that the typical CLASS_STAR value of unclassifiable sources at the faint end of the catalog hovers around the 0.5 mark.

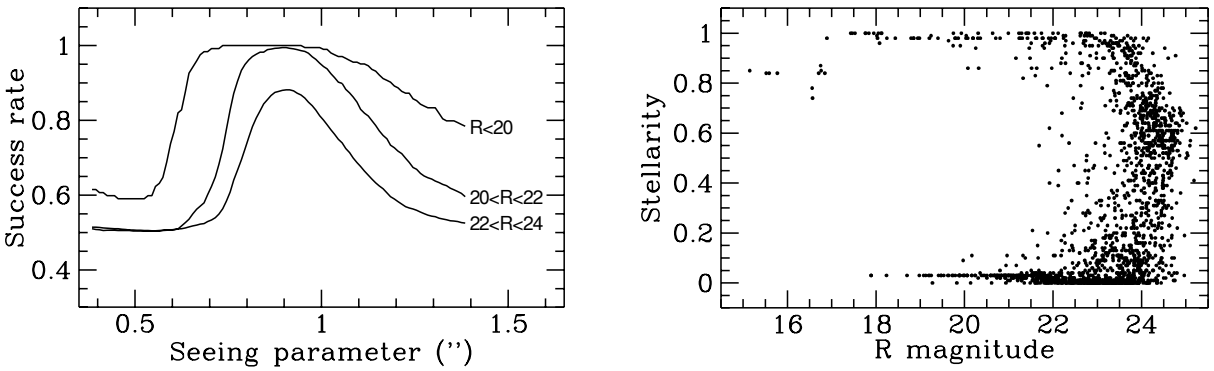


Fig. 1.5: Architecture of **SExtractor**'s CLASS_STAR classifier

Training

This section gives some insight on how the CLASS_STAR classifier has been trained. The main issue with supervised machine learning is the labeling of the large training sample. Hopefully a big percentage of contemporary astronomical images share a common set of generic features, which can be simulated with sufficient realism to create a large training sample together with the ground truth (labels). The CLASS_STAR classifier was trained on such a sample of artificial images.

Six hundred 512×512 simulation images containing stars and galaxies were generated to train the network using an early prototype of the **SkyMaker** package [21]. They were done in the blue band, where galaxies present very diversified aspects. The two parameters governing the shape of the PSF (*seeing* FWHM and Moffat β parameter [22]) were chosen randomly with $0.025 \leq$ FWHM ≤ 5.5 and $2 \leq \beta \leq 4$. Note that the Moffat function used in the simulation is a poor approximation to diffraction-limited images, hence the CLASS_STAR classifier is not optimized for space-based images. The pixel scale was always taken less than ≈ 0.7 FWHM to warrant correct sampling of the image. Bright galaxies are simply too rare in the sky to constitute a significant training sample on such a small number of simulations. So, keeping a constant comoving number density, we increased artificially the number of nearby galaxies by making the volume element proportional to zdz. Stars were given a number-magnitude distribution identical to that of galaxies. **Therefore any pattern presented to the network had a 50% chance to correspond to a star or a galaxy, irrespective of magnitude**[2]. Crowding in the simulated images was higher than what one sees on real images of high galactic latitude fields, allowing for the presence of many "difficult cases" (close double stars, truncated profiles, etc...) that the neural network classifier had to deal with.

SExtractor was run on each image with 8 different extraction thresholds. This led to a catalog with about 10^6 entries with the 10 input parameters plus the class label. Back-propagation learning took about 15 min on a SUN SPARC20 workstation. The final set of synaptic weights was saved to the file `default.nnw` , ready to be used in "feed-forward" mode during source extraction.

1.6.2 Windowed positional parameters

Measurements performed through a *window* function (an *envelope*) do not have many of the drawbacks of isophotal measurements. **SExtractor** implements "windowed" versions for most of the measurements described in the *previous section*:

Note: 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 coordinates or _WORLD, _SKY, _J2000 or _B1950 for WCS coordinates (see *Positions and shapes*).

Isophotal parameters	Equivalent windowed parameters
X, Y	XWIN, YWIN
ERRA, ERRB, ERRTHETA	ERRAWIN, ERRBWIN, ERRTHETAWIN
A, B, THETA	AWIN, BWIN, THETAWIN
X2, Y2, XY	X2WIN, Y2WIN, XYWIN
CXX, CYY, CXY	CXXWIN, CYYWIN, CXYWIN

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; the Gaussian FWHM is the diameter of the disk that contains half of the object flux (d_{50}). Note that in double-image mode (§[chap:using]) the window is scaled based on the *measurement* image.

[2] Faint galaxies have less chance being detected than faint stars, but it has little effect because the ones that are lost at a given magnitude are predominantly the most extended and consequently the easiest to classify.

Windowed centroid: XWIN, YWIN

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

$$\begin{aligned}
\mathtt{XWIN}^{(t+1)} = \quad \overline{x_{\mathtt{WIN}}}^{(t+1)} &= \overline{x_{\mathtt{WIN}}}^{(t)} + 2\,\frac{\sum_{r_i^{(t)}<r_{\max}} w_i^{(t)} I_i\,(x_i - \overline{x_{\mathtt{WIN}}}^{(t)})}{\sum_{r_i^{(t)}<r_{\max}} w_i^{(t)} I_i}, \\
\mathtt{YWIN}^{(t+1)} = \quad \overline{y_{\mathtt{WIN}}}^{(t+1)} &= \overline{y_{\mathtt{WIN}}}^{(t)} + 2\,\frac{\sum_{r_i^{(t)}<r_{\max}} w_i^{(t)} I_i\,(y_i - \overline{y_{\mathtt{WIN}}}^{(t)})}{\sum_{r_i^{(t)}<r_{\max}} w_i^{(t)} I_i},
\end{aligned} \tag{1.45}$$

where

$$w_i^{(t)} = \exp\left(-\frac{{r_i^{(t)}}^2}{2s_{\mathtt{WIN}}^2}\right), \tag{1.46}$$

with

$$r_i^{(t)} = \sqrt{\left(x_i - \overline{x_{\mathtt{WIN}}}^{(t)}\right)^2 + \left(y_i - \overline{y_{\mathtt{WIN}}}^{(t)}\right)^2} \tag{1.47}$$

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

Although they are slower, it is recommended to use whenever possible windowed position parameters instead of their isophotal equivalents; the measurements they provide are generally much more accurate (Fig. 1.6). The centroiding accuracy of XWIN_IMAGE and YWIN_IMAGE is actually very close to that of PSF-fitting on focused and properly sampled star images. Windowed measurements can also be applied to galaxies. It has been verified that for isolated objects with Gaussian-like profiles, their accuracy is close to the theoretical limit set by image noise[1].

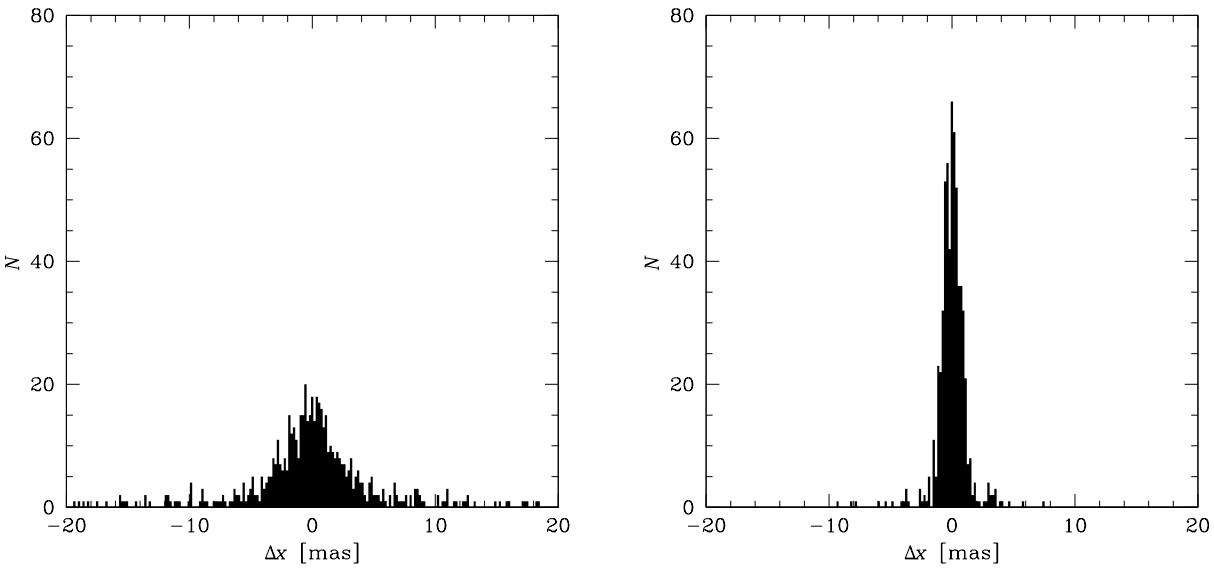


Fig. 1.6: Comparison between isophotal and windowed centroid measurement residuals on simulated, background noise-limited images. *Left*: histogram of the difference between X_IMAGE and the true centroid in x. *Right*: histogram of the difference between XWIN_IMAGE and the true centroid in x.

[1] See http://www.astromatic.net/forum/showthread.php?tid=581 .

Windowed 2nd order moments: X2, Y2, XY

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

$$\begin{aligned}
\texttt{X2WIN} = \overline{x^2_{\texttt{WIN}}} &= \frac{\sum_{r_i<r_{\max}} w_i I_i (x_i - \overline{x_{\texttt{WIN}}})^2}{\sum_{r_i<r_{\max}} w_i I_i}, \\
\texttt{Y2WIN} = \overline{y^2_{\texttt{WIN}}} &= \frac{\sum_{r_i<r_{\max}} w_i I_i (y_i - \overline{y_{\texttt{WIN}}})^2}{\sum_{r_i<r_{\max}} w_i I_i}, \\
\texttt{XYWIN} = \overline{xy_{\texttt{WIN}}} &= \frac{\sum_{r_i<r_{\max}} w_i I_i (x_i - \overline{x_{\texttt{WIN}}})(y_i - \overline{y_{\texttt{WIN}}})}{\sum_{r_i<r_{\max}} w_i I_i}.
\end{aligned} \tag{1.48}$$

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

Windowed shape parameters: AWIN, BWIN, THETAWIN

They are computed from the windowed 2nd order moments exactly the same way as in (1.18) and (1.6.1) from the *isophotal shape parameter* section:

$$\begin{aligned}
\texttt{AWIN}^2 &= \frac{\overline{x^2_{\texttt{WIN}}} + \overline{y^2_{\texttt{WIN}}}}{2} + \sqrt{\left(\frac{\overline{x^2_{\texttt{WIN}}} - \overline{y^2_{\texttt{WIN}}}}{2}\right)^2 + \overline{xy_{\texttt{WIN}}}^2}, \\
\texttt{BWIN}^2 &= \frac{\overline{x^2_{\texttt{WIN}}} + \overline{y^2_{\texttt{WIN}}}}{2} - \sqrt{\left(\frac{\overline{x^2_{\texttt{WIN}}} - \overline{y^2_{\texttt{WIN}}}}{2}\right)^2 + \overline{xy_{\texttt{WIN}}}^2}, \\
\tan(2\,\texttt{THETAWIN}) &= 2\frac{\overline{xy_{\texttt{WIN}}}}{\overline{x^2_{\texttt{WIN}}} - \overline{y^2_{\texttt{WIN}}}}.
\end{aligned} \tag{1.49}$$

Windowed ellipse parameters: CXXWIN, CYYWIN, CXYWIN

Ellipse parameters are computed from the windowed 2nd order moments exactly the same way as in (1.6.1) describing the *isophotal ellipse parameters*:

$$\begin{aligned}
\texttt{CXXWIN} &= \frac{\cos^2\texttt{THETAWIN}}{\texttt{AWIN}^2} + \frac{\sin^2\texttt{THETAWIN}}{\texttt{BWIN}^2} = \frac{\overline{y^2_{\texttt{WIN}}}}{\overline{x^2_{\texttt{WIN}}}\,\overline{y^2_{\texttt{WIN}}} - \overline{xy_{\texttt{WIN}}}^2} \\
\texttt{CYYWIN} &= \frac{\sin^2\texttt{THETAWIN}}{\texttt{AWIN}^2} + \frac{\cos^2\texttt{THETAWIN}}{\texttt{BWIN}^2} = \frac{\overline{x^2_{\texttt{WIN}}}}{\overline{x^2_{\texttt{WIN}}}\,\overline{y^2_{\texttt{WIN}}} - \overline{xy_{\texttt{WIN}}}^2} \\
\texttt{CXYWIN} &= 2\cos\texttt{THETAWIN}\sin\texttt{THETAWIN}\left(\frac{1}{\texttt{AWIN}^2} - \frac{1}{\texttt{BWIN}^2}\right) = -2\frac{\overline{xy_{\texttt{WIN}}}}{\overline{x^2_{\texttt{WIN}}}\,\overline{y^2_{\texttt{WIN}}} - \overline{xy_{\texttt{WIN}}}^2}.
\end{aligned} \tag{1.50}$$

Windowed position uncertainties: ERRX2WIN, ERRY2WIN, ERRXYWIN, ERRAWIN, ERRBWIN, ERRTHETAWIN, ERRCXXWIN, ERRCYYWIN, ERRCXYWIN

Windowed position uncertainties are computed on the image data once the centering process of the *windowed centroid* has converged. Assuming that noise is uncorrelated among pixels, standard error propagation applied to (1.45) writes:

$$\begin{aligned}
\texttt{ERRX2WIN} = \mathrm{var}(\overline{x_{\texttt{WIN}}}) &= 4\,\frac{\sum_{r_i<r_{\max}} w_i^2\sigma_i^2 (x_i - \overline{x_{\texttt{WIN}}})^2}{\left(\sum_{r_i<r_{\max}} w_i I_i\right)^2}, \\
\texttt{ERRY2WIN} = \mathrm{var}(\overline{y_{\texttt{WIN}}}) &= 4\,\frac{\sum_{r_i<r_{\max}} w_i^2\sigma_i^2 (y_i - \overline{y_{\texttt{WIN}}})^2}{\left(\sum_{r_i<r_{\max}} w_i I_i\right)^2}, \\
\texttt{ERRXYWIN} = \mathrm{cov}(\overline{x_{\texttt{WIN}}}, \overline{y_{\texttt{WIN}}}) &= 4\,\frac{\sum_{r_i<r_{\max}} w_i^2\sigma_i^2 (x_i - \overline{x_{\texttt{WIN}}})(y_i - \overline{y_{\texttt{WIN}}})}{\left(\sum_{r_i<r_{\max}} w_i I_i\right)^2}.
\end{aligned} \tag{1.51}$$

Semi-major axis ERRAWIN, semi-minor axis ERRBWIN, and position angle ERRTHETAWIN of the 1σ position error ellipse are computed from the covariance matrix elements $\mathrm{var}(\overline{x_{\mathtt{WIN}}})$, $\mathrm{var}(\overline{y_{\mathtt{WIN}}})$, $\mathrm{cov}(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})$, similarly to the *isophotal error ellipse*:

$$\begin{aligned}
\mathtt{ERRAWIN}^2 &= \frac{\mathrm{var}(\overline{x_{\mathtt{WIN}}}) + \mathrm{var}(\overline{y_{\mathtt{WIN}}})}{2} + \sqrt{\left(\frac{\mathrm{var}(\overline{x_{\mathtt{WIN}}}) - \mathrm{var}(\overline{y_{\mathtt{WIN}}})}{2}\right)^2 + \mathrm{cov}^2(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})}, \\
\mathtt{ERRBWIN}^2 &= \frac{\mathrm{var}(\overline{x_{\mathtt{WIN}}}) + \mathrm{var}(\overline{y_{\mathtt{WIN}}})}{2} - \sqrt{\left(\frac{\mathrm{var}(\overline{x_{\mathtt{WIN}}}) - \mathrm{var}(\overline{y_{\mathtt{WIN}}})}{2}\right)^2 + \mathrm{cov}^2(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})}, \\
\tan(2\mathtt{ERRTHETAWIN}) &= 2\,\frac{\mathrm{cov}(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})}{\mathrm{var}(\overline{x_{\mathtt{WIN}}}) - \mathrm{var}(\overline{y_{\mathtt{WIN}}})}.
\end{aligned} \tag{1.52}$$

The error ellipse parameters are:

$$\begin{aligned}
\mathtt{ERRCXXWIN} &= \frac{\cos^2 \mathtt{ERRTHETAWIN}}{\mathtt{ERRAWIN}^2} + \frac{\sin^2 \mathtt{ERRTHETAWIN}}{\mathtt{ERRBWIN}^2} = \frac{\mathrm{var}(\overline{y_{\mathtt{WIN}}})}{\mathrm{var}(\overline{x_{\mathtt{WIN}}})\mathrm{var}(\overline{y_{\mathtt{WIN}}}) - \mathrm{cov}^2(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})}, \\
\mathtt{ERRCYYWIN} &= \frac{\sin^2 \mathtt{ERRTHETAWIN}}{\mathtt{ERRAWIN}^2} + \frac{\cos^2 \mathtt{ERRTHETAWIN}}{\mathtt{ERRBWIN}^2} = \frac{\mathrm{var}(\overline{x_{\mathtt{WIN}}})}{\mathrm{var}(\overline{x_{\mathtt{WIN}}})\mathrm{var}(\overline{y_{\mathtt{WIN}}}) - \mathrm{cov}^2(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})}, \\
\mathtt{ERRCXYWIN} &= 2\cos \mathtt{ERRTHETAWIN} \sin \mathtt{ERRTHETAWIN} \left(\frac{1}{\mathtt{ERRAWIN}^2} - \frac{1}{\mathtt{ERRBWIN}^2}\right) \\
&= -2\frac{\mathrm{cov}(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})}{\mathrm{var}(\overline{x_{\mathtt{WIN}}})\mathrm{var}(\overline{y_{\mathtt{WIN}}}) - \mathrm{cov}^2(\overline{x_{\mathtt{WIN}}}, \overline{y_{\mathtt{WIN}}})}.
\end{aligned} \tag{1.53}$$

Windowed measurement flags: FLAGS_WIN

The FLAGS_WIN catalog parameter flags various issues which may happen with windowed measurements (see the *Flagging* section for details on how flags are managed in **SExtractor**):

Table 1.4: FLAGS_WIN description

Vali	Meaning
1	*Windowed second-order moments* are inconsistent ($\overline{x^2_{\mathtt{WIN}} y^2_{\mathtt{WIN}}} \leq \overline{x_{\mathtt{WIN}} y_{\mathtt{WIN}}}^2$)
2	*Windowed second-order moments* are less than or equal to 0
4	Windowed flux is less than or equal to 0

1.6.3 Aperture photometry

Besides *isophotal*, PSF and *model-fitting* flux estimates, **SExtractor** can currently perform two types of flux measurements: *fixed-aperture* and *adaptive-aperture*. For every FLUX_ measurement, an error estimate FLUXERR_, a magnitude MAG_ and a magnitude error estimate MAGERR_ are also available: see *Fluxes and magnitudes*.

An estimate of the error is available for each type of flux. For aperture fluxes, the flux uncertainty is computed using

$$\mathtt{FLUXERR} = \sqrt{\sum_{i \in \mathcal{A}} (\sigma_i^2 + \frac{p_i}{g_i})} \tag{1.54}$$

where $\mathcal{A}$ is the set of pixels defining the photometric aperture, and σ_i, p_i, g_i respectively the standard deviation of noise (in ADU) estimated from the local background, p_i the measurement image pixel value subtracted from the background, and g_i the effective detector gain in e^-/ADU at pixel i. Note that this error estimate provides a lower limit of the true uncertainty, as it only takes into account photon and detector noise.

Fixed-aperture flux: FLUX_APER

FLUX_APER estimates the flux from the measurement image above the background inside a circular aperture. The diameter of the aperture in pixels is defined by the PHOTOM_APERTURES configuration parameter. It does not have to be an integer: each "regular" pixel is subdivided in 5×5 sub-pixels before measuring the flux within the aperture. If FLUX_APER is provided as a vector FLUX_APER[n], at least n apertures must be specified with the PHOTOM_APERTURES configuration parameter.

Automatic aperture flux: FLUX_AUTO

FLUX_AUTO provides an estimate of the "total flux" by integrating pixel values within an adaptively scaled aperture. **SExtractor**'s automatic aperture photometry routine derives from Kron's "first moment" algorithm [23]:

1. An elliptical aperture is *defined by the second order moments of the object's light distribution*, with semi-major axis $a =$ A_IMAGE, semi-minor axis $b =$ B_IMAGE, and position angle THETA_IMAGE.
2. The ellipse's major and minor axes are multiplied by 6 (which corresponds roughly to twice the size of the isophotal footprint on each axis).
3. Inside this elliptical aperture $\mathcal{E}$, an analog of Kron's "first moment" is computed:

$$r_{\mathrm{Kron}} = \frac{\sum_{i\in\mathcal{E}} r_i\, p_i^{(d)}}{\sum_{i\in\mathcal{E}} p_i^{(d)}}, \tag{1.55}$$

where $p_i^{(d)}$ is the pixel value *in the detection image*. r_i is what we shall call the "reduced pseudo-radius" at pixel i

$$r_i \equiv \sqrt{\mathtt{CXX\_IMAGE} \times \Delta x_i^2 + \mathtt{CYY\_IMAGE} \times \Delta y_i^2 + \mathtt{CXY\_IMAGE} \times \Delta x_i \Delta y_i}, \tag{1.56}$$

where Δx_i and Δy_i are the pixel coordinates relative to the detection centroid:

$$\Delta x_i = x_i - \mathtt{X\_IMAGE}$$
$$\Delta y_i = y_i - \mathtt{Y\_IMAGE}.$$

[23] and [15] have shown that for stars and galaxy profiles convolved with Gaussian seeing, $\geq 90\%$ of the flux is expected to lie inside a circular aperture of radius kr_{Kron} with $k = 2$, almost independently of the magnitude. Experiments have shown [12] that this conclusion remains unchanged if one replaces the circular aperture with the "Kron elliptical aperture" $\mathcal{K}$ with reduced pseudo-radius kr_{Kron}.

FLUX_AUTO is the sum of pixel values from the measurement image, subtracted from the local background, inside the Kron ellipse:

$$\mathtt{FLUX\_AUTO} = \sum_{i\in\mathcal{K}} p_i. \tag{1.57}$$

The quantity kr_{Kron}, known as the *Kron radius* (which in **SExtractor** is actually a "reduced pseudo-radius") is provided by the KRON_RADIUS. $k = 2$ 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%, at the expense of SNR (Signal-to-Noise Ratio) in the measurement. Very noisy objects may sometimes end up with a Kron ellipse being too small, even smaller that the isophotal footprint of the object itself. For this reason, **SExtractor** imposes a minimum size for the Kron radius, which cannot be less than $r_{\mathrm{Kron,min}}$. The user has full control over the parameters k and $r_{\mathrm{Kron,min}}$ through the PHOT_AUTOPARAMS configuration parameters. PHOT_AUTOPARAMS is set by default to `2.5,3.5`.

Hint: Aperture magnitudes are sensitive to crowding. In **SExtractor** v1, 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 neighbors (2 isophotal radii for instance). This was done automatically when using the MAG_BEST magnitude: MAG_BEST=MAG_AUTO when it is sure that no neighbor can bias MAG_AUTO by more than 10%, and 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 neighbors that have been detected around the measured source (`MASK_TYPE BLANK` option). The second improvement is an automatic correction of parts of the aperture that are suspected to be contaminated by a neighbor. This is done by mirroring the opposite, cleaner side of the measurement ellipse if available (`MASK_TYPE CORRECT` option, which is also the default).

Petrosian aperture flux: FLUX_PETRO

Similar to FLUX_AUTO, FLUX_PETRO provides an estimate of the "total flux" by integrating pixel values within an adaptively scaled elliptical aperture. FLUX_PETRO's algorithm derives from Petrosian's photometric estimator [24, 25, 26]:

1. An elliptical aperture is *defined by the second order moments of the object's light distribution*, with semi-major axis $a =$ `A_IMAGE`, semi-minor axis $b =$ `B_IMAGE`, and position angle THETA_IMAGE.
2. The ellipse's major and minor axes are multiplied by 6 (which corresponds roughly to twice the size of the isophotal footprint on each axis).
3. Within this elliptical aperture $\mathcal{E}$, the *Petrosian ratio* $R_\mathrm{P}(r)$ is computed:

$$R_\mathrm{P}(r) = \frac{\sum_{0.9r<r_i<1.1r} p_i^{(d)}}{\sum_{r_i<r} p_i^{(d)}} \frac{N_{r_i<r}}{N_{0.9r<r_i<1.1r}}, \tag{1.58}$$

where $p_i^{(d)}$ is the pixel value *in the detection image*. r_i is the "reduced pseudo-radius" at pixel i as defined in (1.56). The *Petrosian ellipse* $\mathcal{P}$ is the ellipse with reduced pseudo-radius $N_\mathrm{P} r_\mathrm{P}$, where r_P is defined by

$$R_\mathrm{P}(r_\mathrm{p}) \equiv 0.2 \tag{1.59}$$

The quantity $N_\mathrm{P} r_\mathrm{P}$ is called *Petrosian radius* in **SExtractor**[1] and is provided by the PETRO_RADIUS catalog parameter. The Petrosian factor N_P is set to 2.0 by default. Very noisy objects may sometimes end up with a Petrosian ellipse being too small. For this reason, **SExtractor** imposes a minimum size for the Petrosian radius, which cannot be less than $r_{\mathrm{P},\min}$. The user has full control over the parameters N_P and $r_{\mathrm{P},\min}$ through the `PHOT_PETROPARAMS` configuration parameters. `PHOT_PETROPARAMS` is set by default to `2.0,3.5`.

The Petrosian flux is the sum of pixel values from the measurement image, subtracted from the local background, inside the Petrosian ellipse:

$$\texttt{FLUX\_PETRO} = \sum_{i\in\mathcal{P}} p_i. \tag{1.60}$$

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

$$I = I_0\, 10^{D/\gamma}, \tag{1.61}$$

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:

$$I_0 = \frac{\gamma}{\ln 10}\, 10^{-0.4\, m_0}. \tag{1.62}$$

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 [27] for details).

[1] Some authors prefer to define the Petrosian radius as r_P instead of $N_\mathrm{P} r_\mathrm{P}$.

Local background

Almost all **SExtractor** measurements are done using background-subtracted pixel values. In crowded fields, or in images where the background is irregular, the *background model* may be significantly inaccurate, locally creating biases in photometric estimates.

The user has the possibility to force **SExtractor** to correct, for every detection, the background used to compute fluxes by setting the BACKPHOTO_TYPE configuration parameter to LOCAL (GLOBAL is the default). In LOCAL mode, a mean background residual level is estimated from background-subtracted pixel values within a "rectangular annulus" around the isophotal limits of the object. The user can specify the thickness of the annulus, in pixels, with the BACKPHOTO_SIZE configuration parameter. The default thickness is 24 pixels. The residual background level computed in LOCAL mode is used by **SExtractor** to correct all aperture photometry measurements, as well as basic surface brightness estimations such as FLUX_MAX. However in practice the BACKPHOTO_TYPE LOCAL option has not proven as being really beneficial to photometric accuracy, and it is generally advised to leave BACKPHOTO_TYPE set to GLOBAL.

In both LOCAL and GLOBAL modes, the BACKGROUND catalog parameter gives the value of the background estimated at the centroid of the object.

1.6.4 Model fitting

Fitting procedure

SExtractor can fit models to the images of detected objects since version 2.8. The fit is performed by minimizing the loss function

$$\lambda(\boldsymbol{q}) = \sum_i \left(g\left(\frac{p_i - \tilde{m}_i(\boldsymbol{q})}{\sigma_i} \right) \right)^2 + \sum_j \left(\frac{f_j(Q_j) - \mu_j}{s_j} \right)^2 \tag{1.63}$$

with respect to components of the model parameter vector $\boldsymbol{q}$. $\boldsymbol{q}$ comprises parameters describing the shape of the model and the model pixel coordinates $\boldsymbol{x}$.

Modified least squares

The first term in (1.63) is a modified weighted sum of squares that aims at minimizing the residuals of the fit. p_i, $\tilde{m}_i(\boldsymbol{q})$ and σ_i are respectively the pixel value above the background, the value of the resampled model, and the pixel value uncertainty at image pixel i. $g(u)$ is a derivable monotonous function that reduces the influence of large deviations from the model, such as the contamination by neighbors (Fig. 1.7):

$$g(u) = \begin{cases} u_0 \log\left(1 + \frac{u}{u_0}\right) & \text{if } u \geq 0, \\ -u_0 \log\left(1 - \frac{u}{u_0}\right) & \text{otherwise.} \end{cases} \tag{1.64}$$

u_0 sets the level below which $g(u) \approx u$. In practice, choosing $u_0 = \kappa\sigma_i$ with $\kappa = 10$ makes the first term in (1.63) behave like a traditional weighted sum of squares for residuals close to the noise level.

Caution: The cost function in (1.63) is optimized for Gaussian noise and makes model-fitting in **SExtractor** appropriate only for image noise with a PDF symmetrical around the mean.

The vector $\tilde{\boldsymbol{m}}(\boldsymbol{q})$ is obtained by convolving the high resolution model $\boldsymbol{m}(\boldsymbol{q})$ with the local PSF model $\boldsymbol{\phi}$ and applying a resampling operator $\mathbf{R}(\boldsymbol{x})$ to generate the final model raster at position $\boldsymbol{x}$ at the nominal image resolution:

$$\tilde{\boldsymbol{m}}(\boldsymbol{q}) = \mathbf{R}(\boldsymbol{x})(\boldsymbol{m}(\boldsymbol{q}) * \boldsymbol{\phi}). \tag{1.65}$$

$\mathbf{R}(\boldsymbol{x})$ depends on the pixel coordinates $\boldsymbol{x}$ of the model centroid:

$$\mathbf{R}_{ij}(\boldsymbol{x}) = h\left(\boldsymbol{x}_j - \eta.(\boldsymbol{x}_i - \boldsymbol{x})\right), \tag{1.66}$$

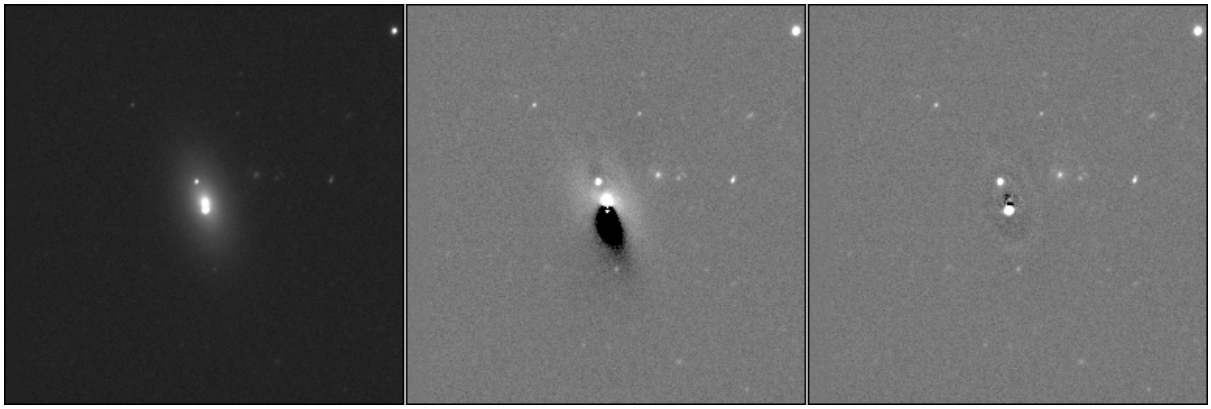


Fig. 1.7: Effect of the modified least squares loss function on fitting a model to a galaxy with a bright neighbor. *Left*: the original image; *Middle*: residuals of the model fitting with a regular least squares ($\kappa = +\infty$); *Right*: modified least squares with $\kappa = 10$.

where h is a 2-dimensional interpolant (interpolating function), $\boldsymbol{x}_i$ is the coordinate vector of image pixel i, $\boldsymbol{x}_j$ the coordinate vector of model sample j, and η is the image-to-model sampling step ratio (sampling factor) which is by default defined by the PSF model sampling. We adopt a Lánczos-4 function [28] as interpolant.

Change of variables

Many model parameters are valid only over a restricted domain. Fluxes, for instance, cannot be negative. In order to avoid invalid values and also to facilitate convergence, a change of variables is applied individually to each model parameter:

$$q_j = f_j(a_j, b_j, Q_j). \tag{1.67}$$

The "model" variable q_j is bounded by the lower limit a_j and the upper limit b_j by construction. The "engine" variable Q_j can take any value, and is actually the parameter that is being adjusted in the fit, although it does not have any physical meaning. In **SExtractor** three different types of transforms $f_j()$ are applied, depending on the parameter (Table 1.5).

Table 1.5: Types of changes of variables applied to model parameters

Type	Model $\xrightarrow{f^{-1}}$ Engine	Engine $\xrightarrow{f}$ Model	Examples
Unbounded (linear)	$Q_j = q_j$	$q_j = Q_j$	SPHEROID_POSANGLE DISK_POSANGLE
Bounded linear	$Q_j = \ln \frac{q_j - a_j}{b_j - q_j}$	$q_j = \frac{b_j - a_j}{1+\exp -Q_j} + a_j$	XMODEL_IMAGE SPHEROID_SERSICN
Bounded logarithmic	$Q_j = \ln \frac{\ln q_j - \ln a_j}{\ln b_j - \ln q_j}$	$q_j = a_j \frac{\ln b_j - \ln a_j}{1+\exp -Q_j}$	FLUX_SPHEROID DISK_ASPECT

In practice, this approach works well, and was found to be much more reliable than a box constrained algorithm [29].

Regularization

Although minimizing the (modified) weighted sum of least squares gives a solution that fits best the data, it does not necessarily correspond to the most probable solution given what we know about celestial objects. The discrepancy is particularly significant in very faint (SNR $\leq$ 20) and barely resolved galaxies, for which there is a tendency to overestimate the elongation, known as the "noise bias" in the weak-lensing community [30, 31, 32, 33]. To mitigate this issue, **SExtractor** implements a simple Tikhonov regularization scheme on selected engine parameters, in the form of an additional penalty term in (1.63). This term acts as a Gaussian prior on the selected *engine* parameters. However for the associated *model* parameters, the change of variables can make the (improper) prior far from Gaussian. Currently the only regularized parameter is SPHEROID_ASPECT_IMAGE (and its derivatives SPHEROID_ASPECT_WORLD, ELLIP1MODEL_IMAGE, etc.), for which $\mu_{\texttt{SPHEROID\_ASPECT}} = 0$ and $s_{\texttt{SPHEROID\_ASPECT}} = 1$ (Fig. 1.8).

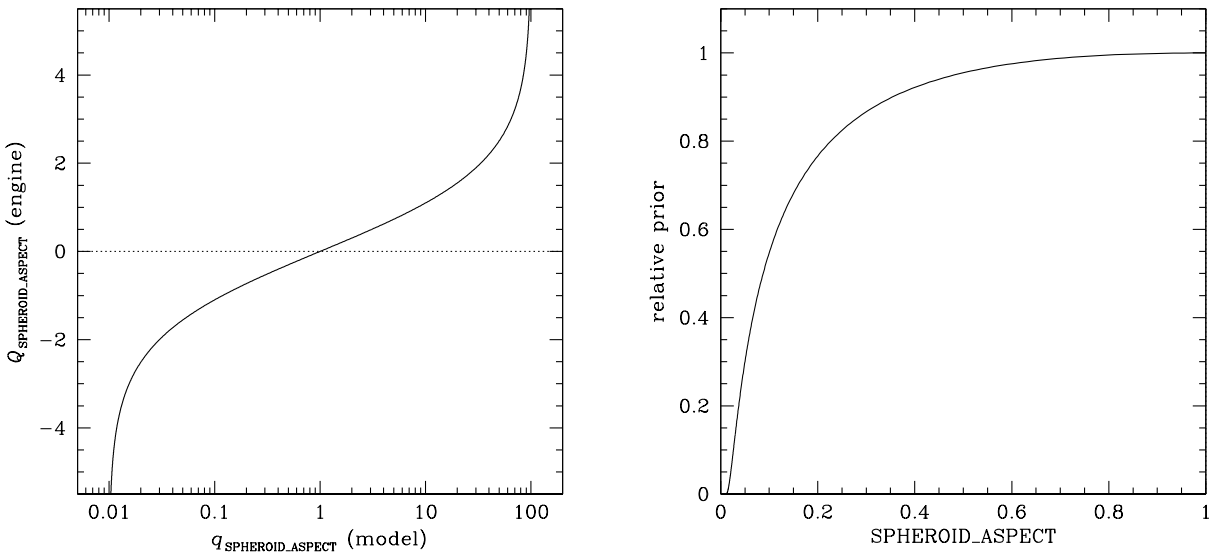


Fig. 1.8: Effect of the Gaussian prior on the SPHEROID_ASPECT_IMAGE model parameter. *Left:* change of variables between the model (in abscissa) and the engine (in ordinate) parameters. *Right*: equivalent (improper) prior applied to SPHEROID_ASPECT_IMAGE for $\mu_{\texttt{SPHEROID\_ASPECT}} = 0$ and $s_{\texttt{SPHEROID\_ASPECT}} = 1$ in equation (1.63).

Minimization

Minimization of the loss function $\lambda(\boldsymbol{q})$ is carried out using the Levenberg-Marquardt algorithm, and more specifically the **LevMar** implementation [34]. The library approximates the Jacobian matrix of the model from finite differences using Broyden's [35] rank one updates. The fit is done inside a disk which diameter is scaled to include the isophotal footprint of the object, plus the FWHM of the PSF, plus a 20 % margin.

The number of iterations is returned in the NITER_MODEL measurement parameter. It is generally a few tens. The final value of the modified chi square term in (1.63), divided by the number of degrees of freedom, is returned in CHI2_MODEL.

The FLAGS_MODEL catalog parameter flags various issues which may happen during the fitting process (see the *Flagging* section for details on how flags are managed in **SExtractor**):

Table 1.6: FLAGS_MODEL flag description

Valı	Meaning
1	the unconvolved, supersampled model raster exceeds 512×512 pixels and had to be resized
2	the convolved, resampled model raster exceeds 512×512 pixels and had to be resized
4	not enough pixels are available for model fitting on the measurement image (less pixels than fit parameters)
8	at least one of the fitted parameters hits the lower bound
16	at least one of the fitted parameters hits the upper bound

$1\,\sigma$ error estimates are provided for most measurement parameters; they are obtained from the full covariance matrix of the fit, which is itself computed by inverting the approximate Hessian matrix of $\lambda(\boldsymbol{q})$ at the solution.

Models

Models contain one or more components, which share their central coordinates. For instance, a galaxy model may be composed of a spheroid (bulge) and a disk components. Both components are concentric but they may have different scales, aspect ratios and position angles. Adding a component is done simply by invoking one of its measurement parameters in the parameter file, e.g., DISK_SCALE_IMAGE.

The present version of **SExtractor** supports the following models

- BACKOFFSET: flat background offset

  Relevant measurement parameters: FLUX_BACKOFFSET, FLUXERR_BACKOFFSET

$$m_{\texttt{BACKOFFSET}}(r) = m_0 \tag{1.68}$$

- POINT_SOURCE: point source

  Relevant measurement parameters: FLUX_POINTSOURCE, FLUXERR_POINTSOURCE, MAG_POINTSOURCE, MAGERR_POINTSOURCE, FLUXRATIO_POINTSOURCE, FLUXRATIOERR_POINTSOURCE

$$m_{\texttt{POINTSOURCE}}(r) = m_0\delta(r) \tag{1.69}$$

- DISK: exponential disk

  Relevant measurement parameters: FLUX_DISK, FLUXERR_DISK, MAG_DISK, MAGERR_DISK, FLUXRATIO_DISK, FLUXRATIOERR_DISK, FLUX_MAX_DISK, MU_MAX_DISK, FLUX_EFF_DISK, MU_EFF_DISK, FLUX_MEAN_DISK, MU_MEAN_DISK, DISK_SCALE_IMAGE, DISK_SCALEERR_IMAGE, DISK_SCALE_WORLD, DISK_SCALEERR_WORLD, DISK_ASPECT_IMAGE, DISK_ASPECTERR_IMAGE, DISK_ASPECT_WORLD, DISK_ASPECTERR_WORLD, DISK_INCLINATION, DISK_INCLINATIONERR, DISK_THETA_IMAGE, DISK_THETAERR_IMAGE, DISK_THETA_WORLD, DISK_THETAERR_WORLD, DISK_THETA_SKY, DISK_THETA_J2000, DISK_THETA_B1950

$$m_{\texttt{DISK}}(r) = m_0 \exp\left(-\frac{r}{h}\right) \tag{1.70}$$

- SPHEROID: Sérsic ($R^{1/n}$) spheroid

  FLUX_SPHEROID, FLUXERR_SPHEROID, MAG_SPHEROID, MAGERR_SPHEROID, FLUXRATIO_SPHEROID, FLUXRATIOERR_SPHEROID, FLUX_MAX_SPHEROID, MU_MAX_SPHEROID, FLUX_EFF_SPHEROID, MU_EFF_SPHEROID, FLUX_MEAN_SPHEROID, MU_MEAN_SPHEROID, SPHEROID_SCALE_IMAGE, SPHEROID_SCALEERR_IMAGE, SPHEROID_SCALE_WORLD, SPHEROID_SCALEERR_WORLD, SPHEROID_ASPECT_IMAGE, SPHEROID_ASPECTERR_IMAGE, SPHEROID_ASPECT_WORLD, SPHEROID_ASPECTERR_WORLD, SPHEROID_INCLINATION, SPHEROID_INCLINATIONERR, SPHEROID_THETA_IMAGE, SPHEROID_THETAERR_IMAGE, SPHEROID_THETA_WORLD, SPHEROID_THETAERR_WORLD, SPHEROID_THETA_SKY, SPHEROID_THETA_J2000, SPHEROID_THETA_B1950 SPHEROID_SERSICN, SPHEROID_SERSICNERR

$$m_{\texttt{SPHEROID}}(r) = m_0 \exp\left(-b(n)\,\left(\frac{R}{R_e}\right)^{1/n}\right), \tag{1.71}$$

where, for the [36] model, $b(n)$ is the solution to

$$2\gamma[2\,n, b(n)] = \Gamma(2\,n) \tag{1.72}$$

An accurate approximation for the solution for $b(n)$ of (1.72) is [37]:

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

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.

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 $R = 4\,\text{pixels}$, $I(r) = I_0 + (r/a)^3$, 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.

Model-based star-galaxy separation: SPREAD_MODEL

The SPREAD_MODEL estimator has been developed as a star/galaxy classifier for the DESDM pipeline [38], and has also been used in other surveys [39, 40]. SPREAD_MODEL indicates which of the best fitting local PSF model resampled at the current position $\tilde{\boldsymbol{\phi}}$ (representing a point source) or a slightly "fuzzier" resampled model $\tilde{\boldsymbol{G}}$ (representing a galaxy) matches best the image data. $\tilde{\boldsymbol{G}}$ is obtained by convolving the local PSF model with a circular exponential model with scalelength = 1/16 FWHM, and resampling the result at the current position on the pixel grid. SPREAD_MODEL is normalized to allow comparing sources with different PSFs throughout the field:

$$\texttt{SPREAD\_MODEL} = \frac{\tilde{\boldsymbol{G}}^{\mathsf{T}}\mathbf{W}\,\boldsymbol{p}}{\tilde{\boldsymbol{\phi}}^{\mathsf{T}}\mathbf{W}\,\boldsymbol{p}} - \frac{\tilde{\boldsymbol{G}}^{\mathsf{T}}\mathbf{W}\,\tilde{\boldsymbol{\phi}}}{\tilde{\boldsymbol{\phi}}^{\mathsf{T}}\mathbf{W}\,\tilde{\boldsymbol{\phi}}}, \tag{1.73}$$

where $\boldsymbol{p}$ is the image vector centered on the source. $\mathbf{W}$ is a weight matrix constant along the diagonal except for bad pixels where the weight is 0.

Warning: The definition of SPREAD_MODEL above differs from the one given in previous papers, which was incorrect, although in practice both estimators give very similar results.

By construction, SPREAD_MODEL is close to zero for point sources, positive for extended sources (galaxies), and negative for detections smaller than the PSF, such as cosmic ray hits. On images taken with linear detectors, the average SPREAD_MODEL of point sources should not depend on flux nor SNR. This property may be used to identify bad exposures or PSF modeling issues (Fig. 1.9). More importantly, this makes SPREAD_MODEL a very convenient estimator for star-galaxy classification, using a positive threshold to identify extended sources.

The PDF of SPREAD_MODEL is close to Gaussian for isolated point sources at a given SNR; it gets larger for the fainter sources because of image noise. In order to maintain a certain level of purity or completeness across the whole magnitude range, it is therefore necessary to take into account the uncertainty on SPREAD_MODEL, which can be estimated by propagating the uncertainties on individual pixel values:

$$\begin{aligned}\texttt{SPREADERR\_MODEL} \;=\; & \frac{1}{(\tilde{\boldsymbol{\phi}}^{\mathsf{T}}\mathbf{W}\,\boldsymbol{p})^2}\Big((\tilde{\boldsymbol{G}}^{\mathsf{T}}\mathbf{V}\,\tilde{\boldsymbol{G}})\,(\tilde{\boldsymbol{\phi}}^{\mathsf{T}}\mathbf{W}\,\boldsymbol{p})^2 \\ & +(\tilde{\boldsymbol{\phi}}^{\mathsf{T}}\mathbf{V}\,\tilde{\boldsymbol{\phi}})\,(\tilde{\boldsymbol{G}}^{\mathsf{T}}\mathbf{W}\,\boldsymbol{p})^2 \\ & -2\,(\tilde{\boldsymbol{G}}^{\mathsf{T}}\mathbf{V}\,\tilde{\boldsymbol{\phi}})\,(\tilde{\boldsymbol{G}}^{\mathsf{T}}\mathbf{W}\,\boldsymbol{p})\,(\tilde{\boldsymbol{\phi}}^{\mathsf{T}}\mathbf{W}\,\boldsymbol{p})\Big)^{1/2},\end{aligned} \tag{1.74}$$

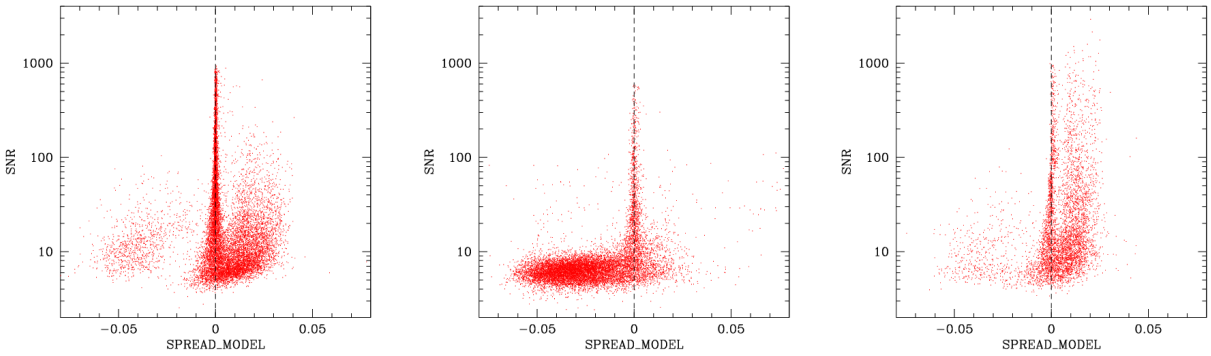


Fig. 1.9: SNR *vs* SPREAD_MODEL for three exposures from [40]. *Left plot*: "good" exposure; extended sources (galaxies and nebulous features) are on the right handside of the stellar locus, and electronic glitches create a small cloud of points on the left handside. *Middle*: exposure with an unusual amount of electronic glitches. *Right*: exposure with tracking/guiding issues; the PSF is too complex for individual sources to be identified as a single objects.

where $\mathbf{V}$ is the noise covariance matrix, which we assume to be diagonal. In practice, one may for instance adopt a threshold for star-galaxy separation which is a combination of a fixed and a variable components, such as $\sqrt{\epsilon^2 + (\kappa \times \texttt{SPREADERR\_MODEL})^2}$, with $\epsilon \approx 5.10^{-3}$ and $\kappa \approx 4$.

BIBLIOGRAPHY

[1] D. C. Wells, E. W. Greisen, and R. H. Harten, 1981. FITS - a Flexible Image Transport System. *A&AS*, 44:363.

[2] A. S. Szalay, A. J. Connolly, and G. P. Szokoly, 1999. Simultaneous Multicolor Detection of Faint Galaxies in the Hubble Deep Field. *AJ*, 117:68–74.

[3] E. Bertin, Y. Mellier, M. Radovich, G. Missonnier, P. Didelon, and B. Morin, 2002. The TERAPIX Pipeline. In D. A. Bohlender, D. Durand, and T. H. Handley, editors, *Astronomical Data Analysis Software and Systems XI*, volume 281 of Astronomical Society of the Pacific Conference Series, 228. 2002.

[4] N. Pogson, 1856. Magnitudes of Thirty-six of the Minor Planets for the first day of each month of the year 1857. *MNRAS*, 17:12–15.

[5] M. R. Calabretta and E. W. Greisen, 2002. Representations of celestial coordinates in FITS. *A&A*, 395:1077–1122.

[6] E. W. Greisen and M. R. Calabretta, 2002. Representations of world coordinates in FITS. *A&A*, 395:1061–1075.

[7] P. B. Stetson, 1987. DAOPHOT - A computer program for crowded-field stellar photometry. *PASP*, 99:191–222.

[8] G. S. Da Costa, 1992. Basic Photometry Techniques. In S. B. Howell, editor, *Astronomical CCD Observing and Reduction Techniques*, volume 23 of Astronomical Society of the Pacific Conference Series, 90. 1992.

[9] K. Pearson, 1895. Contributions to the mathematical theory of evolution.—II. Skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London*, 186:343–414.

[10] G. U. Yule. *An introduction to the theory of statistics*. C. Griffin and Company, limited, 1911.

[11] A. Stuart and K. Ord. *Kendall's Advanced Theory of Statistics: Volume 1: Distribution Theory*. Number vol. 1 ;vol. 1994 in Kendall's Advanced Theory of Statistics. Wiley, 2009. ISBN 9780340614303.

[12] E. Bertin and S. Arnouts, 1996. SExtractor: Software for source extraction. *A&AS*, 117:393–404.

[13] S. Everett, B. Yanny, N. Kuropatkin, E. M. Huff, Y. Zhang, J. Myles, A. Masegian, J. Elvin-Poole, S. Allam, G. M. Bernstein, I. Sevilla-Noarbe, M. Splettstoesser, E. Sheldon, M. Jarvis, A. Amon, I. Harrison, A. Choi, W. G. Hartley, A. Alarcon, C. Sánchez, D. Gruen, K. Eckert, J. Prat, M. Tabbutt, V. Busti, M. R. Becker, N. MacCrann, H. T. Diehl, D. L. Tucker, E. Bertin, T. Jeltema, A. Drlica-Wagner, R. A. Gruendl, K. Bechtol, A. Carnero Rosell, T. M. C. Abbott, M. Aguena, J. Annis, D. Bacon, S. Bhargava, D. Brooks, D. L. Burke, M. Carrasco Kind, J. Carretero, F. J. Castander, C. Conselice, M. Costanzi, L. N. da Costa, M. E. S. Pereira, J. De Vicente, J. DeRose, S. Desai, T. F. Eifler, A. E. Evrard, I. Ferrero, P. Fosalba, J. Frieman, J. García-Bellido, E. Gaztanaga, D. W. Gerdes, G. Gutierrez, S. R. Hinton, D. L. Hollowood, K. Honscheid, D. Huterer, D. J. James, S. Kent, E. Krause, K. Kuehn, O. Lahav, M. Lima, H. Lin, M. A. G. Maia, J. L. Marshall, P. Melchior, F. Menanteau, R. Miquel, J. J. Mohr, R. Morgan, J. Muir, R. L. C. Ogando, A. Palmese, F. Paz-Chinchón, A. A. Plazas, M. Rodriguez-Monroy, A. K. Romer, A. Roodman, E. Sanchez, V. Scarpine, S. Serrano, M. Smith, M. Soares-Santos, E. Suchyta, M. E. C. Swanson, G. Tarle, C. To, M. A. Troxel, T. N. Varga, J. Weller, R. D. Wilkinson, and R. D. Wilkinson, 2022. Dark Energy Survey Year 3 Results: Measuring the Survey Transfer Function with Balrog. *apjs* , 258(1):15.

[14] J. F. Jarvis and J. A. Tyson, 1981. FOCAS - Faint Object Classification and Analysis System. *AJ*, 86:476–495.

[15] L. Infante, 1987. A faint object processing software - Description and testing. *A&A*, 183:177–184.

[16] C. Marmo and E. Bertin, 2008. MissFITS and WeightWatcher: Two Optimised Tools for Managing FITS Data. In R. W. Argyle, P. S. Bunclark, and J. R. Lewis, editors, *Astronomical Data Analysis Software and Systems XVII*, volume 394 of Astronomical Society of the Pacific Conference Series, 619. August 2008.

[17] R. S. Stobie, 1980. Application of moments to the analysis of panoramic astronomical photographs. In D. A. Elliott, editor, *Conference on Applications of Digital Image Processing to Astronomy*, volume 264 of Proc. SPIE, 208–212. 1980.

[18] S. J. Maddox, G. Efstathiou, and W. J. Sutherland, 1990. The APM Galaxy Survey - Part Two - Photometric Corrections. *MNRAS*, 246:433.

[19] M. D. Richard and R. P. Lippmann, 1991. **Neural network classifiers estimate bayesian a posteriori probabilities.** *Neural Computation*, pages 461–483.

[20] M. Saerens, P. Latinne, and C. Decaestecker, 2002. Any reasonable cost function can be used for a posteriori probability approximation. *IEEE Transactions on Neural Networks*, 13(5):1204–1210.

[21] E. Bertin, 2009. SkyMaker: astronomical image simulations made easy. *Mem.Soc.Ast.It*, 80:422.

[22] A. F. J. Moffat, 1969. A Theoretical Investigation of Focal Stellar Images in the Photographic Emulsion and Application to Photographic Photometry. *A&A*, 3:455.

[23] R. G. Kron, 1980. Photometry of a complete sample of faint galaxies. *ApJS*, 43:305–325.

[24] V. Petrosian, 1976. Surface brightness and evolution of galaxies. *ApJL*, 209:L1–L5.

[25] M. R. Blanton, J. Dalcanton, D. Eisenstein, J. Loveday, M. A. Strauss, M. SubbaRao, D. H. Weinberg, J. E. Anderson, Jr., J. Annis, N. A. Bahcall, M. Bernardi, J. Brinkmann, R. J. Brunner, S. Burles, L. Carey, F. J. Castander, A. J. Connolly, I. Csabai, M. Doi, D. Finkbeiner, S. Friedman, J. A. Frieman, M. Fukugita, J. E. Gunn, G. S. Hennessy, R. B. Hindsley, D. W. Hogg, T. Ichikawa, Ž. Ivezić, S. Kent, G. R. Knapp, D. Q. Lamb, R. F. Leger, D. C. Long, R. H. Lupton, T. A. McKay, A. Meiksin, A. Merelli, J. A. Munn, V. Narayanan, M. Newcomb, R. C. Nichol, S. Okamura, R. Owen, J. R. Pier, A. Pope, M. Postman, T. Quinn, C. M. Rockosi, D. J. Schlegel, D. P. Schneider, K. Shimasaku, W. A. Siegmund, S. Smee, Y. Snir, C. Stoughton, C. Stubbs, A. S. Szalay, G. P. Szokoly, A. R. Thakar, C. Tremonti, D. L. Tucker, A. Uomoto, D. Vanden Berk, M. S. Vogeley, P. Waddell, B. Yanny, N. Yasuda, and D. G. York, 2001. The Luminosity Function of Galaxies in SDSS Commissioning Data. *AJ*, 121:2358–2380.

[26] N. Yasuda, M. Fukugita, V. K. Narayanan, R. H. Lupton, I. Strateva, M. A. Strauss, Ž. Ivezić, R. S. J. Kim, D. W. Hogg, D. H. Weinberg, K. Shimasaku, J. Loveday, J. Annis, N. A. Bahcall, M. Blanton, J. Brinkmann, R. J. Brunner, A. J. Connolly, I. Csabai, M. Doi, M. Hamabe, S.-I. Ichikawa, T. Ichikawa, D. E. Johnston, G. R. Knapp, P. Z. Kunszt, D. Q. Lamb, T. A. McKay, J. A. Munn, R. C. Nichol, S. Okamura, D. P. Schneider, G. P. Szokoly, M. S. Vogeley, M. Watanabe, and D. G. York, 2001. Galaxy Number Counts from the Sloan Digital Sky Survey Commissioning Data. *AJ*, 122:1104–1124.

[27] E. Bertin. *Photométrie automatique de galaxies et contraintes sur leur évolution récente*. PhD thesis, Université Paris VI, June 1996.

[28] C. E. Duchon, 1979. Lanczos Filtering in One and Two Dimensions. *Journal of Applied Meteorology*, 18:1016–1022.

[29] C. Kanzow, N. Yamashita, and M. Fukushima, 2004. Levenberg-Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints. *Journal of Computational and Applied Mathematics*, 172(2):375–397.

[30] C. M. Hirata, R. Mandelbaum, U. Seljak, J. Guzik, N. Padmanabhan, C. Blake, J. Brinkmann, T. Budávari, A. Connolly, I. Csabai, R. Scranton, and A. S. Szalay, 2004. Galaxy-galaxy weak lensing in the Sloan Digital Sky Survey: intrinsic alignments and shear calibration errors. *MNRAS*, 353:529–549.

[31] P. Melchior and M. Viola, 2012. Means of confusion: how pixel noise affects shear estimates for weak gravitational lensing. *MNRAS*, 424:2757–2769.

[32] A. Refregier, T. Kacprzak, A. Amara, S. Bridle, and B. Rowe, 2012. Noise bias in weak lensing shape measurements. *MNRAS*, 425:1951–1957.

[33] T. Kacprzak, J. Zuntz, B. Rowe, S. Bridle, A. Refregier, A. Amara, L. Voigt, and M. Hirsch, 2012. Measurement and calibration of noise bias in weak lensing galaxy shape estimation. *MNRAS*, 427:2711–2722.

[34] M.I.A. Lourakis, 2004. Levmar: levenberg-marquardt nonlinear least squares algorithms in C/C++.

[35] C. Broyden, 1965. **A class of methods for solving nonlinear simultaneous equations.** *Mathematics of Computation*, 19:577–593.

[36] J. L. Sérsic. *Atlas de Galaxias Australes*. Cordoba, Argentina: Observatorio Astronomico, 1968, 1968.

[37] L. Ciotti and G. Bertin, 1999. Analytical properties of the R^(1/m) law. *A&A*, 352:447–451.

[38] J. J. Mohr, R. Armstrong, E. Bertin, G. Daues, S. Desai, M. Gower, R. Gruendl, W. Hanlon, N. Kuropatkin, H. Lin, J. Marriner, D. Petravic, I. Sevilla, M. Swanson, T. Tomashek, D. Tucker, and B. Yanny, 2012. The Dark Energy Survey data processing and calibration system. In *Software and Cyberinfrastructure for Astronomy II*, volume 8451 of Proc. SPIE, 84510D. September 2012.

[39] S. Desai, R. Armstrong, J. J. Mohr, D. R. Semler, J. Liu, E. Bertin, S. S. Allam, W. A. Barkhouse, G. Bazin, E. J. Buckley-Geer, M. C. Cooper, S. M. Hansen, F. W. High, H. Lin, Y.-T. Lin, C.-C. Ngeow, A. Rest, J. Song, D. Tucker, and A. Zenteno, 2012. The Blanco Cosmology Survey: Data Acquisition, Processing, Calibration, Quality Diagnostics, and Data Release. *ApJ*, 757:83.

[40] H. Bouy, E. Bertin, E. Moraux, J.-C. Cuillandre, J. Bouvier, D. Barrado, E. Solano, and A. Bayo, 2013. Dynamical analysis of nearby clusters. Automated astrometry from the ground: precision proper motions over a wide field. *A&A*, 554:A101.