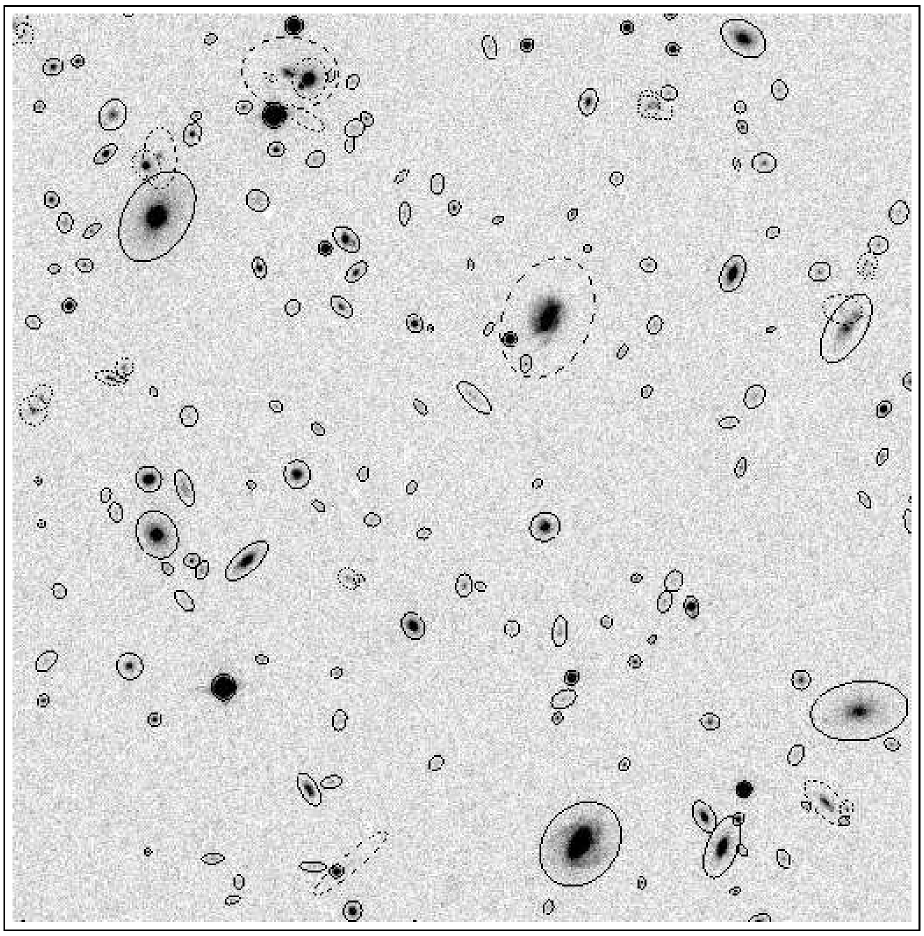
SExtractor
1.0a
User's guide

E. BERTIN
Institut d'Astrophysique de Paris [*]



[*]present address: Sterrewacht Leiden, PO Box 9513, 2300 RA, Leiden, The Netherlands

Contents

1 What is SExtractor ?

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

- Simplicity of usage and configuration.
- Speed: typically 50 kpixel/s with a SUN Sparc 20.
- Ability to work with very large images (up to 65534×65534 pixels), without being limited by memory.
- Robust deblending of overlapping extended objects.
- Possibility to convolve the image "on-the-fly" to improve detectability.
- Neural-Network-based star/galaxy classifier.
- Flexible catalog output of desired parameters only.
- Special mode for photographic scans.
- Modularity of the code that enables one to implement ones own parameters.

In short, the goal in making SExtractor was to find a compromise between refinement in both detection and measurements, and computational speed. Note: the original software was made for Schmidt plate scans only. This version is an adaptation which is intended to run on any 2-dimensional astronomical FITS frame, coming for example from a CCD or an InfraRed array. But, for historical reasons, many tests shown here have been conducted on photographic material or simulations.

2 Technical overview of the software

The global working of SExtractor is represented in fig. 1. From now on, we assume that before using SExtractor :

For CCD frames: flatfielding, removal of cosmics and bad columns,

For IR-arrays: skysubstraction, flatfielding and removal of glitches

has been done in order to obtain reliable results.

The complete analysis of the image is done in seven steps: estimation of the sky background, thresholding, deblending, filtering of detections, photometry, classification and finally catalog output. Let's now enter a little more into the details of each of these operations (in the following, keywords in typewriter font refer to parameters from the configuration file or from the parameter file, see §3.1 and §3.2).

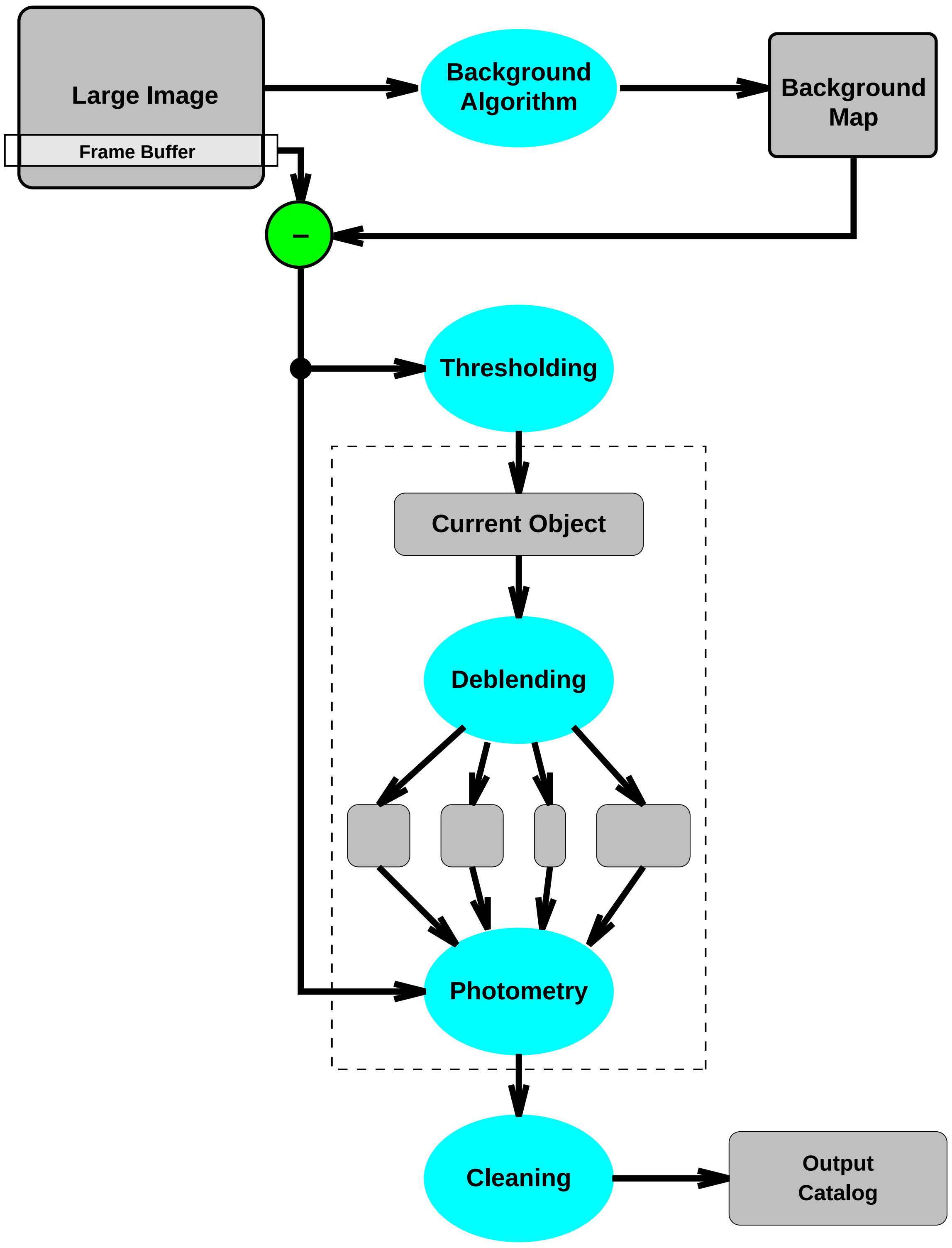


Figure 1: Global organization of the main SExtractor procedures. The "pipeline" through which all detected objects are processed is enclosed within dashed lines.

2.1 Background estimation

The value measured at each pixel is a function of the sum of a "background" signal and light coming from the objects we are interested in. To be able to detect the faintest of these objects and also to measure accurately their fluxes, we need to have an accurate estimate of the background level in any place of the image, a "background map". Strictly speaking, there should be one background map per object, that is, what would the image look like if that object was absent. But here we suppose that most objects do not overlap, which is generally the case for high galactic latitude fields.

To construct its background map, SExtractor makes a first pass through the pixel data, computing an estimator for the local background in each mesh of a grid that covers the whole frame. Our background estimator is a combination of $\kappa.\sigma$ clipping and mode estimation, similar to the one employed in Stetson's DAOPHOT program (see e.g. Da Costa 1992). Briefly, the local background histogram is clipped iteratively until convergence at $\pm 3\sigma$ around its median; if σ is changed by less than 20% during that process, we consider that the field is uncrowded and we simply take the mean of the clipped histogram as a value for the background; otherwise we estimate the mode with:

$$\text{Mode} = 2.5 \times \text{Median} - 1.5 \times \text{Mean} \tag{1}$$

This expression is different from the usual approximation

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

(e.g. Kendall and Stuart 1977), but was found more accurate with our clipped distributions, from the simulations we made. Fig. 2 shows that the expression of the mode above is considerably less affected[1] by crowding than a simple clipped mean — like the one used in FOCAS (Jarvis and Tyson 1981) or by Infante (1987) — but is $\approx 30\%$ noisier. That's why we turn back to the mean for uncrowded fields.

Once the grid is set up, a median filter can be applied to it, in order to suppress possible local overestimations due to bright stars. The resulting background map is then simply a bilinear interpolation between the meshes of the grid.

The choice of the mesh size is very important (`BACK_XSIZE` and `BACK_YSIZE`). When it is too small, the background estimation is affected by the presence of objects and random noise. When it is too large, it cannot reproduce the small scale variations of the background. Therefore a good compromise has to be found by the user. Typically, for reasonably sampled images, a width[2] of 32 to 256 pixels works fine. The user has again some control over the background map by specifying the size of the median filter (`BACK_FLTRXSIZE` and `BACK_FLTRYSIZE`). A width and height of 1 means that no filtering will be applied to the background grid. Usually a size of 3×3 is enough, but it may be necessary to use larger dimensions, especially to compensate, in part, for small background mesh sizes.

The background estimation operation can take a considerable time on the largest images (e.g. half an hour for a 32000×32000 frame on an HP 755 workstation).

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

[2] SExtractor offers the possibility to have rectangular background meshes; but it is advised to use square ones, except in some very special cases (rapidly varying background in one direction for ex.).

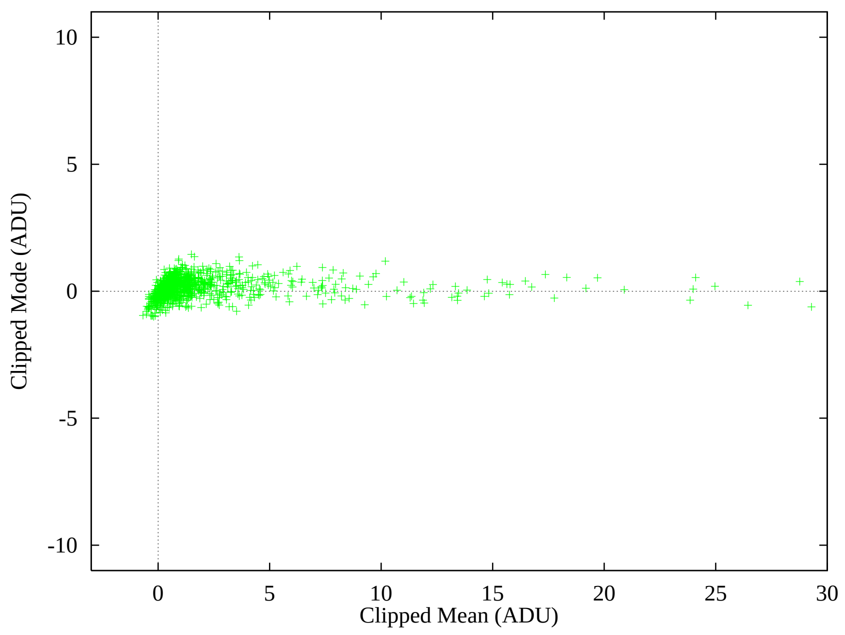


Figure 2: Simulations of 32×32 pixels background meshes polluted by random gaussian profiles. The true background lies at 0 ADU. While being slightly noisier, the clipped "Mode" gives a more robust estimate than a clipped Mean in crowded regions.

2.2 Thresholding

Once the background map has been created, the program enters its actual extraction phase: search for connected sets of pixels above a given threshold. For large images, the whole data cannot fit into the machine's memory. That's why only a fraction of the frame is accessible to SExtractor at any time. In the following, we shall call that fraction the *frame buffer*. The frame buffer can be seen as a window that has the same width as the full image and moves along line by line. Its height, in scan lines, has to be specified by the user according to the amount of memory available on the machine (`MEMORY_BUFSIZE`). It can be as small as 4 lines high; but while very small frame buffers don't affect source extraction or deblending, they certainly slow down the program a bit, and above all they represent a handicap for aperture photometry (see §2.4).

To perform thresholding, SExtractor uses a very efficient one-pass, 8-connectivity algorithm by Lutz (1979). The extraction parameters definable by the user are the threshold (expressed in number of background noise standard deviations or in mag/arcsec^2 if `THRESHOLD_TYPE = MAGNITUDE`) and the minimum number of connected pixels required (`EXTRACT_MINAREA`). Typical values of 1.0 to 2.0 for `THRESHOLD`, and 5 for `EXTRACT_MINAREA` generally give the best results.

A convolution can be applied to the image before extraction by setting `CONVOLVE` to "Y". The convolution mask `CONVOLVE_NAME` is supplied by the user (see §3.4) and can be of any size (watch out for processing time !). It has been shown that an optimum detection of faint point sources is achievable by convolving the data with a gaussian whose FWHM is roughly equal to the seeing (Irwin 1985). Of course it is also possible to use more specific convolution masks, for instance to detect structures at given scales. Note that convolution affects only detection and deblending. However it has for instance little effect on aperture magnitudes.

Until all the pixels above the threshold of an object have been scanned, they are stored in a *pixel stack*. This stack is shared with pixels from all the other "unfinished" objects. The stack size (MEMORY_PIXSTACK) is set by the user. It should be as large as possible, depending on the memory available on your machine. Indeed, when SExtractor has no more room to store incoming pixels, it "sacrifices" the largest object extracted so far to gain memory space. It will in most cases be a very bright star, or an artefact such as one of the strips often found at the edges of images. "Sacrificed" objects are nevertheless included in the output catalog. Of course they are flagged accordingly, because their parameters have been corrupted, and no deblending procedure is applied to them.

2.3 Deblending

Each time an object extraction is completed, the connected set of pixels passes through a sort of filter that tries to split it into eventual overlapping components. This case appears more frequently when the field is crowded or when the detection threshold is set very low. The deblending method adopted in SExtractor , is based on *multithresholding*, and works on any kind of object, but it is unable to deblend components that are so close that no saddle is present in their profile. However, as no assumption has to be made on the shape of the objects, it is perfectly suited for galaxies (and high galactic latitude stellar fields).

Our method is an attempt to deblend practically any kind of astronomical object. Typical problematic cases include patchy, extended **Sc** galaxies (which have to be considered as single entities), and close or interacting pairs of optically faint galaxies (which have to be considered as separate objects). Basically, the multi-thresholding algorithm employes a multiple isophotal analysis technique similar to those in use at the APM and the COSMOS machines (Beard, McGillivray and Thanish 1991); in a first time, each extracted set of connected pixels is rethresholded at N levels linearly or exponentially spaced between its primary extraction threshold and its peak value. This gives us a sort of 2-dimensional "model" of the light distribution within the object(s), which is stored in the form of a tree structure (fig. 3). Then the algorithm goes downwards, from the tips of branches to the trunk, and decides at each junction whether it shall extract two (or more) objects or continue its way down. To meet the conditions described earlier, the following simple decision criteria are adopted: at any junction threshold t_i, any branch will be considered as a separate component if

(1) the integrated pixel intensity (above t_i) of the branch is greater than a certain fraction δ_c of the total intensity of the composite object.

(2) condition (1) is verified for at least one more branch at the same level i.

Note that ideally, condition (1) is both flux- and scale-invariant. However for faint, poorly resolved objects, the efficiency of the deblending is limited mostly by seeing and sampling. From the analysis of both small and extended galaxy images, a compromise value for the constrast parameter δ_c of 0.005 has proved to be optimum. This should normally exclude to separate objects with a difference in magnitude greater than ≈ 6. However, using a linear scale for the thresholds with photographic material, it is possible to go up to ≈ 10 mag. The same figure applies for an exponential scale with a linear device (CCD).

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

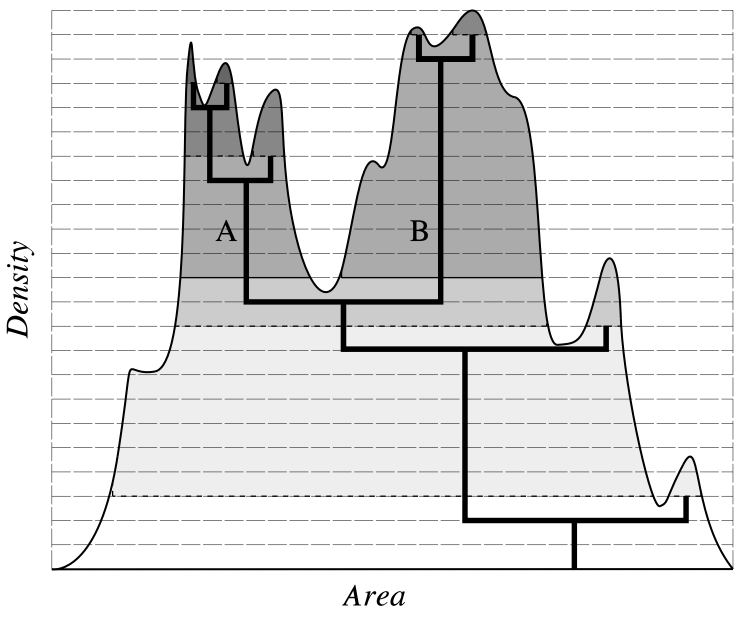


Figure 3: A schematic diagram of the method used to deblend a composite object. The areal profile of the object (smooth curve) can be described in a tree-structured way (thick lines). The decision to regard or not a branch as a distinct object is determined according to its relative integrated intensity (tinted area). In that case above, the original object shall split into two components A and B. Remaining pixels are assigned to their most credible "progenitors" afterwards.

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

To test the effects of deblending on photometry and astrometry measurements, we made several simulations of photographic images of double stars with different separations and magnitudes under typical observational conditions (fig. 4). It is obvious that multiple isophotal techniques fail when there is no saddle point present in profiles (i.e. for distance between stars $< 2\sigma$ in the case of gaussian images). We measured a magnitude error ≤ 0.2 mag and a shift of the centroid (≤ 0.4 pixels) for the fainter star in the very worst cases, but no other systematic effects were noticeable.

The user can control the multi-thresholding operation through 3 parameters. The first one is the number of deblending thresholds (`DEBLEND_NTHRESH`). A good value is 32. Higher values are generally useless, except perhaps for images having an unusually high dynamic range. In case of memory problems, decreasing the number of thresholds to say, 8 or even less may be a solution. But then of course a degradation of the deblending performances may occur. The second parameter is the contrast parameter (`DEBLEND_MINCONT`). As described above, a value of 0.005 gives best results. Putting it to 0 means that even the faintest local peaks in the profile will be considered as separate objects. Putting it to 1 means that no deblending will be authorized. The last parameter concerns the kind of scale used for the thresholds. If the image comes from photographic material, then a linear scale has to be used (`DETECTION_TYPE = PHOTO`). Otherwise, for an image obtained with a linear device like a CCD, an exponential scale is more appropriate (`DETECTION_TYPE = CCD`).

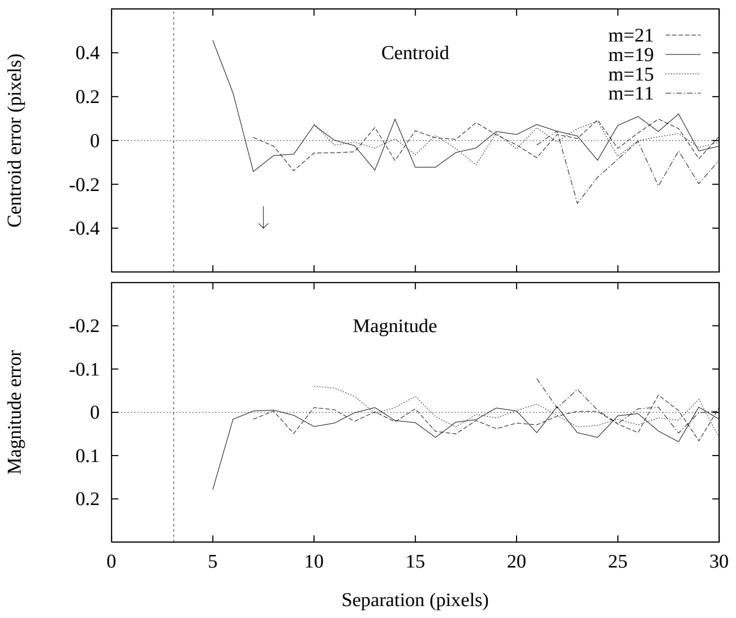


Figure 4: Centroid and corrected isophotal magnitude errors for a simulated 19^{th} magnitude star blended with a $11, 15, 19$ and 21^{th} mag. companion as a function of distance (expressed in pixels). Lines stop at the left when the objects are too close to be deblended. The dashed vertical line is the theorical limit for unsaturated stars with equal magnitudes. In the centroid plot, the arrow indicates the direction of the neighbour. The simulation assumes a 1 hour exposure with the OCA telescope on a IIIaJ plate and Moffat profiles with a seeing FWHM of 3 pixels (2 ").

2.4 Photometry

SExtractor has currently the possibility to compute four types of magnitude: isophotal, *corrected-isophotal*, fixed-aperture and *adaptive-aperture*. For all magnitudes, a zero-point correction can be applied with the `MAG_ZEROPOINT` keyword [3].

Isophotal magnitudes are computed simply, using the detection threshold as the lowest isophote.

Corrected isophotal magnitudes can be considered as a quick-and-dirty way for retrieving the fraction of flux lost by isophotal magnitudes. If we make the assumption that the intensity profiles of the faint objects recorded on the plate are roughly gaussian because of atmospheric blurring, then the fraction $\eta = \frac{I_{iso}}{I_{tot}}$ of the total flux enclosed within a particular isophote reads (see Maddox et al. 1990):

$$(1 - \frac{1}{\eta}) \ln(1 - \eta) = \frac{A.t}{I_{iso}} \qquad (3)$$

where A is the area and t the threshold related to this isophote. Eq. (3) 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:

$$\eta \approx 1 - 0.1961 \frac{A.t}{I_{iso}} - 0.7512 \left(\frac{A.t}{I_{iso}} \right)^2 \qquad (4)$$

[3]the `THRESHOLD` surface brightness magnitude is also affected by `MAG_ZEROPOINT`

A "total" magnitude m_{tot} estimate is then

$$m_{tot} = m_{iso} + 2.5 \log \eta \tag{5}$$

Clearly this cheap correction works best with stars; and although it is shown to give tolerably accurate results with most disk galaxies, it fails with ellipticals because of the broader wings of their profiles. Therefore we recommend to apply it only as a *better-than-nothing* substitute to aperture magnitude in crowded cases.

Fixed-aperture magnitudes estimate the flux above the background within a circular aperture. The diameter of the aperture in pixels (`PHOTOM_APERTURE`) is supplied by the user (in fact it does not need to be an integer since each "normal" pixel is subdivided in 5×5 subpixels before measuring the flux within the aperture).

Automatic aperture magnitudes are intended to give the most precise estimate of "total magnitudes", at least for galaxies. Our automatic aperture photometry routine is inspired by Kron's "first moment" algorithm (1980). (1) We define an elliptical aperture whose elongation ϵ and position angle θ are defined by second order moments of the object's light distribution. The ellipse is scaled to $R_{max}.\sigma_{iso}$ (typically $6\sigma_{iso}$, which corresponds roughly to 2 isophotal "radii"). (2) Within this aperture we compute the "first moment":

$$r_1 = \frac{\sum rI(r)}{\sum I(r)} \tag{6}$$

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

Aperture magnitudes are unfortunately sensitive to crowding. Therefore we suggest to replace the aperture magnitude by the corrected-isophotal one when an object is too close from its neighbours (2 isophotal radii for instance). This is done automatically when using the `MAG_BEST` magnitude: `MAG_BEST` = `MAG_AUTO` when it is sure that no neighbour can bias `MAG_AUTO` by more than 10%, or `MAG_BEST` = `MAG_ISOCOR` otherwise. Experience shows that the `MAG_ISOCOR` and `MAG_AUTO` magnitude loose about the same fraction of flux on most images: around 0.06 % for default extraction parameters.

Figure 5 shows the mean loss of flux measured with isophotal (threshold = 24.4 mag.arsec^{-2}), corrected isophotal and automatic aperture photometries for simulated galaxy B_J on a typical Schmidt-survey plate image.

Geometrical constraints When measuring aperture magnitudes, all the pixels of an object, plus many others around it, must be accessible in memory. That is, they must lie within the upper and lower boundaries of the frame buffer. So we have introduced an "isophotal-to-aperture ratio" ρ parameter (`SCAN_ISOAPRATIO`) that enables the user to control the geometrical

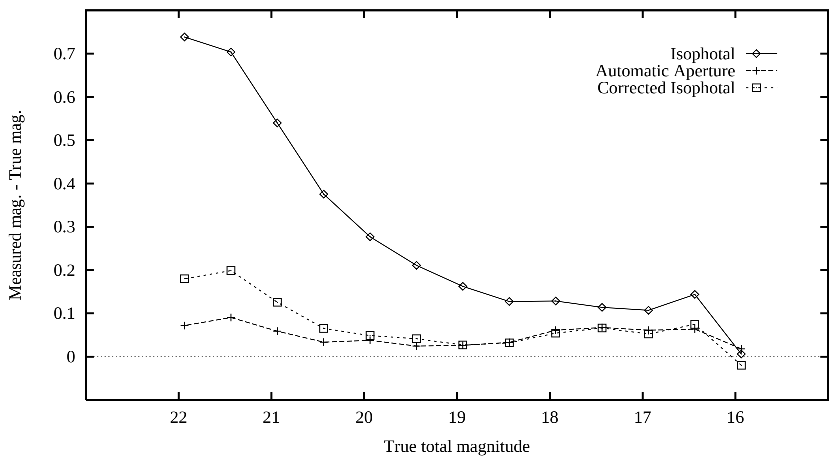


Figure 5: Flux lost (expressed as a mean magnitude difference) with different faint-object photometry techniques as a function of total magnitude (see text). Only isolated galaxies (no blends) of the simulations have been considered.

constraints of aperture photometry. If ρ is less than the ratio $\frac{\text{object (vertical) size}}{\text{buffer size}}$, then the object is flagged as "aperture truncated" (flag 16 in §3.3). It is easy to show that it gives also an optimum buffer *margin* size[4] equal to $h.\frac{(1-\rho)}{2}$, where h is the frame-buffer size in pixels (see fig. 6). A secure value for `SCAN_ISOAPRATIO` is 0.6.

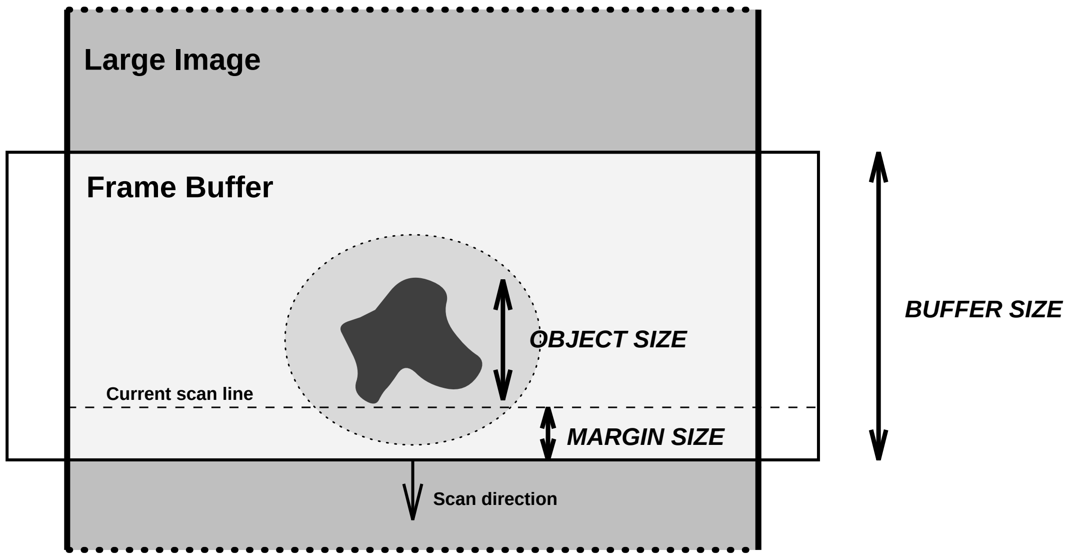


Figure 6: Geometrical constraints concerning aperture magnitudes and frame-buffer size. Aperture magnitudes are computed if and only if the ratio $\frac{\text{object size}}{\text{buffer size}}$ is less than a certain factor (`SCAN_ISOAPRATIO`), specified by user. `SCAN_ISOAPRATIO` also determines the buffer margin size.

Photographic photometry In `PHOTO` mode (`DETECT_TYPE = PHOTO`), 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

[4]The buffer margin size is the number of lines that is kept constant (except at the end of the frame) between the current scan line and the bottom limit of the frame buffer.

transformation

$$I = I_0.10^{\frac{D}{\gamma}} \tag{7}$$

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

$$I_0 = \frac{\gamma}{\ln 10}.10^{-0.4m_0} \tag{8}$$

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

Errors on magnitude An estimate of the error[5] is available for each type of magnitude. It is computed through

$$\Delta m = 1.0857\frac{\sqrt{A\sigma^2 + \frac{F}{g}}}{F} \tag{9}$$

where A is the area (in pixels) over which the total flux F (in ADU) is summed, σ the standard deviation of noise (in ADU) estimated from the background, and g the detector gain (GAIN parameter[6] , in e^-/ADU). For corrected-isophotal magnitudes, a term, derived from Eq. 4 is quadratically added to take into account the error on the correction itself.

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

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

where $I(x,y)$ is the contribution of pixel (x,y) to the total flux (Eq. 7). The GAIN is ignored in PHOTO mode.

Background is the last point relative to photometry. The assumption made in §2.1 — that the "local" background associated to an object can be interpolated from the global background map — is no longer valid in crowded regions. An example is a globular cluster superimposed on a bulge of galaxy. SExtractor offers the possibility to estimate locally the background used to compute magnitudes. When this option is switched on (BACKPHOTO_TYPE = LOCAL instead of GLOBAL), the "photometric" background is estimated within a "rectangular annulus" around the isophotal limits of the object. The thickness of the annulus (in pixels) can be specified by the user with BACKPHOTO_SIZE. 24 is a typical value.

2.5 CLEANing

When using low thresholds, spurious detections are often created at the wings of objects with shallow profiles (for example, elliptical galaxies). It comes from the fact that the background is locally higher there, leading to a lower relative threshold and thus a higher detection rate of noise peaks. One solution to this problem is to verify for each detection if the object would have been detected provided there were no neighbours. This is what the CLEANing procedure does:

[5] Important: this error must be considered only as a lower value since it does not take into account the (complex) uncertainty on the local background estimate.

[6] Setting GAIN to 0 in the configuration file is equivalent to $g = +\infty$

while detections are made, objects are put in a *FIFO* (*First-In First-Out*) stack. When the stack is full, objects are examined one by one before being sent to the final catalog. SExtractor checks that their mean surface brightness is still higher than the threshold when neighbours are "removed". The contribution from the wings of neighbours is computed assuming a gaussian extrapolation of their profiles. As real profiles have, in general, broader wings than a pure gaussian, it is, most of the time, necessary to expand their estimated width by a certain factor $f_{\rm clean}$.

Practically, it is advised to always leave the `CLEAN` option to `Y`, except for heavily crowded fields or when speed is really critical. The `CLEAN_PARAM` parameter controls the $f_{\rm clean}$ factor. Typical values range between 1.0 and 2.0; a good compromise is 1.5. Finally, the object stack-size, `CLEAN_OBJSTACK`, should be large enough so that a significant height of the image is considered during cleaning.

2.6 Star/galaxy separation

In most imaging surveys, a separation of stars and galaxies is required. This is traditionally a tricky job; the seeing and the limited signal/noise both contribute to bluring the distinction between faint point-sources and extended objects. We have chosen to confide this classification task to a neural network. The superior performances of neural networks in the classification of photographic images have already been demonstrated (Odewahn *et al* 1992, Bertin 1994). Even with linear images, they can lead to significant improvement over traditional techniques, especially concerning their robustness when dealing with overlapped objects. We will now present briefly the classifier; for a more complete description, please refer to Bertin and Arnouts (1996).

Principle First of all, the goal of the neural classifier is to provide an optimal transformation from the parameter-space defined by a set of observables describing the objects, to the space of classes. Here the space of classes is one-dimensional, and from now on we shall refer to it through the "stellarity- index" `CLASS_STAR`, which, by definition, will equal 0 for galaxies, and 1 for stars. The input parameters are simply 8 isophotal areas and the peak intensity, plus one control parameter. It can be shown in pratice that "raw" parameters such as those do provide a very good description of object profiles, and particularly of their fuzziness (see Bertin 1996 for more details). The network needs to know what is the instrinsic fuzziness of the image: we give it the seeing FHWM as a control parameter.

Training In order to make the network "learn" how to distinguish point-sources from galaxies, we trained it with more than 10^6 images of stars and galaxies simulated with different conditions of pixel-scale, seeing and detection limits. The PSF used was a Moffat (1969) function, with a variable $2 < \beta < 4$ parameter (defining the extent of the wings). With this training, the network should be able to deal with most ground-based telescope linear images, with a seeing FWHM comprised between 0.025 (!) and 5.5". The result of the training (connection weights) was saved to the `default.nnw` ASCII table. `default.nnw` is presently the only neural-network-weights table available[7]. Therefore the `STARNNW_NAME` should be set to `default.nnw` in the configuration file.

Usage It is very simple to classify stars and galaxies with SExtractor. Still, for best results with faint objects, one needs to do some fine-tuning. This is the price one has to pay for having

[7]We may add other ones in the future...

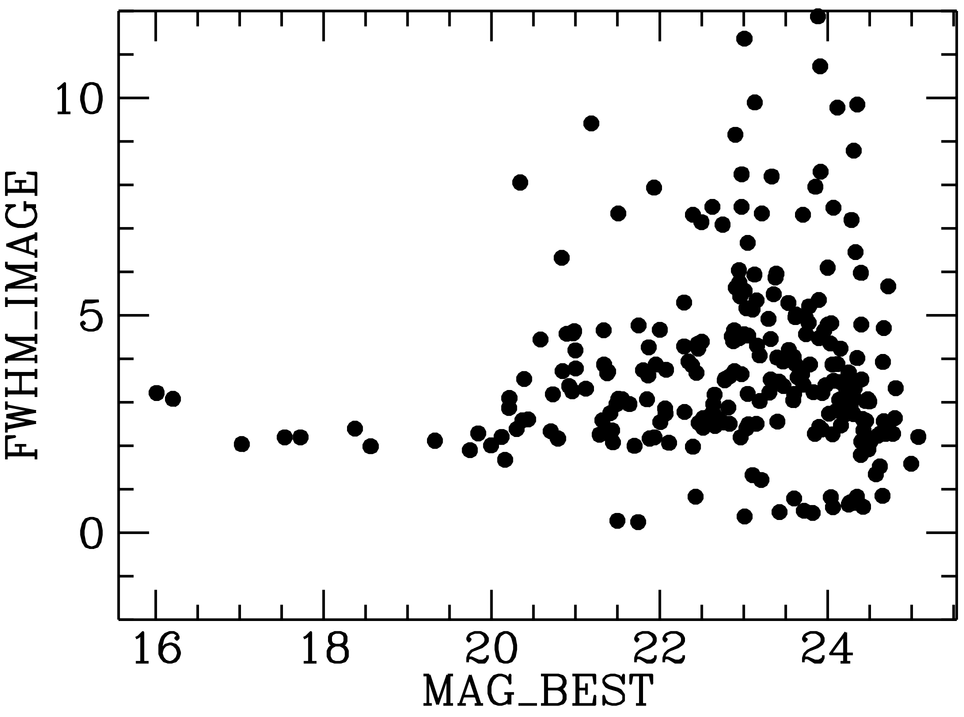


Figure 7: FWHM_IMAGE parameter as a function of MAG_BEST magnitude for objects detected in a deep R-band CCD frame. Note the horizontal alignment of non-saturated stars at FWHM = 2.1 pixels, as well as the cosmic rays in the lower-right part of the plot.

a high discrimination power. Two configuration-file parameters are concerned: the pixel size PIXEL_SCALE, and the seeing Full-Width-at-Half-Maximum SEEING_FWHM, both in arcsec. In fact, the only sensitive parameter is the SEEING_FWHM/PIXEL_SCALE ratio. You don't need to know precisely the pixel scale, as long as you know the seeing FWHM in pixels with a good accuracy (5-10%). If not, you can run SExtractor a first time on the same image and ask for the FWHM_IMAGE (or FWHM_WORLD) parameter[8]; it is intended to give a good estimate of the FWHM for non-saturated bright stars. Besides, the FWHM can also be used to separate astronomical objects from image defects like cosmic-rays (fig 7). *Note: there are several ways to compute the FWHM of a profile. They can lead to significantly different estimates because the PSF is almost never perfectly gaussian (especially in the wings). The true FWHM is generally unusable since there are not enough pixels involved. The* FWHM_IMAGE *parameter is computed by doing a gaussian fit over the upper 80% of the profile. It is corrected for undersampling effects (down to FWHM $\leq$ 1 pixel) assuming an underlying gaussian profile. It proves to be accurate within a few percents for bright stars.* **This definition serves as the reference for tuning the SEEING_FWHM parameter.**

Here is a trick to tune SEEING_FWHM at best: pick out a faint, ambiguous object in the image, (what may be a compact galaxy for instance). By adjusting SEEING_FWHM, you should be able to bring its CLASS_STAR parameter to an intermediate value (between 0.1 and 0.9). You are then nearly sure that the SEEING_FWHM parameter is very close to its optimal value. If your image contains stars which are bright enough, you might also use the sexseeing SExTool.

[8]The amount of CPU involved in FWHM computations is far from beeing negligible in a crowded field; so be careful not to leave FWHM_IMAGE by mistake in the .param file if you don't really need it.

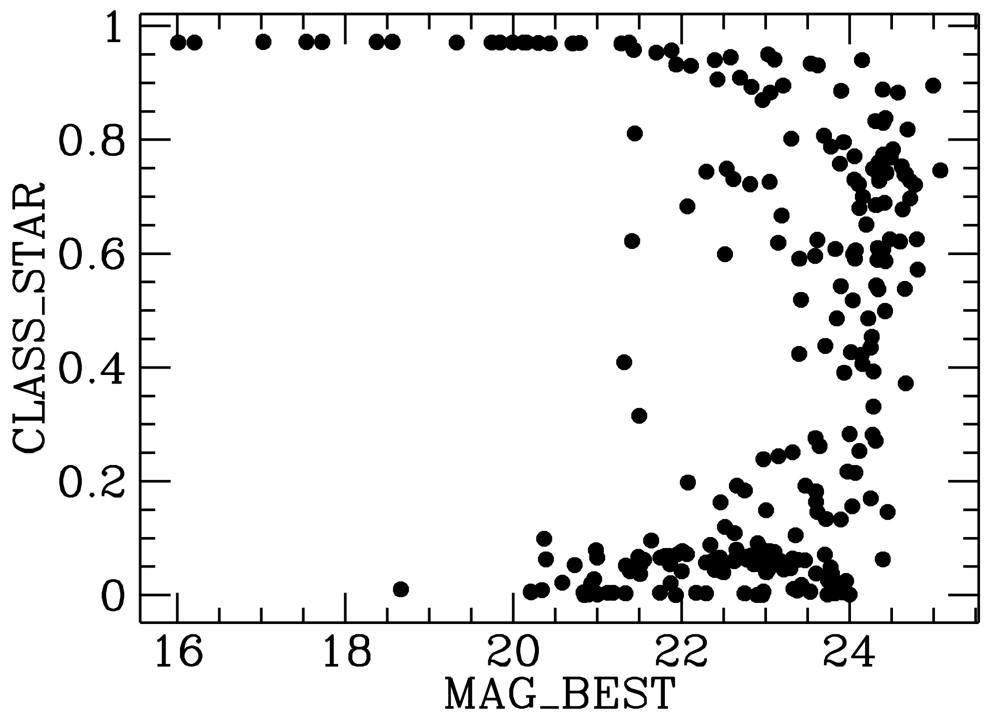


Figure 8: CLASS_STAR parameter as a function of MAG_BEST magnitude after adjustment of SEEING_FWHM.

Interpretation For each classified object, the value of CLASS_STAR is an estimate of the confidence given by the classifier. It would be dangerous to interpret this directly as a bayesian probability to be a star, since the network training has been done with a synthetic sample having slightly different properties. The fig. 8 shows objects from the same image as fig 7, in the MAG_BEST-CLASS_STAR space: note the vanishing constrast between the two classes as magnitude increases.

!!NOTE: THE STAR/GALAXY CLASSIFIER IS STILL EXPERIMENTAL. HENCE BE SURE TO CHECK CLASSIFICATION RESULTS BEFORE USING THEM!!

2.7 Catalog output

The catalog is written object by object[9], during the extraction. The file name should be specified with CATALOG_NAME. It is possible to choose between two formats (through the CATALOG_TYPE keyword): ASCII or FITS. ASCII catalogs contain one object per line, in a human readable form, whereas FITS catalogs store data as a FITS "binary-table" (IEEE format). For large fields, we recommend to use the FITS option, because it produces smaller catalogs. However, ASCII catalogs are easier to handle when they contain less than say, 10^3 objects. FITS binary-tables can be read by astronomical packages like IDL, IRAF of MIDAS[10].

Whichever format used, one selects the nature and the name of each parameter listed in the

[9]However, if an abort occurs at some moment during the working of SExtractor , the catalog may not reflect the real extraction status because of buffering.

[10]The FITS binary-table format has recently been formally approved by IAU as part of the FITS standard (Cotton et al. 1995).

catalog in a "parameter file" (see §3.2). The name of that file has to be supplied with the `PARAMETERS_NAME` keyword.

3 Using SExtractor

SExtractor is run from your shell with the following syntax:

% `sex` *Picture* `[ -c` *configuration-file*`]` `[ -`*Parameter1 Value1*`]` `[ -`*Parameter2 Value2*`]` ...

The part enclosed within brackets is optional. Any "`-`*Parameter Value*" statement in the command-line overrides the corresponding definition in the configuration-file (see below).

3.1 The Configuration file

Each time SExtractor is run, it looks for a configuration file. If no filename is given in the command line, the program takes all its parameters from a file called `default.sex` in the current directory.

3.1.1 Format of the configuration file

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

Parameter   *Value*

Extra spaces or linefeeds are ignored. Comments must begin with a "#" and end with a linefeed. Values can be of different types: strings (enclosed between double quotes), floats, integers, keywords or boolean (`Y/y` or `N/n`). Any missing parameter is set to its default value, and a warning message is printed.

3.1.2 Parameter list

Here is a list of all the parameters known to SExtractor . For a detailed description of their meaning, please refer to chapter 2.

Parameter	type / keyword	Description
`BACK_FLTRXSIZE`	*integer*	Width (in pixels) of the background-filtering mask.
`BACK_FLTRYSIZE`	*integer*	Height (in pixels) of the background-filtering mask.
`BACK_XSIZE`	*integer*	Width of a background mesh (in pixels).
`BACK_YSIZE`	*integer*	Height of a background mesh (in pixels).
`BACKPHOTO_THICK`	*integer*	Thickness (in pixels) of the background `LOCAL` annulus.
`BACKPHOTO_TYPE`	*keyword*	Background used to compute magnitudes:
	`GLOBAL`	– taken directly from the background map,
	`LOCAL`	– recomputed in a "rectangular annulus" around the object.
`CATALOG_NAME`	*string*	Name of the output catalog. If the name "`STDOUT`" is given and `CATALOG_TYPE` is `ASCII`, then the catalog will be piped to the standard output (*stdout*)

CATALOG_TYPE	*keyword*	Format of output catalog:
	ASCII	– the simplest, but space and time consuming,
	FITS	– FITS format (binary table).
CHECKIMAGE_NAME	*string*	Filename for the “check-image”.
CHECKIMAGE_TYPE	*keyword*	Type of information to put in the “check-image”:
	NONE	– normal mode: no check-image,
	BACKGROUND	– background map,
	-BACKGROUND	– background-substracted image,
	CONVOLVED	– background-substracted convolved image (if CONVOLVE = Y),
	OBJECTS	– objects detected,
	SEGMENTATION	– display patches corresponding to pixels attributed to each object,
	APERTURES	– MAG_APER and MAG_AUTO integration limits.
CLEAN	*boolean*	If true, a “cleaning” of the catalog is done before being written to disk.
CLEAN_OBJSTACK	*integer*	Number of catalog entries in the stack of objects for “cleaning”. Multiply by ≈ 100 to have the memory space occupied in bytes.
CLEAN_PARAM	*float*	Efficiency of “cleaning”.
CONVOLVE	*boolean*	If true, a convolution mask is applied to the data before extraction.
CONVOLVE_NAME	*string*	Name of the file containing the convolution mask.
CONVOLVE_NORM	*boolean*	If true, a normalization of the convolution mask is done before using it.
DEBLEND_MINCONT	*float*	Minimum contrast parameter for deblending.
DEBLEND_NTHRESH	*integer*	Number of deblending sub-thresholds.
DETECTION_TYPE	*keyword*	Type of device that produced the image:
	CCD	– for linear detectors like CCDs,
	PHOTO	– for photographic scans.
EXTRACT_MINAREA	*integer*	Minimum number of pixels above threshold for detection.
GAIN	*float*	Gain used for error estimates of CCD magnitudes (e^-/ADU).
MAG_GAMMA	*float*	γ of the emulsion (takes effect only in PHOTO mode).
MAG_ZEROPOINT	*float*	Zero-point offset to be applied to magnitudes.
MEMORY_BUFSIZE	*integer*	Number of scan-lines in the image-buffer. Multiply by 4 times the frame width to get equivalent memory space in bytes.
MEMORY_PIXSTACK	*integer*	Maximum number of pixels that the pixel-stack can contain. Multiply by 16 to get equivalent memory space in bytes.
PARAMETER_NAME	*string*	The name of the file containing the list of parameters that will be computed and put in the catalog for each object.
PHOTOM_APERTURE	*float*	Aperture diameter in pixels (used by MAG_APER).
PHOTOM_KRMIN	*float*	Minimum radius (in sigmas) for MAG_AUTO magnitudes.
PHOTOM_KPAR	*float*	Kron parameter for MAG_AUTO magnitudes.
PHOTOM_KRMAX	*float*	Analysis radius (in sigmas) for MAG_AUTO magnitudes.

PIXEL_SCALE	*float*	Pixel size in arcsec (for surface brightness parameters, FWHM and star/galaxy separation only).
SATUR_LEVEL	*float*	Pixel value above which an object is considered as saturated (affects the saturation flag and FWHM_IMAGE).
SCAN_ISOAPRATIO	*float*	Maximum isophotal-to-aperture object size allowed.
SEEING_FWHM	*float*	FWHM of stellar images in arcsec (only for star/galaxy separation).
STARNNW_NAME	*string*	Name of the file containing the neural-network weights for star/galaxy separation.
THRESHOLD	*float*	Threshold to be used for detection.
THRESHOLD_TYPE	*keyword*	Type of threshold value:
	SIGMA	– number of standard deviations in the background noise,
	MAGNITUDE	– magnitude per sq. arcsec. (interesting for calibrated images).
VERBOSE_TYPE	keyword	How much SExtractor comments its operations:
	QUIET	– run silently,
	NORMAL	– display warnings and limited info concerning the work in progress,
	FULL	– display a more complete information and the principal parameters of all the objects extracted.

3.2 The Parameter file

All the parameters that will be listed in the output catalog for every detection should be given in this file. The format is ASCII, and there must be only one keyword per line. The order in which the parameters will be listed in the catalog are the same as that of the keywords in the parameters file. Comments are allowed, they must begin with a "#". Here is a descriptive list of available parameter keywords [11]:

NUMBER	Identification number.
ISOAREA_PIXEL	Area (in pixels) within the extraction thresholds isophote.
X_IMAGE	X centroid in image coordinates (1.0 = center of the first pixel).
Y_IMAGE	Y centroid in image coordinates (1.0 = center of the first pixel).
X_WORLD	X centroid in world coordinates.
Y_WORLD	Y centroid in world coordinates.
A_IMAGE	2nd order moment along the major axis (image).
B_IMAGE	2nd order moment along the minor axis (image).
A_WORLD	2nd order moment along the major axis (world).
B_WORLD	2nd order moment along the minor axis (world).
THETA_IMAGE	Position angle of the major axis (counter-clockwised, 0.0 = X axis).
THETA_WORLD	Position angle of the major axis (counter-clockwised, 0.0 = X world axis).
MAG_ISO	Isophotal magnitude.
MAG_ISO_ERR	Isophotal magnitude *rms* error.
MAG_ISOCOR	Corrected isophotal magnitude.
MAG_ISOCOR_ERR	Corrected isophotal magnitude *rms* error.
MAG_APER	Fixed-aperture magnitude.
MAG_APER_ERR	Fixed-aperture magnitude *rms* error.

[11] "WORLD" coordinates are computed using informations found in the image FITS header (such as the pixel scale).

MAG_AUTO	Automatic-aperture magnitude.
MAG_AUTO_ERR	Automatic-aperture magnitude *rms* error.
MAG_BEST	= MAG_AUTO if there are no neighbours, or MAG_ISOCOR otherwise
MAG_BEST_ERR	BEST_MAG magnitude *rms* error.
KRON_RADIUS	Extension of the AUTO aperture, in units of A (or B).
FLUX_MAX	Peak surface brightness (ADU).
MU_MAX	Peak surface brightness (mag.arcsec^{-2}).
THRESHOLD	Level of the lowest isophote (ADU).
MU_THRESHOLD	Level of the lowest isophote (mag.arcsec^{-2}).
BACKGROUND	Local background (ADU).
ISOAREA_IMAGE	Area of lowest isophote (pixels).
ISOAREA_WORLD	Area of lowest isophote (arcsec^2).
FWHM_IMAGE	FWHM of profile (from a gaussian fit to the core).
FWHM_WORLD	FWHM of profile (arcsec).
ELONGATION	Isophotal, weighted elongation (equivalent to A_IMAGE/B_IMAGE).
ISOn (n=*0,..,7*)	Isophotal area for subsequent profile-classification (pixels).See Bertin 1994.
CLASS_STAR	"Stellarity index" given by the neural network output: 0.0 = Galaxy and 1.0 = Star.
FLAGS	Extraction flags (see below).

3.3 Flag meaning

In order to be easily readable by most programs, SExtractor 's FLAGS parameter contains, coded in decimal, all the extraction flags as a sum of powers of 2:

1	The object has neighbours, bright and close enough to bias significantly ($\Delta m \approx 0.1$) MAG_AUTO.
2	The object was originally blended with another one,
4	At least one pixel of the object is saturated (or very close to),
8	The object is truncated (too close to an image boundary),
16	Object's aperture data are incomplete or corrupted,
32	Object's isophotal data are incomplete or corrupted,
64	A memory overflow occured during deblending,
128	A memory overflow occured during extraction.

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

3.4 The Convolution file

The convolution file contains the elements of the convolution matrix applied to the data before extraction (the CONVOLVE preference flag must have been set to "Y" before). The format is ASCII. Here is an example of a 3×4 matrix:

```
CONV
# a comment.
1.5  3.46 1.8e-02
4    .01  -58
2e-1 20.2 -2.2
1.   11   5.7
```

The string "CONV" in the first line specifies that the file is a convolution matrix. It is not necessary to enter normalized data: just put the CONVOLVE_NORM flag to Y if you want the sum of weights to be 1.

3.5 "Check-image"

It is often interesting to see what SExtractor is doing with your frame, especially when one tries to tune the extraction parameters to obtain the best result. When the frame is small enough (so that it can fit into memory), one has the possibility to ask SExtractor to produce a "check-image" while it processes the data. This "check-image" is a FITS image containing information about the extraction. There are currently 7 kinds (`CHECKIMAGE_TYPE`) of "check-images":

- `BACKGROUND`: interpolated background; useful to adjust the background parameters like the mesh-size,...
- `-BACKGROUND`: difference between the image and the interpolated background.
- `CONVOLVED`: background-substracted convolved image. Note: `CONVOLVE` must be set to `Y`.
- `OBJECTS`: objects detected above the threshold.
- `SEGMENTATION`: each pixel of the image is assigned a value corresponding to the object it belongs to. Very useful to adjust deblending parameters.
- `APERTURES`: `MAG_AUTO` elliptical and `MAG_APER` circular aperture limits are superimposed to the background-substracted image. `MAG_AUTO` ellipses surrounding objects flagged as "crowded" are dashed.

Fig. 9 gives an idea of what we obtain on a crowded field.

Acknowledgments Many many thanks to S. Arnouts, J.F. Claeskens, E. Copet, E. Deul, P. Fouqué, P. Gitton, J.P. Kneib, V. Lebrun and Ian Smail for testing and comments. Porting to VMS has been made possible thanks to Roderick Johnstone at Cambridge.

References

[1] Beard S.M., McGillivray H.T., Thanisch P.F., 1990, *MNRAS* **247**, 311

[2] Bertin E., 1994, in *Science with Astronomical Near-Infrared Arrays*, eds. Epchtein N., Omont A., Burton B. and Persi P. (Kluwer)

[3] Bertin E., Arnouts S., 1996, A&AS, in press

[4] Bertin E., 1996, *Thèse de Doctorat*, Université Paris VI.

[5] Cotton W.D., Tody D., Pence W.D., 1995, *A&AS* **113**, 159

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

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

[8] Irwin M.J., 1985, *MNRAS* **214**, 575

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

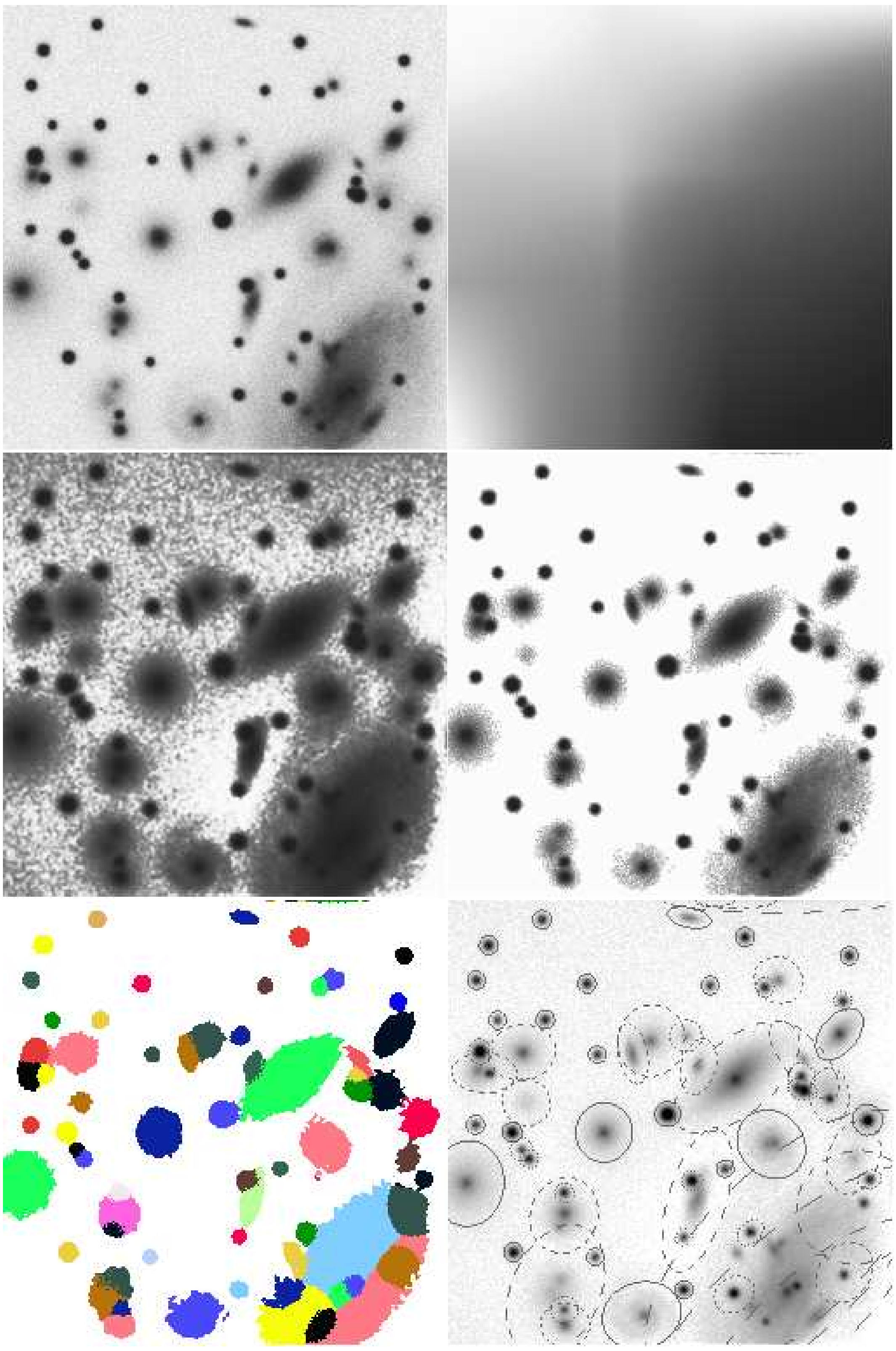


Figure 9: “Check-images” produced by SExtractor , from a 256 × 256 simulated image. From left to right, top to bottom: Original, `BACKGROUND`, `CONVOLVED`, `OBJECTS`, `SEGMENTATION` and `APERTURES` (see text for details). The constrast of the `BACKGROUND` image has been greatly exagerated.

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

[11] Kron R.G., 1980, *ApJS* **43**, 305

[12] Lutz R.K., 1979, *The Computer Journal* **23**, 262

[13] Maddox S.J., Efstathiou G., Sutherland W.J., 1990, *MNRAS* **246**, 433

[14] Moffat A.F.J., 1969, *A&A* **3**, 455

[15] Odewahn S.C., Stockwell E.B., Pennington R.L., Humphreys R.M., Zumach W.A., 1992, *AJ* **103**, 318

A FAQ and Troubleshooting

Here is a short compilation of answers to Frequently Asked Questions, and solutions to some usual problems that a SExtractor user may encounter. Don't hesitate to send me a mail in case of trouble (*bertin@iap.fr*).

Question: *Can I do precise photometry with SExtractor ?*

Answer: the purpose of SExtractor magnitudes is mainly to obtain photometry of faint sources with the lowest error. For bright objects, `MAG_ISOCOR`, `MAG_AUTO` and obviously `MAG_BEST` magnitudes do not permit to go beyond $\approx 1\%$ in photometric accuracy. Moreover, they contain a systematic offset compared to "true" total magnitudes of about 0.05 mag (see § 2.4). Of course the offset vanishes when the frames are calibrated using the same magnitudes. However, only by measuring standard stars with `MAG_APER` aperture magnitudes, one will be able to establish very precise zero-points (down to a few thousandths of magnitude). In that case, the offset within `MAG_ISOCOR`, `MAG_AUTO` and `MAG_BEST` has to be taken into account.

Q: *I cannot make the neural network classification work properly on my digitized Schmidt-plate images.*

A: Unfortunately, the present neural classifier makes the assumption that the detector is linear, and therefore is not suited to work on photographic images, which require a training more specific (see Bertin 1995).

Q: *In my faint galaxy survey, half of the dimmest objects are classified as "stars" (i.e.* `CLASS_STAR`$>$ 0.5*), although I know that most of them are galaxies.*

A: This is "normal": when the SExtractor classifier meets an object too faint to be classifiable, there is roughly 50 % probability to get a stellarity index greater than 0.5. This is a natural consequence of the fact that the classifier doesn't know *a priori* the relative fraction of objects which are stars or galaxies. There is one solution to this problem: one should consider that a faint object might be reliably classified as a point-source only if its stellarity index is greater than say, 0.9, instead of 0.5 (as seen in §2.6, the stellarity index is also a kind of confidence estimate of the classification).

Q: *Some saturated stars are classified as "galaxies" by the neural network, despite correct* `PIXEL_SCALE` *and* `SEEING_FWHM` *parameters in the configuration file.*

A: It's normal that saturated stars may be sometimes misclassified, because the training of the neural network was done with saturation features that may differ from your image ones. As galaxies are generally unsaturated, one can easily identify very bright stars according to their saturation flag (§3.3).

Q: *I get absurd or unaccurate values for* `FWHM_IMAGE` *and/or* `FWHM_WORLD` .

A: `FWHM_IMAGE` results from a fit done only on *unsaturated* pixels of the object: Check if the `SATUR_LEVEL` parameter is set to a proper value.

Q: *Strange values are given for magnitudes, and the deblending does not work well on my CCD frame.*

A: Check if the `DETECTION_TYPE` is not set to `PHOTO` by mistake.

Q: *I don't understand how the "WORLD" parameters are computed.*

A: "WORLD" parameters are computed using astrometric information found in the FITS header of the image file (like position offset, step per pixel, rotation angle). `ISOAREA_WORLD` and `FWHM_WORLD` parameters are a little bit special: if `PIXEL_SCALE` is set to 0 in the config file, they are computed using a pixel size deduced from FITS header informations (**assuming these FITS informations are expressed in degrees**). Otherwise, the `PIXEL_SCALE` value (in arcsec per pixel is used.

Q: *The noise estimate given for the background becomes higher as the mesh size increases.*

A: Perfectly normal if the background is far from beeing uniform (any variation of the background within a mesh can be considered as low-frequency noise).

Q: *On an "*`APERTURES`*" check-image, I get one circle and one ellipse around each object.*

A: It happens because you selected both `MAG_APER` and `MAG_AUTO` (or `MAG_BEST`) in the parameter file. The circle has a diameter equal to `PHOTOM_APERTURE`.

Q: *On an "*`APERTURES`*" check-image, ellipses are flattened in the* `y` *direction for the largest objects, and the* `MAG_AUTO` *are too high (flux lost).*

A: The number of lines in `MEMORY_BUFSIZE` is too small compared to the size of the largest objects in the frame (anyway, such objects are flagged as corrupted from the point of view of automatic magnitudes).

Q: *I can hardly deblend a cluster of small objects, even with a very small* `DEBLEND_MINCONT`.

A: The background mesh size (`BACK_XSIZE` and `BACK_YSIZE`) is too large compared to the size of objects to extract (see §2.1).

Q: *On my SUN workstation, I get a* `bus error` *on a very large object with SExtractor . On some other workstation with not much memory space, I get an error message.*

A: Too much memory is required for deblending: try to decrease `DEBLEND_NTHRESH`. On SUN, a problem with the `malloc()` function sometimes produces a `bus error` in such occasions.

B Information for programmers

SExtractor is completely written in `ANSI C`, and needs no specific library to be compiled. If you're familiar with `C`, you can easily add new functions and parameters to SExtractor . Here is a short guide. For more details, please refer to comments in the source code, or send me a mail (*bertin@iap.fr*)!

B.1 Add a new parameter

How SExtractor is storing information about objects it detects. After detection, Parameters related to the object (ex: magnitude, isophotal area,...) are stored within two kinds of structures:

> The first one concerns data that need information coming from the *pixel stack* or the *image buffer* (see 2.2). This includes total flux, positions, etc... When `CLEAN`ing is on, all these informations are kept temporarily in the *object stack*. The *object* structure is typedef'ed `objstruct` in the include file `types.h`. `objstruct`s can be grouped in *objlists* (typedef'ed `objliststruct`), which can have pointers to *pixel lists*.
>
> The second one contains data that can be processed without pixel information, in a "blind" way. There's no need to store arrays of such a structure (typedef'ed `obj2struct`), as its elements can be computed at any time.

"Pixel" parameters are computed in two functions: `preanalyse()`, for basic parameters needed at all stages of processing, and `analyse()` for more complex ones. In most functions, they are generally called `obj->`*param*. "Blind" parameters are only computed just before being written to the catalog file, in `endobject()`. They are always called `outobj2.`*param*. Many parameters are computed only if they appear in the `.param` file.

Modify the source to compute a new parameter Let's suppose you want SExtractor to compute a new parameter, "`myparam`", and write it to the catalog:

- First, insert `myparam` in the `objstruct` (or `obj2struct` type definition in `types.h`. Valid types are: `BYTE`, `short`, `long`, `float` and `double`.
- Add a parameter name (less than 16 characters long), the right "T_type", a pointer (`&outobj.myparam` or `&outobj2.myparam`) and a format to `param[]` in `param.h`.
- Insert the portion of code computing the parameter, or a call to a function, in `analyse()` or `endobject()` (in `analyse.c`).
- If there are dependencies with other parameters, add them in `updateparamflags()` (`catout.c`).
- If `myparam` is a "pixel" parameter, you can define its behaviour during the "`CLEAN`ing" process in `mergeobject()` (`clean.c`).

That's all!

B.2 Add a new option

Adding a new option to SExtractor is also quite easy; let's see how to add the option "`myoption`":

- First insert `myoption` in the `prefstruct` type definition in `types.h`. Valid types are: `int`, `double`, `char strings` or `enum`.
- Add an option name, a "P_type", the pointer (`&prefs.myoption`), imin and imax limits (if it's an `int`), dmin and dmax limits (if it's a `double`) and a list of keywords (if it's an `enum`) to `key[]` in `prefs.h`. None of the keywords, or the option name, should have more than 15 characters.

- Add a FITS keyword (`SEXMYOPT` for instance) with no more than 8 characters, a "T_type", the pointer (`&prefs.myoption`), a format, and a "H_type" to `hparam[]` in `param.h`. Insert the corresponding line in the FITS-header template of `initfitscat` (`catout.c`).
- If necessary, edit `useprefs()` in `prefs.c` to take into account informations taken from the image.

You can now use the `prefs.myoption` variable to do whatever you want in SExtractor from the configuration file.