SourceXtractor++ readme

February 2021

A. Álvarez-Ayllón[1], E. Bertin[2], P. Dubath[1], R. Gavazzi[2] W. Hartley[1], D. Hernandez Lang[3], M. Kümmel[3] and M. Schefer[1] [1]Department of Astronomy, University of Geneva, Chemin d'Écogia 16, CH-1290, Versoix, Switzerland [2]Sorbonne Université, CNRS, UMR 7095, Institut d'Astrophysique de Paris, 98 bis bd Arago, F-75014 Paris, France; bertin@iap.fr [3]LMU Faculty of Physics, Scheinerstr. 1, 81679 München, Germany

Introduction

The following document outlines the files, data structure and computational performance of the `SourceXtractor++` submission to the morphology challenge.

`SourceXtractor++` [2, 3] is a ground-up re-write of the widely used `SExtractor2` software [1], written in `C++` with a strong focus on extensibility and model fitting photometry. The first official release was in October 2019, the software is under active development and the results submitted to the challenge represent a snapshot in this process. Each `SourceXtractor++` run comprises two stages: detection and measurement, each controlled via a pair of configuration files. The detection and segmentation stage follows broadly the same procedure as used in the "old" `SExtractor2`. A detection image is filtered with a convolution kernel (most frequently a Gaussian) and groups of pixels with signal-to-noise ratio significantly above the background are identified and formed into detections. Blended sources are then partitioned, and within the overall source list groups of objects that require simultaneous measurement are constructed.

Measurement in one or several bands controlled via a python configuration file with flexible model fitting at its core. Various components, point source, exponential disk, free-Sersic etc., can be used singly or in combination. With such a flexible scheme, it is important to provide reasonable priors to the fitting engine in order to cover sensible fits. The native form of the prior is in the form of a Gaussian, with other prior shapes available through a change of variable. The priors adopted for our runs on the data challenge images are detailed below.

The output from our `SourceXtractor++` runs were finally cross-matched via position (x,y) to the input position lists with a tolerance of 3 pixels, using `STILTS`. We find the best match for each object in the input positions (all) lists, and fill the columns for objects that we do not have a match for (i.e. were not detected in our runs) with values -9999.

Image pre-processing

In preparation for model fitting with `SourceXtractor++`, we made a few modifications to the data provided by the challenge team, in order to treat them the same way we would real data. In particular, the pixel manipulations that were needed to bring the LSST and NIR bands onto the VIS pixel grid, are really detrimental and ought to be avoided as much as possible. The induced noise correlations, if not properly dealt with, would give too much weight to the ground and NIR bands in the multiband run for instance and would yield misleading over uncertainties.

- The LSST-like images were resampled back to their original pixel scale (0.2 arcsec)
- The NIR images were resampled back to their original pixel scale (0.3 arcsec).
- We extracted oversampled PSF images using the `PSFEx` software.

With this pre-processing we could use the provided PSF images images in the given sampling and have oversampled PSF images in all bands, which we consider as an essential condition to deliver satisfactory results. The PSF images that we supplied by the challenge organisers turned out being sufficiently well oversampled ($\times 2$ for LSST-like data, and $\times 3$ for the Euclid NIR bands). We however found that the PSF model we built for the VIS band could yield substantially better results for morphometry and stellar photometry with a $\times 1/0.45$ oversampling factor.

Object detection

Source detection parameters necessitate a compromise between completeness of the true object list and the number of spurious objects extracted or deblended. Over-extraction of sources impacts the performance of the run time required, and may also reduce the accuracy of morphological measurements if objects are over-deblended. In tuning the parameters we aimed for good performance overall, and did not demand 100% completeness of the 5σ source list. As a result, some of the 5σ sources do not have parameters recovered. We do, however, detect and measure a fraction of the lower signal-to-noise sources.

Adopted priors

The flexible model fitting approach of `SourceXtractor++` requires a set of priors in order to obtain reliable results for low signal to noise sources. Table 1 describes the priors adopted for the four set-ups used: single-Sersic, bulge+disk, Sersic+disk and additional multi-band priors. For the data set named "Realistic Morphologies" we have applied the same priors as in the single-Sersic setup. Even though source ellipticity is treated internally by `SourceXtractor++`with the standard axis ratio q and orientation angle (ccw wrt to x axis), we found it useful to express the priors that are needed for these two degrees of freedom in terms of the e_1, e_2 cartesian ellipticity parameters that usually enter weak lensing studies. More specifically, in complex notations, $e = \frac{1-q}{1+q}\exp(2\imath\varphi) \equiv e_1 + \imath e_2$. Hence, we just set a Normal prior on both e_1 and e_2, centered on 0 and of width 0.3. We checked that this was in broad agreement with the population distribution of axis ratio in the Single Sersic channel of Field 4 for which ground truth is provided. A similar prior was built for the Sersic index as well as for the (log) effective radius. Those priors are listed in Table 1.

In addition, in multi-component models, the bulge and disk share a common position angle while axis ratios do not seem to be correlated. Therefore, the use of (e_1, e_2) ellipticities proves much less useful, and we instead go back to the default q and φ variables. In the multi-band run, a single fit across all bands was made for effective radii, ellipticity and position angle. The amplitudes of the bulge and disk components were left free to vary. Here again, the fitting procedure prefers the definition of a set of total magnitudes and a set of bulge-to-total ratios (with their own associated priors), instead of using the flux of each component in each band as a set (details of the band-dependent priors can be preovided if required).

Table 1: Priors adopted during production runs.

Quantity	Prior expression	parameters
	Single Sersic	
Sersic index, n	$\ln((n-0.25)/(30-n))$	$\mathcal{N}(-3.9, 0.6)$
Effective radius, r_e	$\ln(\mathrm{r}_e)$	$\mathcal{N}(1.0, 0.4)$
e1	e1	$\mathcal{N}(0, 0.3)$
e2	e2	$\mathcal{N}(0, 0.3)$
	Realistic Morphology	
Sersic index, n	$\ln((n-0.25)/(30-n))$	$\mathcal{N}(-3.9, 0.6)$
Effective radius, r_e	$\ln(\mathrm{r}_e)$	$\mathcal{N}(1.0, 0.4)$
e1	e1	$\mathcal{N}(0, 0.3)$
e2	e2	$\mathcal{N}(0, 0.3)$
	Bulge + Disk	
Disk effective radius, $r_{e,d}$	$\ln(\mathrm{r}_{e,d})$	$\mathcal{N}(0.7, 0.5)$
Bulge radius, $r_{e,b}$	$\log_{10}(r_{e,b}/r_{e,d}^2)$	$\mathcal{N}(-1.0, 0.5)$
Bulge fraction, bf	$\ln((bf+0.0001)/(1.0001-bf))$	$\mathcal{N}(-2.2, 1.7)$
	Sersic + Disk	
Sersic index, n	n	$\mathcal{N}(4.0, 0.2)$
Disk effective radius, $r_{e,d}$	$\ln(\mathrm{r}_{e,d})$	$\mathcal{N}(0.7, 0.5)$
Bulge radius, $r_{e,b}$	$\log_{10}(r_{e,b}/r_{e,d}^2)$	$\mathcal{N}(-1.0, 0.5)$
Bulge fraction, bf	$\ln((bf+0.0001)/(1.0001-bf))$	$\mathcal{N}(-2.2, 1.7)$
	Multiband	
Disk effective radius, $r_{e,d}$	$\ln(\mathrm{r}_{e,d})$	$\mathcal{N}(0.33, 0.25)$
Bulge radius, $r_{e,b}$	$\log_{10}(r_{e,b}/r_{e,d}^{1.14})$	$\mathcal{N}(-1.2, 0.4)$
Bulge fraction, bf	band dependent	

File naming convention

Submission files are grouped into folders, with the name corresponding to the section of the data challenge. For the multiband case, a single catalogue is produced as most parameters are shared between bands: only the total magnitudes and bulge-to-total ratios vary.

Table 2: Submission files naming convention. *field* is to be replaced by an integer, corresponding to the challenge image files.

Run type	File name pattern
Single Sersic	ssersic_cat_{*field*}.fits
Bulge (n=4) + disk	dsersic_{*field*}.fits
Sersic + disk	dsersic_var_{*field*}.fits
Multiband	dsersic_multiband_{*field*}.fits
Realistic morphologies	ssersic_realistic_{*field*}.fits

Column names

Single Sersic and realistic morphologies

All single-Sersic runs contain the following table columns. The information is split into three tables: original challenge list columns (Table 3), challenge entry columns (Table 4) and additional columns (Table 5).

Table 3: Columns replicated from input source lists provided

Catalogue Column	Units	Description
ID	-	Object identifier
X	pixels	input x-coordinate
Y	pixels	input y-coordinate

Table 4: Challenge entry columns for single-Sersic runs.

Catalogue Column	Units	Description
pixel_centroid_x	pixels	x-centroid
pixel_centroid_y	pixels	y-centroid
x_fit	pixels	x-coordinate of model fit
x_fit_err	pixels	x-coordinate of model fit uncertainty
y_fit	pixels	y-coordinate of model fit
y_fit_err	pixels	y-coordinate of model fit uncertainty
sersic	-	Sersic index
sersic_err	-	Sersic index uncertainty
radius	pixels	Half-light semi-major axis
radius_err	pixels	Half-light semi-major axis uncertainty
mag	AB mag	Total magnitude
mag_err	AB mag	Total magnitude uncertainty
ratio	-	Axis ratio
ratio_err	-	Axis ratio uncertainty
angle_deg	degrees	Position angle
angle_deg_err	degrees	Position angle uncertainty

Table 5: Additional output colomns in our entry files. * corresponds to columns that are present only in the single-Sersic files.

Catalogue Column	Units	Description
source_id	-	SourceX++ identifier
world_centroid_alpha	deg.	Right ascension
world_centroid_delta	deg.	Declinaiton
source_flags	-	Source flag information
fmf_reduced_chi_2	-	Reduced χ^2 of flexible model fit
fmf_iterations	-	Number of iterations taken to fit
fmf_flags	-	Flag information for model fit
e1*	-	x-component of ellipticity
e1_err*	-	Uncertainty in x-component of ellipticity
e2*	-	y-component of ellipticity
e2_err*	-	Uncertainty in y-component of ellipticity
emod*	-	Ellipticity
emod_err*	-	Ellipticity uncertainty

Double Sersic

Table 6: Challenge entry columns for Bulge (n=4) + Disk (n=1) runs. * indicates columns that are dropped in the multiband case, in favour of the band-specific columns described in Table 8.

Catalogue Column	Units	Description
pixel_centroid_x	pixels	x-centroid
pixel_centroid_y	pixels	y-centroid
x_fit	pixels	x-coordinate of model fit
x_fit_err	pixels	x-coordinate of model fit uncertainty
y_fit	pixels	y-coordinate of model fit
y_fit_err	pixels	y-coordinate of model fit uncertainty
mag*	AB mag	Total magnitude
mag_err*	AB mag	Total magnitude uncertainty
bt*	-	Bulge to total flux ratio
bt_err*	-	Bulge to total flux ratio uncertainty
disk_effR_px	pixels	Disk half-light semi-major axis
disk_effR_px_err	pixels	Disk half-light semi-major axis uncertainty
bulge_effR_px	pixels	Bulge half-light semi-major axis
bulge_effR_px_err	pixels	Bulge half-light semi-major axis uncertainty
disk_axr	-	Disk axis ratio
disk_axr_err	-	Disk axis ratio uncertainty
bulge_axr	-	Bulge axis ratio
bulge_axr_err	-	Bulge axis ratio uncertainty
angle_deg	degrees	Position angle
angle_deg_err	degrees	Position angle uncertainty
flux_disk*	μJy	Disk flux
flux_disk_err*	μJy	Disk flux uncertainty
flux_bulge*	μJy	Bulge flux
flux_bulge_err*	μJy	Bulge flux uncertainty
flux_tot*	μJy	Total flux
flux_tot_err*	μJy	Total flux uncertainty

Table 7: Challenge entry columns for Sersic (n=free) + Disk (n=1) runs.

Catalogue Column	Units	Description
pixel_centroid_x	pixels	x-centroid
pixel_centroid_y	pixels	y-centroid
x_fit	pixels	x-coordinate of model fit
x_fit_err	pixels	x-coordinate of model fit uncertainty
y_fit	pixels	y-coordinate of model fit
y_fit_err	pixels	y-coordinate of model fit uncertainty
mag	AB mag	Total magnitude
mag_err	AB mag	Total magnitude uncertainty
st	-	Sersic to total flux ratio
st_err	-	Sersic to total flux ratio uncertainty
disk_effR_px	pixels	Disk half-light semi-major axis
disk_effR_px_err	pixels	Disk half-light semi-major axis uncertainty
sersic_effR_px	pixels	Bulge half-light semi-major axis
sersic_effR_px_err	pixels	Bulge half-light semi-major axis uncertainty
disk_axr	-	Disk axis ratio
disk_axr_err	-	Disk axis ratio uncertainty
sersic_axr	-	Bulge axis ratio
sersic_axr_err	-	Bulge axis ratio uncertainty
sersic	-	Bulge sersic index
sersic_err	-	Bulge sersic index uncertainty
angle_deg	degrees	Position angle
angle_deg_err	degrees	Position angle uncertainty
flux_disk	μJy	Disk flux
flux_disk_err	μJy	Disk flux uncertainty
flux_sersic	μJy	Bulge flux
flux_sersic_err	μJy	Bulge flux uncertainty
flux_tot	μJy	Total flux
flux_tot_err	μJy	Total flux uncertainty

Table 8: Additional columns for multiband runs. {*band*} refers to the following: vis, u_lsst, g_lsst, r_lsst, i_lsst, z_lsst, y_nir, j_nir and h_nir.

Catalogue Column	Units	Description
mag_{*band*}	AB mag	Total flux
mag_{*band*}_err	AB mag	Total flux uncertainty
bt_{*band*}	-	Bulge to total ratio
bt_{*band*}_err	-	Bulge to total ratio uncertainty

System architecture and runtime

Single-band runs (single Sersic, real morphology, bulge + disk and Sersic + disk) were performed on small cluster at LMU Munich. The cluster has a mix of server machine with 36 or 56 cores and ~ 2 GB RAM per core. It is managed with the `slurm` Workload Manager. For the Morphology Challenge we have run `SourceXtractor++` on the large, 56 core machines. Multi-band runs were performed on a local server at IAP with 24 AMD EPYC 7402P cores (ignoring hyperthreading) running at 3.3 GHz, using an improved version of the code with increased parallelisation efficiency.

Table 9: Number of processors, available memory and run-time for each run type. Values are for a whole image (25k x 25k pixels) unless otherwise stated.

Run type	Code version	Description	Run time per field
Single Sersic	0.12	56 cores, 110GB RAM	50 mins
Bulge (n=4) + disk	0.12	56 cores, 110GB RAM	1 hour 45 mins
Sersic + disk	0.12	56 cores, 110GB RAM	2 hours
Multiband	0.13	24 cores, 64GB RAM	45 hours $(25\text{k})^2 + 5 \times (12.5\text{k})^2 + 3 \times (8.3\text{k})^2$

References

[1] E. Bertin and S. Arnouts. SExtractor: Software for source extraction. , 117:393–404, June 1996.

[2] E. Bertin, M. Schefer, N. Apostolakos, A. Álvarez-Ayllón, M. Kümmel, and P. Dubath. The SourceXtractor++ software. In TBD, editor, *ADASS XXIX*, volume TBD of *ASP Conf. Ser.*, page TBD, San Francisco, 2020. ASP.

[3] M. Kümmel, M. Schefer, N. Apostolakos, A. Álvarez-Ayllón, E. Bertin, and P. Dubath. Working with the SourceXtractor++ software. In TBD, editor, *ADASS XXIX*, volume TBD of *ASP Conf. Ser.*, page TBD, San Francisco, 2020. ASP.