StarFinder: code description and operation

1 Purpose

StarFinder is an IDL program for stellar fields analysis. It is able to detect stars and determine their position and relative flux. It has been designed basically for Adaptive Optics (AO) data, keeping in mind the following aspects:

- a typical AO observation is characterized by a 'complex' PSF, with a sharp peak, one ore more fragmented diffraction rings and an extended halo;
- AO observations are well-sampled, i.e. there are at least two resolution elements (pixels) in the PSF FWHM.

The complexity of the PSF might lead to false detections and affect the photometry of faint objects. On the other hand the good sampling of a typical AO observation is an important advantage that should be exploited.

The approach implemented in *StarFinder* is to analyze the stellar field with a digital image of the PSF, without any analytic approximation: in practice the PSF is used as a template, which can be scaled and translated by sub-pixel offsets. Accurate positioning of the PSF onto a given star is accomplished by means of interpolation techniques; different methods have been tested, but the most promising seems to be the so called *cubic convolution interpolation* (Park and Schowengerdt, 1983, 'Image Reconstruction by Parametric Cubic Convolution', Computer Vision, Graphics & Image Processing 23, 256), which approaches very closely the optimal *sinc* interpolation for well-sampled data.

StarFinder is not a general purpose algorithm for objects recognition, like *SExtractor*: it should be intended as a tool to obtain high precision astrometry and photometry of point sources in adequately sampled high-resolution images of crowded fields, i.e. targets containing stars superposed on a smooth background, originated by unresolved sources, nebulosity, etc. In practice it can be applied not only to high-Strehl AO observations, but also to low-Strehl images (with or without AO) and HST fields, provided the sampling is satisfactory. We are also evaluating the photometric accuracy on undersampled data.

An important topic in AO observations is represented by the anisoplanatic effects arising in wide-field imaging: different strategies are under development to handle this problem, related to stellar fields photometry. The current release of the code assumes a constant PSF.

2 General description

The basic analysis procedure consists of 3 phases:

1. detection of presumed stars above a pre-fixed threshold in the background-removed image;
2. check and analysis of presumed stars, sorted by decreasing intensity;
3. re-fitting of detected stars.

The procedure above may be iterated: the previously detected stars are temporarily removed from the image to search for lost objects. Then the analysis proceeds on the original frame. This iteration is very useful to detect close binaries and components of crowded groups, down to separations comparable to the PSF FWHM. At the end of the last iteration, a de-blending strategy may be applied optionally: if the extension of a given detected star results to be significantly larger than the PSF area, the object is assumed to be a blend and de-blending is attempted by iteratively searching for statistically significant residuals and subsequent fitting. One may state that the iteration of the basic analysis procedure (steps 1-2-3) is to detect isolated sources and 'resolved' stars, whereas the final de-blending procedure is to detect even closer components, at separations somewhat smaller than the PSF FWHM. As a rule of thumb, two iterations of the basic analysis procedure with no final de-blending are sufficient to analyze crowded fields, like the simulated image included in this package.

The analysis of a given object (step 2 above) includes the following operations:

- re-identification, after subtraction of already known stars, to reject spurious detections associated to PSF features of brighter sources
- correlation check, to measure the similarity of the object with the PSF; a correlation threshold is fixed to distinguish between stellar-like objects and spurious detections (e.g. noise spikes)
- fitting, to determine position and flux, taking into account the contribution of neighboring brighter stars and of the local background, approximated with a slanting plane
- updating of a stellar field model, which contains a replica of the PSF for each detected star; it is basically used to keep track of the contribution of bright sources when analyzing fainter ones.

3 Code implementation

StarFinder has been provided with a collection of auxiliary routines for PSF extraction, noise estimation, basic image processing, in order to be able to fully analyze a stellar field and produce an output list of stars. The input image is supposed to be just corrected for instrumental effects and calibrated.
A widget-based Graphical User Interface has been created, which allows one to use *StarFinder* without knowing IDL. The main widget appearing on the screen (called *XStarFinder*) is nothing more than an interface to call different widget-based applications, in order to perform various operations on the stellar field image. The main widget defines a memory area to store some global variables, whose name and meaning is listed below:
image: stellar field;
PSF: image of the Point Spread Function;
noise: array of noise standard deviation for each image pixel;
background: array containing an estimate of the background emission;
detected stars: synthetic image, containing one replica of the PSF for each detected star in field;
synthetic field: sum of **background** and **detected stars**.
The global variables are maintained and modified in the course of the current session. As one of these variables is processed by some application, its previous value is overwritten: if the user wishes to keep track of the values assumed by the variable in the course of the session, he/she should save the variable or the entire session whenever necessary.
All the global data are basically divided into two groups: those associated to the image and those associated to the PSF. Whenever the user loads a new image, for instance, the data associated to it are erased.
Every secondary widget application has a set of default parameters, which can be modified interactively. The user-defined values become the new default values of the widget, until the user quits *XStarFinder* without saving the current session. This behavior may be helpful when analyzing two images of the same field: it will be easier to analyze the two frames in the same way. Of course if the images have been taken at different wavelengths, all the 'structural' parameters (e.g. those related to the PSF FWHM, as the box size for PSF extraction) will have to be scaled properly in the second run.
The secondary widgets called by *XStarFinder* prevent access to both the IDL command line and to all the other open widgets, until they are dismissed. The arrangement and layering of the widgets on the computer screen depends on the platform being used. On some systems (e.g. Windows) the active widget floats on top of all the others, which cannot be either moved or highlighted. On other systems it is possible to move or highlight non-active widgets: to restore the original arrangement and layering, the user may try to iconize the *XStarFinder* widget and then restore it again.
IDL users might wish to run the *StarFinder* routines directly from the IDL command line. Complete documentation on each module can be retrieved with the IDL procedure *doc_library*: to display the documentation about the module *stack_median*, for instance, enter the following command at the IDL prompt:
`IDL> doc_library, 'stack_median'.`
A list of routines in the *StarFinder* library is contained in the ASCII file 'list_of_modules.txt'.
The more interesting programs to analyze a stellar field are:
circ_mask: apply a circular mask to a 2D array, for instance the PSF
click_on_max: select local maxima by mouse click
compare_lists: find coincidences between two sets of points on a plane
crosses: mark interesting points in an image with crosses
display_image: display an image according to a set of options
fitstars: multi-component PSF-fitting of one or more stars
fwhm: compute the FWHM of a peak or stellar image
gauss_noise_std: compute the standard deviation of normally distributed noise
halo_smooth: smooth the halo of the PSF by means of a variable box size median filtering technique
image_background: estimate the background emission
image_core: extract the component connected to a specified starting position and above a pre-fixed threshold
image_model: create a synthetic image given by a sum of shifted scaled replicas of the PSF
match_coord: given two sets of coordinates on a plane, representing the same points in reciprocally translated and rotated reference frames, map one of them onto the other
psf_extract: estimate the PSF by combination of a set of stars
read_float_data: read an ASCII file containing a set of data on columns made of a known number of elements
repair_saturated: repair saturated stars
replace_pix: replace known bad pixels by local median
search_objects: search significant intensity enhancements
starfinder: stars detection, astrometry and photometry

4 Installation

StarFinder needs no particular installation procedure. If you have received the code as a compressed archive, decompress it and put its content in a new directory called 'starfinder'; if you have received the code in a CD-ROM, just copy the 'starfinder' directory from the CD to your computer. The second (and last) step is to tell IDL where the new 'starfinder' directory is located modifying the IDL system variable !Path. This can be done very easily within the IDL Development Environment, by means of the 'Preferences' sub-menu in the 'File' menu. Otherwise the path can be modified from the IDL command line: the way to do it is described in the IDL Online Help, under the topic !Path. Once the IDL session is open and the path is set, it is sufficient to enter the following command at the IDL prompt:

```
IDL> xstarfinder
```

The *XStarFinder* widget will appear on the screen.
If you experience any trouble with color images, remember to set up your monitor in 256 color mode.
The code uses the routines included in the standard IDL distribution and some modules of the 'astron' library, which can be retrieved from the WEB site 'http://idlastro.gsfc.nasa.gov/homepage.html'. You can also find the required 'astron' routines in the 'astron_for_starfinder' archive included in the 'starfinder' directory. The 'astron' library has been recently updated, though the November 1998 version has been used here.

5 How to analyze a stellar field

5.1 Introduction

This section is a kind of tutorial and describes how to analyze the simulated stellar field included in the released package. Complete documentation on supported facilities and operations can be found in the on-line help pages.
The simulated image (Fig. 1 left) contains 1000 stars at random positions, with magnitudes extracted from a realistic luminosity function with a range of 8 magnitudes. Each star is represented by a shifted scaled replica of a typical high-Strehl Adaptive Optics PSF (Fig. 1 center). The field has been contaminated with background nebulosity (Fig. 1 right), having a total flux equal to the total flux of the stellar sources, and additional noise, given by the combination of gaussian and photon noise.
The data concerning the synthetic field are contained in the following files included in the package: 'synfield.fits' (stellar field), 'psf.fits' (PSF), 'background.fits' (background nebulosity), 'noise.fits' (array of noise standard deviation for each pixel in the image), 'stars.txt' (ASCII file including positions and fluxes of the point souces) and 'psfstars.txt' (ASCII file including positions of the stars for PSF estimation).

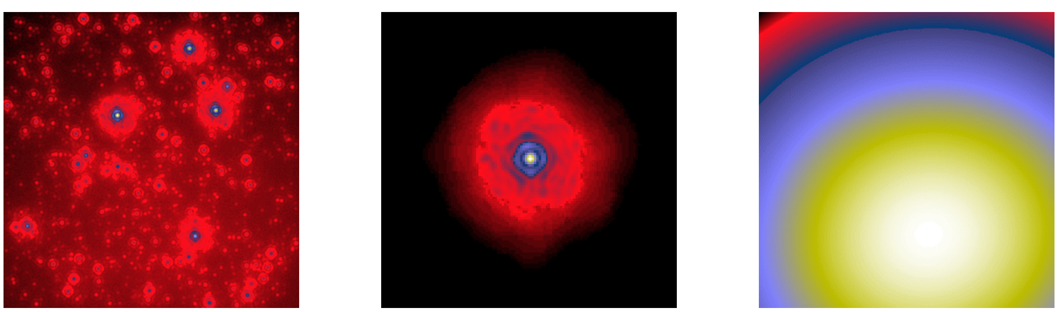

Figure 1. Left: simulated stellar field (square root display). Center: PSF used to generate the field (square root display). Right: background nebulosity added to the field (linear display).

5.2 Image loading and visualization

After starting *XStarFinder* (Sect. 4), click on the 'Image' pull-down menu, select the 'Load' sub-menu and choose the 'Image' option (this sequence of operations will be referred simply as 'Image/Load/Image'). Select a valid file name in the dialog window (in this case search for the 'starfinder' directory and select the file 'synfield.fits'). The loaded image is automatically displayed in the graphic area of the *XStarFinder* widget. You might wish to change the display options: select 'Display/Options' and modify the visualization parameters of the currently displayed image by means of the *XDisplayOpt* widget application. We suggest to choose a minimum and maximum intensity of 250 and 1e5 respectively (this choice will only affect the display, of course!), a square root stretch and the STERN SPECIAL Color table. After entering these parameters click on 'Apply options'. Dismiss *XDisplayOpt* pushing the 'Exit' button.

5.3 Bad pixels repair

The simulated field you have just loaded has no bad pixels, even though your actual data might have some. Bad pixels (either 'dead' or 'hot') should always be repaired because they might lead to false detections (especially under poor sampling condition), loss of significant sources, degradation of astrometric and photometric accuracy. Bad pixels at known positions may be corrected with the 'Image/Bad Pixels' task which invokes the *XReplace_Pix* widget. First of all you have to read the bad pixels positions, which must be stored in a FITS-format file as a binary mask defined as follows:

$$b(x, y) = \begin{cases} 1, \text{ if pixel } (x, y) \text{ is 'good'} \\ 0, \text{ if pixel } (x, y) \text{ is 'bad'} \end{cases}$$

Read the bad pixels mask with the 'Read bad pixels' button; then click on 'Replace'. A dialog appears asking you to enter the box size for the local median computation (the bad pixel will be replaced with this median): try with the default box size first. After replacing the known bad pixels, the program displays automatically the repaired image with the current display options.

5.4 Noise estimation

An estimate of the noise standard deviation is useful to define the detection threshold (Sect. 2) and to compute formal errors on the estimated photometry and astrometry.

In principle it is possible to derive the correct noise statistic knowing the photon and detector noise standard deviations. If you have done this and you have created a FITS file containing an array of noise standard deviation with the same size as the science image, then you can load it with the 'Image/Noise/Load' task. In this tutorial you can try this options loading the noise array in the 'noise.fits' file released with the code.

Another possibility is to estimate the noise with the 'Image/Noise/Compute' task, which invokes the *XNoise* widget. This widget application can be essentially used in one of two ways:

1. estimate the correct noise statistic based on the knowledge of parameters like read-out-noise, dark current level, detector gain, etc.;
2. estimate the mean noise level associated to the diffuse nebulosity present in the data, which should be the major noise source for all but the brightest sources.

The first strategy requires the knowledge of additional parameters which are not always available; in practice you have to enter the parameters related to the normally distributed noise (see the online help for more details) and to the photon noise, then click on the 'Compute' button.

In this tutorial we will apply the second strategy, which is the default. Leaving the default options unchanged, just click on the 'Compute' button; a dialog message will appear, to inform you that another widget, called *XNoise_StDev,* is being called to estimate the noise (by means of an histogram fitting technique). A complete description of the parameters of *XNoise_StDev* may be found in the online help; for the moment, we use the default values. Click on the 'Processing…' button and after a while a dialog message should appear, giving you the result (mode and half-width of the histogram, the latter value representing the noise standard deviation). Of course you need not remember these numbers, because everything is retained in memory by the program. If you wish to check the quality of the fit to the histogram, click on the 'Plot histogram' button to invoke the *XPlot* widget. In the third square from the top, called 'Type', select the 'Steps' options under the column 'Main plot', then click on the 'Plot' button. The histogram will be plotted as a continuous step-like line and the best fit model, given by a gaussian + constant term in this case, will be over-plotted as a dashed line. As you see, the plot has been done on the graphic window of the main widget, which might be hidden by the secondary widget applications (*XNoise, XNoise_StDev* and *XPlot*). On some platforms it is possible to move the widgets away from the graphic area; in other cases, unfortunately, the only solution is to close all the widgets, returning back to the *XStarFinder* level, run again the *XNoise* task, positioning it *out* of the graphic window area, and then call the other applications in sequence. After examining the plot, you may notice that the fit can be improved. For this purpose exit *XPlot*, select the option 'gaussian + quadratic' in the 'Histogram fitting' square of the *XNoise_StDev* widget and hit again the 'Processing…' button. Then repeat the plot: as you see now, the fit has slightly improved. With some additional effort, you can improve also the aspect of the plot itself, save it on a GIF file and obtain something like Fig. 2.

Now you can dismiss all the invoked widgets and return back to the *XStarFinder* level. The required noise information is automatically retained in the memory area associated to the image data (Sect. 3). If you have plotted the histogram, you might wish to restore the image display in the graphic window: you can do it with 'Display/Select data/Image'.

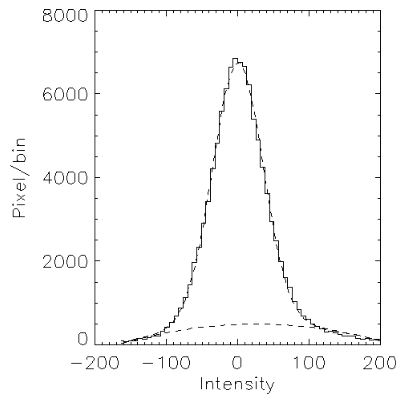

Figure 2. Plot of data histogram (steps) and best fit model (dashed line), given by a gaussian on a quadratic term.

5.5 PSF extraction

If you have an accurate estimate of the PSF on a FITS file, you can load it with the 'PSF/Load…' task. In this tutorial we want to extract the PSF from the image. We do it with the 'PSF/Extract from image' task, which invokes the *XPsf_Extract* widget. The PSF is formed essentially as the median of a set of stars you are requested to select.

The only parameter you have to set for the moment in *XPsf_Extract* is the size of the array where the PSF will be stored: in this case select 128. Then click on the 'Processing…' button. A dialog message invites you to select the stars to be extracted and superposed to form the PSF estimate; these stars must be selected with a click of the left button of your mouse (if you are left-handed please replace the word 'left' in the message with 'right'). We suggest to select the four brightest stars in the image (Fig. 3).

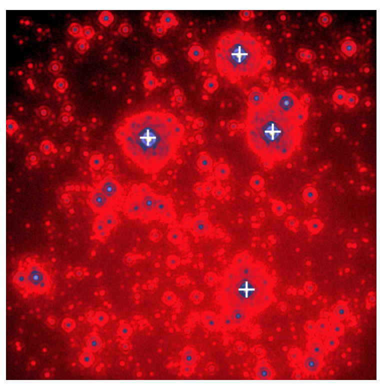

Figure 3. The crosses indicate the stars for PSF extraction.

After selecting the PSF stars, click anywhere in the graphic area with the right button (left if you are left-handed) to exit the selection procedure. What happens now? All the previously selected stars are displayed one by one and you are requested to confirm your previous selection (in general you might have selected an undesired candidate source, a binary for instance). After examining the stars, the program asks you if you wish to select the most contaminating sources around the PSF stars, in order to subtract them. This step might be useful when the PSF stars are only few and the median superposition is not able to get rid of the secondary sources. To see how it works, answer 'Yes' to the previous question: the PSF stars are displayed again one by one and you have to click on the secondary sources with the left button of your mouse, pushing the right button to examine the next PSF star. The secondary sources to be selected in this case are indicated in Fig. 4. In general only the most contaminating sources (i.e. quite bright and quite close to the central star) should be selected. The secondary sources can be recognized by the presence of the diffraction ring. If you think this operation is too boring or too dangerous, you can skip it, even though the final PSF estimate might be slightly worse, especially if there are not many PSF stars to superpose.

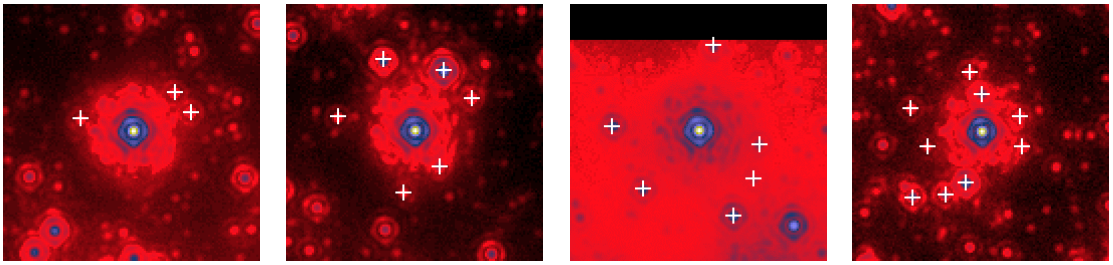

Figure 4. Secondary sources to be selected and subtracted around the candidate PSF stars.

After selecting all the secondary sources, the program starts estimating the PSF and awards you with another dialog message when the computation is done. Now click on the 'Display PSF' button in the *XPsf_Extract* widget to display the retrieved PSF. You can modify the display options with the 'Display Options' task, which invokes the *XDisplayOpt* widget you have already seen in Sect. 5.2. We suggest to enter a minimum and maximum intensity value of 0 and 1 respectively, square root stretch and STERN SPECIAL color table. Notice that the PSF estimate is still contaminated by residual features. After dismissing *XDisplayOpt,* exit *XPsf_Extract*. You are requested to save the coordinates of the PSF stars: these coordinates might be useful when comparing lists of stars (Sect. 5.8). In this case you may find the coordinates of the PSF stars in the file 'psfstars.txt' included in the released package. However, if you wish to save your own PSF stars, answer 'Yes' to the previous question and provide a file name (e.g. 'psf_stars') without any extension (the extension '.txt' will be automatically appended to the file name).
Display now the extracted PSF in the *XStarFinder* graphic area with 'Display/Select data/PSF'. How can we discard the residual features in the PSF array? First of all run the 'PSF/Post process/Support' task, invoking the *XImage_Support* widget, which allows you to operate on the support of the retrieved PSF. Select 'Yes' in the 'Principal component extraction' square, leaving the default value of 0 in the 'Threshold for principal component extraction' field and click on the 'Processing…' button: you may notice that some of the stuff at the frame edge has disappeared. You can try different thresholds, because the original PSF array is left unchanged until you exit *XImage_Support*: as a rule of thumb you should choose the lowest threshold in order to avoid cutting the PSF halo. You can also apply a circular mask to the retrieved PSF: select 'Yes' in the 'Circular mask' square, enter a value of 40 in the 'Circular mask radius' field and hit again the 'Processing…' button, to apply both the options together (masking and principal component extraction). Then exit *XImage_Support*.
The last step is PSF normalization. Actually the PSF retrieved with *XPsf_Extract* was normalized to a total flux of 1, but the post-processing operations might have altered the normalization. To restore it, use 'PSF/Normalize'.
One last comment on PSF extraction. *XPsf_Extract* is able to repair the core of saturated stars, provided the saturation is restricted to the central spike of the PSF (of course you have not to worry about saturation effects now, because all the stars are unsaturated in this simulated field). In practice the procedure works as follows: when you select the candidate stars for PSF estimation, you have to select both the saturated stars you wish to repair and a suitable number of unsaturated sources. The latter will be used to form a preliminary estimate of the PSF, then the core of the selected saturated stars will be repaired and finally a new PSF estimate will be formed, including the unsaturated stars and the halo of the repaired ones (notice that the core of the repaired saturated sources is not used to form the PSF). More details can be found in the online help and in a forthcoming paper (Sect. 7).

5.6 Stars detection

You are now ready to detect the field stars: click on the 'Astrometry and Photometry' button and the *XStarFinder_Run* widget appears. Instead of using the default values and perform two iterations of the analysis procedure at the 3-sigma detection level, we suggest to perform a preliminary run with just one iteration at a higher detection level, say 5-sigma: the reason for this choice will be clear in a while. Just enter 5 in the 'Detection threshold(s)' field and click on the 'Processing' button. After a few minutes, a message gives you the number of detected stars (it should be about 554, i.e. 55% of the total number of stars in this field). Now exit the *XStarFinder_Run* widget and display the synthetic field (with 'Display/Select data/Synthetic field'), given by 554 replicas of the PSF, one for each detected star, superposed on a background estimate. Modify the display options with the task 'Display/Options', setting the same values you have chosen before for the image data (minimum and maximum intensity of 250 and 1e5 respectively, square root stretch and STERN SPECIAL color table). You can compare the image of the stellar field with the synthetic image model created by the program.
Why did we suggest you to perform a rough analysis and find only the stars above the 5-sigma level? The reason is that we wish to improve the PSF estimate before going deeper.

5.7 PSF enhancement and re-start

If you run the 'PSF/Repeat extraction' task, the PSF extraction procedure will be automatically repeated, with the same parameters used before (Sect. 5.5), the only difference being that all the currently known contaminating sources around the PSF stars will be automatically removed. When the computation has terminated, you can display the new PSF estimate with 'Display/Select data/PSF': notice the great improvement. Now you can post-process the support of the retrieved PSF as before (with 'PSF/Post process/Support'), using a circular mask radius of 45 pixels and a threshold for principal component extraction of 0 in the *XImage_Support* widget. We suggest also to smooth a little bit the halo of the PSF, by means of the 'PSF/Post process/Halo smoothing' task, invoking the *XPsf_Smooth* widget. Just hit the 'Processing…' button with the default values of the parameters; the smoothed PSF is automatically displayed at the end of the computation, which might take some minutes. If you are not satisfied with the result, you can modify the parameters and click again on 'Processing…': only the last result will be saved on exit. When you have finished, dismiss *XPsf_Smooth*. Finally normalize the PSF array with 'PSF/Normalize'.

The PSF estimates retrieved in this section and in the last one are shown in Fig. 5. The PSF you would obtain skipping the somewhat tedious step of secondary stars subtraction (Sect. 5.5) would be quite similar to the one in Fig. 5, though with a slightly worse reconstruction of the halo. The reason is quite obvious, since repeating automatically the PSF extraction allows you to subtract the contaminating sources around the PSF stars, but the knowledge of these contaminating sources is somewhat limited by the quality of the first PSF estimate obtained with *XPsf_Extract*.

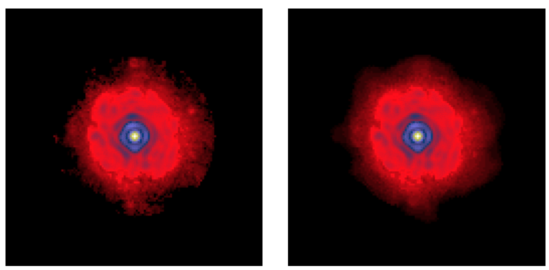

Figure 5. PSF estimates before (left) and after (right) running the 'PSF/Repeat extraction' task.

With this new PSF estimate, run again the stars detection algorithm with 'Astrometry and Photometry', invoking *XStarFinder_Run*. Now you can use a more suitable detection threshold, namely the default value of 3, 3 (indicating two iterations of the analysis procedure at the 3-sigma level). After a few minutes (*how many minutes* depends on your computer) a message informs you that the number of detected stars is around 884, i.e. almost 90% of the total number of sources in this field. It will be difficult to do better (at least with this program!…): you can try with the de-blending strategy, even though you should find just ~1% more stars. Before leaving the *XStarFinder_Run* widget, remember to save the parameters (astrometry and photometry) of the detected stars, even though you can do it also with the 'Image/Save/List of stars' task in the main widget. After closing *XStarFinder_Run*, display again the synthetic field (you need not modify the visualization parameters now) and compare it to the image.

As you may guess, the PSF estimate is extremely important to obtain accurate results, especially at faint magnitudes. Fig. 6 shows an axial plot of the *true* PSF (see also Fig. 1) compared to the retrieved PSF used here. There are two important aspects:

1. The two PSFs are practically coincident until about 7-8 magnitudes below the central peak. Below this level there are some discrepancies which might induce photometric errors on a faint star lying at the edge of the PSF of a 8 magnitudes brighter source. Actually the total photometric range of the stellar population in this field is just 8 magnitudes, so this kind of error should be negligible.
2. Assuming to normalize the two PSF arrays to the same central value, it is clear that the total enclosed light is slightly different. The estimated fluxes are consequently undetermined by an unknown factor, which results in a systematic offset in the magnitudes. In practice the relative photometry between any two sources in the field is preserved, even though this zero-point offset must be taken into account, e.g. when combining magnitudes derived from different exposures of the same field.

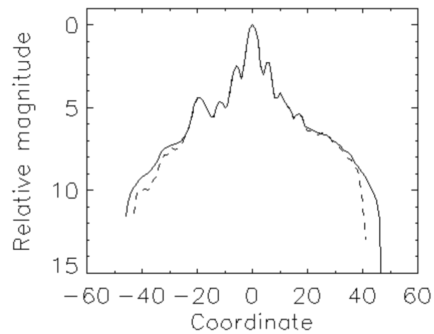

Figure 6. Axial plot of true PSF (continuous line) and estimated PSF (dashed line). The two PSFs are normalized to the same central maximum.

5.8 Lists comparison

In stellar population studies one has to analyze images taken at different wavelengths. The retrieved lists of stars must be compared by matching the reference frames, here assumed to be reciprocally translated and rotated, and then identifying the coincident sources.

In this tutorial you have analyzed a simulated image and you know the answer, so you may compare the retrieved list of stars with the original one (see the file 'stars.txt' included in the released package). Click on the 'Compare Lists' button to invoke the *XCompare_Lists* widget. Load the original list of stars ('stars.txt') and the one you have saved before with 'File/Load/List 1' and 'File/Load/List 2' respectively; then load the reference list with 'File/Load/Reference list' (a good choice is to load the list of PSF stars coordinates you have saved in Sect. 5.5 or the file 'psfstars.txt' included in the released package). The reference list must contain a subset of the objects in list no. 1 and it is used to find a suitable set of points to match the reference frames corresponding to lists no. 1 and 2 respectively. Of course in our case there is neither rotation nor translation, but it is interesting to see how it works. Click on the 'Match coordinates' button to invoke the *XMatch_Coord* widget; specify a guess of the parameters defining the reciprocal translation and rotation of the two frames (in this case use the default values of 0 of course) and click on the 'Match' button. After a while a message appears giving you the refined estimates of the parameters, which in this case are very close to 0. The small discrepancy in the x-coordinate (~0.02 pixels in this case) is due to the fact that the estimated sub-pixel positions of the stars are referred to the absolute centering of the retrieved PSF. After dismissing the *XMatch_Coord* widget and returning back to *XCompare_Lists*, the second list of stars (the original one in this case) has been transformed automatically to map its coordinates onto the reference frame of the first list. Now click on 'Find coincident' to establish coincidences between the stars in the two lists. You are requested to enter a tolerance (in pixels) for matching: a good choice is the PSF FWHM, which in this case is about 3.5 pixels. A dialog message appears with the number of successfully matched stars, which amount in this case to 883, i.e. 1 less than the total number of detected stars. This might suggest that there is one false detection. Before quitting *XCompare_Lists*, remember to save the subsets of common stars out of the two lists, using 'File/Save/List 1' and 'File/Save/List 2'.

Notice that the *XCompare_Lists* task allows you to compare just two lists; in principle you could find out ingenious schemes to compare many lists two at a time. If you are an expert in the data reduction of stellar fields, you surely have much more sophisticated tools for lists comparison than the simple *XCompare_Lists* task: what you have to do perhaps is just to convert the *StarFinder* output to the proper format required by your own programs (keeping in mind that *StarFinder* measures fluxes, not magnitudes).

5.9 Off-line processing

That's all you can do with the widget version of the code. The released package includes some useful routines (see the list in Sect. 3) which may be run from the IDL command line and may help you in analyzing your data. Suppose you wish to plot the photometric errors for the detected stars in the simulated case presented in this tutorial. Assume that in the previous analysis (Sect. 5.8) you have saved the matched lists of 883 objects to the files 'stars_true.txt' and 'stars_found.txt'; the procedure to plot the photometric errors is the following:

```
IDL> stars_true = read_float_data('stars_true.txt', 7)
IDL> stars_found = read_float_data('stars_found.txt', 7)
IDL> mag_true  = 2.5 * alog10(max(stars_true[2,*]) / transpose(stars_true[2,*]))
IDL> mag_found = 2.5 * alog10(max(stars_found[2,*]) / transpose(stars_found[2,*]))
```

```
IDL> plot, mag_true, mag_found - mag_true, PSYM=1, SYMSIZE=0.5, YRANGE=[-1,+1], $
IDL> CHARSIZE=2, XMIN=2, XTITLE='Magnitude', YTITLE='Photometric error'
```

Notice that when you read the data you have to specify the full path of the files 'stars_true.txt' and 'stars_found.txt'. The result is a plot like the one shown in Fig. 7.

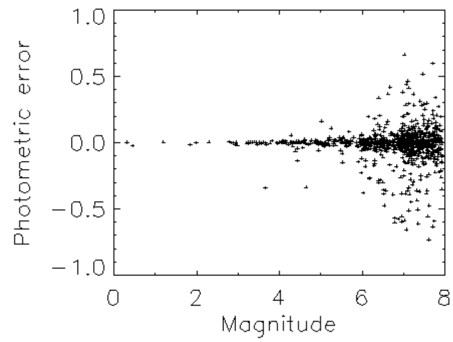

Figure 7. Plot of photometric errors for the detected stars.

Another interesting test is to display the image and mark with different symbols the true stars (1000) and the ones you have detected with the program (884): this allows you to inspect the successfully detected stars, the false detection cases and the lost sources. Assuming you have saved the full list of detected stars in the file 'stars_detected.txt', the procedure is the following:

```
IDL> fits_read, 'synfield.fits', ima
IDL> stars = read_float_data('stars.txt', 7)
IDL> stars_detected = read_float_data('stars_detected.txt', 7)
IDL> window, XSIZE=512, YSIZE=512
IDL> tvscl, congrid(sqrt(ima), 512, 512)
IDL> loadct,15
IDL> crosses, ima, /EX, stars[0,*], stars[1,*], $
IDL> X2=stars_detected[0,*], Y2=stars_detected[1,*]
```

As before, remember to specify the correct path of the data files 'synfield.fits', 'stars.txt' and 'stars_detected.txt'. You should obtain something like Fig. 8, where + symbols indicate the true stars and × symbols indicate the detected objects; the superposition of a + and a × symbol indicates a successful detection; a + alone stands for a lost star and a × alone for a false detection.

6 Future developments

Apart from the improvements related to the widget interface and the addition of auxiliary tools which might be helpful in the analysis of a stellar field, the most important task concerns the development of one or more strategies to handle fields with space-variant PSF. Two possible approaches to the problem, tested on simulated data and described in Ref. 3 (Sect. 7), are

1. removing the anisoplanatic effect from the image, in order to have uniform PSF for the photometric analysis
2. supplying a set of local PSF measurements and performing a kind of space-variant photometry in which the code picks up the appropriate PSF for each star to be analyzed.

A *preliminary* implementation of the second strategy above is already included in the program module for stars detection, called *starfinder* (Sect. 3), though not still accessible from the widget interface. This program module includes another additional feature not available in the widget version of the code, concerning the de-blending of crowded groups of stars. This additional strategy, though very powerful on simulated data, has not been sufficiently tested on real images.

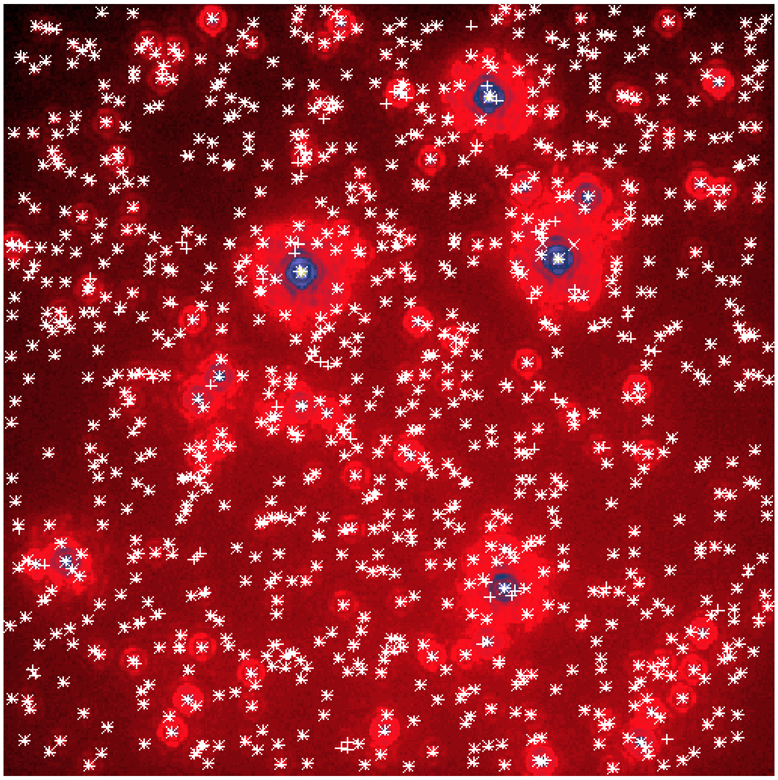

Figure 8. Synthetic stellar field; + symbols indicate the true stars (1000); × symbols indicate the objects detected in the previous run (884).

7 *StarFinder* references

Details on the *StarFinder* code may be found in

1. Diolaiti, E., Bendinelli, O., Bonaccini, D., Parmeggiani, G., Rigaut, F., Proc. of the ESO/OSA Meeting on Astronomy with Adaptive Optics, D. Bonaccini (ed.), ESO Garching (Germany), p. 175 (1999)
2. Diolaiti, E., Bendinelli, O., Bonaccini, D., Close L.M., Currie D.G., Parmeggiani, G., in Proc. of ADASS IX, D. Crabtree, N. Manset and C. Veillet (eds.), ASP Conference Series, San Francisco, in press (2000)
3. Diolaiti E., Bendinelli O., Bonaccini D., Close L.M., Currie D.G., Parmeggiani G., SPIE Proc. 4007, in press (2000)

A further more detailed paper is going to be submitted to A&A Suppl. Ser..

The *StarFinder* code can be used freely. Please just reference *StarFinder* (e.g. with the most recent paper above) in any publication resulting from its use.

Please send requests, suggestions, bug reports to Emiliano Diolaiti (*email*: diolaiti@bo.astro.it).