T-PHOT version 2.0 - Documentation

E. Merlin

May 4, 2016

Contents

1 Introduction

T-PHOT (Merlin et al. 2015; Merlin et al. 2016) is a software designed to perform precision photometry of a low resolution image using the information given by a higher resolution image of the same field. It is the direct heir of TFIT, a software package developed ad STScI, but largely improves on it combining its features with most of the features of CONVPHOT, a similar software independently developed at INAF-OAR, and including many new important features. To learn more about these two codes, please refer to Laidler et al. (2007) and De Santis et al. (2007); the official T-PHOT paper is Merlin et al. 2015A&A, 582A, 15M).

T-PHOT is a robust and easy-to-handle code, with a precise structural architecture (a PYTHON envelop calling `C/C++` "hidden" codes) in which many different routines are included, implementing various numerical/conceptual methods, to be chosen by simple switches in a parameterfile. While a standard, default "best choice" is provided and suggested, the user is thus let free to select her own preferred way of obtaining her catalog.

T-PHOT ensures, as a first advantage, a large saving of computational time with respect both to TFIT and CONVPHOT. Such a result must be credited to three disinct factors: 1) the substitution of many PYTHON scripts with `C` codes, which are *way* faster, 2) a global re-organization of the structure of the code, and 3) the introduction of the FFT method (which was already implemented in TFIT but not in CONVPHOT) in the convolution stage (see below).

Also, T-PHOT is capable of handling large datasets thanks to a clever use of the CPU memory, allowing for the processing of large fields in one single run.

However, T-PHOT also improves on the numerical results with respect to previous codes, thanks to different implementation of many routines and conceptual recipes.

1.1 What's new in this version

This is the version 2.0 of the T-PHOT software package. It features a number of new options with respect to the previous public release 1.5.11. All changes, including minor revisions, bug corrections and chronological developments, are reported in the `log.txt` file. The most important new options are:

- **Background fitting**: two methods (global subtraction of a constant fitted value, and local fit of individual "background templates"), Section 4.3.4
- **Local/individual kernel fitting**: Section 4.7
- **Flux prioring**: Section 4.3.5
- **Statistics on the residuals**: Section 4.4.1

- **RMS threshold for excluding sources from the fit**: see Section 4.3.3
- **Individual kernel registration**: Section 4.5

A technical new option is the *command line input*: it is now possible to enter parameters from command line, in case over-writing the ones specified in the parameter file. Keywords and corresponding values can be entered typing `-<keyword> <value>` after the parameter file specification, e.g.

```
$ tphot param.file -order cutout -hiresfile newimage.fits
```

A revision of the code architecture in the kernel local registration and in the convolution stages has also been performed, to make the workflow simpler and better organized. Now templates are registered on the fly during the second pass convolution stage, and the second pass local kernels are not stored anymore.

2 Instructions for installation and usage

To install and use T-PHOT you need:
- PYTHON (version 2.7.6 or later), and its modules NUMPY, ASTROPY, MATPLOTLIB (on Linux-Ubuntu OS, you can get them using:

```
$ sudo apt-get install python-numpy
```

and similar commands)
- CFITSIO library (download the tarball from website
`http://heasarc.nasa.gov/fitsio/fitsio.html`
and follow instructions in the README file)
- FFTW3 library (download the tarball from website
`http://www.fftw.org/download.html`
and follow instructions in the README file).

Move to your chosen directory

```
$ cd /your/path/
```

move the T-PHOT tarball there and unzip it with

```
$ tar -xvf TPHOTv1p0.tar.gz
```

The `lib` directory contains the PYTHON library and the subdirectory `src`, in which `C` and `C++` source codes are found.

To install T-PHOT,

```
$ cd /your/path/tphot/lib/src
```

and edit the `Macrodefs` file relative to your OS, indicating the paths for the CFITSIO libraries; e.g., use

```
$ locate libcfitsio.so
```

Then, edit the first line in `/your/path/tphot/lib/src/c_src/Makefile` and in `/your/path/tphot/lib/src/fitter_src/Makefile` including the `Macrodefs.*OS*` you are using.

Also, edit your `/.envrc` including `/your/path/tphot/bin` in `$PATH`.

Then

```
$ cd /your/path/tphot/
```

Change permissions if needed (`$ chmod +x install.py clean.py` and

```
$ sudo ./install.py
```

NOTE that sudo-ing is required to install T-PHOT in the /bin directory of the user.

To run T-PHOT make sure the `.envrc` file contains something like:

```
export LD_LIBRARY_PATH=~/cfitsio/lib:$LD_LIBRARY_PATH
export FFTW_PATH=/usr/local/lib
export PYTHONPATH=/usr/local
export PATH=/your/path/tphot/bin:$PATH
```

After having rehashed your shell, `$ cd` to the directory where data files are, copy the parameterfiles you can find in `/your/path/tphot/lib`, edit them as you need, check all the needed input is there (see Sect. 2.2), and type e.g.:

```
$ tphot taskparfile1.param
```

or

```
$ tphot taskparfile1.param > out_pass1.log &
```

or, if you wish to run both passes (see Sect. 3) in sequence,

```
$ tphot taskparfile1.param ; tphot taskparfile2.param
```

(NOTE: the parameterfiles contain an explanatory guide to edit them; it should be read carefully). The parameterfiles come with default "best-choice" options: store cutouts, convolve using FFT, fit with *cells-on-objects* method (see Sect. 4.3.2), etc., but they can be edited as desired.

One can also choose to run the code in *interactive* mode, typing

```
$ tphot -i taskparfile1.param
```

which will start a run with some input to be given during the process.

Finally, from version 2.0 it is possible to add command line options, specifying the value of any parameter and over-writing the one in the parameter file: e.g.,

```
$ tphot taskparfile1.param -order cutout -hiresfile newimage.fits
```

To uninstall,

```
$ ./clean.py
```

2.1 Testing the installation

The tarball includes a directory `test`. After installing T-PHOT, move to the directory and type

```
$ tphot test.param
```

In a few seconds the test should end and the results should be compared with the one stored in the `output_check` subdirectory.

2.2 Input

T-PHOT can use different types of datasets as input priors. Depending on the choice the user makes, the needed input files will be different. NOTE: *the IDs of the objects must always be consistent*, i.e. each object/component/model must have its own unambiguous identification number.

To run T-PHOT you need:

- for the `PRIORS` stage:
  - for *true high-resolution cut-out priors*, a detection, high resolution image (HRI hereafter) in `.fits` format (values in counts/s); a catalog of the sources in the HRI, obtained e.g. using SExtractor or similar codes; the catalog *must* have the following format:

    ```
    # id x_obj y_obj xmin ymin xmax ymax loc_bckg obj_flux
    ```

    where `x_obj` and `y_obj` are the pixel coordinates of the source centroid, `xmin ymin xmax ymax` are the pixel boundaries of the source, `loc_bckg` is a constant local background [1], and `obj_flux` is the total flux of the object (in counts/s) as determined within some aperture; and a segmentation map of the HRI, in `.fits` format, again obtained e.g. using SExtractor or similar codes, having the value of the `id` of each source in the pixels belonging to it, and zero everywhere else;
  - for *theoretical model priors (e.g.* Galfit *models*, the stamps of the models (one per object) and a catalog listing them; if models are multi-components, one separate stamp per component is needed, and one catalog for each component is needed (e.g. one catalog for bulges and one catalog for disks), having the same names plus a "_1", "_2" etc. suffix (this is because the components will be treated as different objects for all purposes); the catalog must have the same format described above, and *in particular it must report in the 9-th (and last) column the magnitudes of the models* (this will be used to sort the sources by flux in the fitting procedure). A stand-alone module, `galfit2tphot.py`, is provided and can be used to convert Galfit

[1] If the HRI is background subtracted, this columns should be zero for all sources. However, T-PHOT includes a switch in the parameterfile to choose whether this values should be subtracted when cutting out stamps, see Sect. 4.1, or ignored.

stamps having arbitrary names and header format, and GALFIT catalog listing `x` and `y` positions in the GALFIT stamps reference frame (which is the standard GALFIT output), into T-PHOT readable format. To do so, run

```
galfit2tphot HRI ZeroPoint modelscat modelsdir outcat outdir
```

(note that if `modelsdir` and `outdir` are the same directory, models will be over-written in the new format; otherwise, they will be copied in the new output directory);

– for *unresolved, point-like priors*, a catalog of positions with format

```
# id x_obj y_obj
```

T-PHOT can process together mixed priors of different types, provided all input is given correctly.

- a low resolution image (LRI), on which the fluxes will be measured, in `.fits` format, with the same orientation of the HRI (i.e., no rotation; values in counts/s). The pixel scale can be equal to, or an integer multiple of, the HRI pixel scale, with pixels boundaries consistently overlapping. The LRI should be background subtracted; however, from version 1.6.1 T-PHOT can subtract a constant background over the whole image, and a local background for each source, during the fitting stage (see Sect. 4.3.4);
- the LRI RMS map, in `.fits` format;
- for *real or model priors*, a convolution kernel K, in the format of a `.fits` image or of a `.txt` file, matching the PSFs of the HRI and the LRI so that $PSF_{LRI} = K * PSF_{HRI}$, where $*$ is the symbol for convolution. The kernel *must* have the HRI pixel scale; for *unresolved priors*, the LRI PSF (if both types of priors are used, both the LRI PSF and the convolution kernel must be given).

2.2.1 Header of files: needed keywords

Before starting a run of T-PHOT, be sure that all the input images have the following keywords in their headers: `CRPIXn, CRVALn, CDn_n, CTYPEn` (n=1,2). If models are used as priors and they are produced externally, they also must have these keywords in the header, plus a `TOTFLUX` keyword, set to 1.0.

3 Basic global structure of the code

The T-PHOT global structure is similar to the TFIT structure, with some semplifications and adjustments. The code goes through "stages", each of which performs a well defined task. The best results are obtaining performing two runs ("pass 1" and "pass 2"), the second one being run using local kernels registered after the X, Y shifts determined in the first pass during the `dance` stage; it is

important to set the keyword `multikernel` to `true` for the second pass (see Sect. 4).

The pipeline followed by the code is specified by the keyword `order` in the parameterfile. The simplest way to run the code is to simply put

```
order standard
```

This is interpreted by T-PHOT as a typical first pass run, i.e. the following pipeline:

```
order priors, convolve, fit, diags, dance
```

Both forms are accepted, as well as other combinations of stages (see their description below), provided they make sense!

A second typical second pass run would be

```
order standard2
```

or, equivalently,

```
order convolve, fit, diags, archive
```

It is also possible to set

```
order FIRstandard
```

and/or

```
order FIRstandard2
```

for standard far-infrared (FIR) processing, i.e. using point-like priors and PSF-shaped templates during the fit. In this case the real pipelines will be

```
order positions, fit, diags, dance
```

and/or

```
order positions, fit, diags, archive
```

respectively. NOTE that if the `FIRstandard` options are used, any input given for high resolution real priors or model priors will be ignored.

The possible stages are the following:

- `priors`: creates/organizes stamps for sources as listed in the input priors catalog(s). T-PHOT automatically re-organizes the pipeline on the basis of the input files listed in the parameter files; if the interactive option is used, it will ask to confirm the re-organized pipeline after proceeding on. Always remember to set as `True` the keyword(s) in the parameter-files describing the type(s) of priors that are going to be used (`usereal`, `usemodels`, `useunresolved`). The real pipeline stages for priors processing are `cutout` and/or `models`, and they can be used in place of the general `priors` keyword. For unresolved priors, the stage `positions` can be explicitely specified and put *after* the keyword `convolve`; however, this stage can be as well managed by the general `priors` keyword. See Sect. 4.1 for more details;

- `convolve`: convolves each high resolution stamp (after culling) with a convolution kernel K to obtain normalized models ("templates") of the sources in low resolution; also, if the pixel scale of the images is different transforms templates accordingly. If the keyword `multikernels` is set to `TRUE`, uses a list of local X, Y shifts to produce locally registered kernels on-the-fly. If the keyword `mkext` is set to `TRUE`, uses external local kernels as provided by the user (see Sect. 4.7). Finally, if an input catalog of unresolved sources is given, merges the catalog of such templates with the one produced in the `position` stage;

- `fit`: performs the fit obtaining the multiplicative factors to match each template flux with the measured one (see Sect. 4.3);

- `diags`: selects the best fits[2] and produces the final output catalog with fluxes and errors, plus some other diagnostics, see Sect. 5. If the keyword `residstats` is set to `TRUE`, computes some statistics on the residual image produced subtracting the fitted models from the original measurement image, see Sect. 4.4.1. If the output catalog has a different number of sources with respect to the input templates catalog, rearranges the output to match it (see Section 5);

- `dance`: obtains shifts in the X, Y directions to individually register kernels in a subsequent convolution stage; it can be skipped if the user is interested in a single pass run;

- `plotdance`: plots a diagnostic image of the shifts obtained in the dance stage; it can be skipped for any other purpose than diagnostics;

- `archive`: archives all results in a subdirectory whose name is based on the LRI and the chosen fittin method.

T-PHOT works in the HRI reference frame for all the `priors` stages, turning to the LRI reference frame during the `convolve` stage. This is the whole scheme:
- priors catalog is produced in HRI reference;
- templates catalog is produced in LRI reference;
- in the fitting routine, sources are read in from the templates catalog, loaded into a grid with coordinates in LRI reference, and the fit is done in LRI reference;
- the final output catalog is LRI in reference.
Also, the header of the LRI image is modified with the addition of the two parameters `Xoff` and `Yoff` giving the relative offset of the LRI w.r.t. the HRI.

[2]Each source is fitted more than once if an arbitrary grid is used, as in the standard TFIT approach.

4 Description of T-PHOT stages

4.1 Priors

T-PHOT creates/organizes stamps for sources listed in the given input priors catalog(s). The stamps are in `.fits` format, each containing a single source. An output catalog is produced, merging all priors input catalogs.

As already said, T-PHOT automatically re-organizes the real pipeline on the basis of the keywords given in the parameter files: set as `True` the keyword(s) in the parameter files describing the type(s) of priors that are going to be used (`usereal`, `usemodels`, `useunresolved`), and set to `False` or comment out the other(s). The real stages are `cutout` (for real priors: creates/saves stamps using the HRI, the segmentation map and the input catalog; pixels outside the segmentation of each source are put to zero in their stamps; if the keyword `savecut` is put to `true`, the stamps will be stored, while this is automatically done if mixed priors are used; otherwise, they will be produced on-the-fly during the subsequent convolution stage; if the relative switch is set to `true` in the parameterfile, the local background as reported in the input catalog is subtracted as a constant from each pixel flux value; finally, the total flux within the cutout area is computed and listed in the output catalog, and the stamp total flux is normalized to one) and `models` (for models: organizes models stamps and catalog).

If a catalog of unresolved point-like sources is given in input, the stage `positions` is also performed. In this stage, templates for unresolved sources are produced shifting the LRI PSF to the exact position given in the catalog, including intra-pixel interpolation.

The general order of the mixed priors processing is

```
cutout - models - convolve - positions
```

where the `convolve` stage is described below. At the end of each stage, the processed catalog is merged with the one outputted by the previous stages, if necessary.

If the keyword `culling` is set to `true`, T-PHOT will check which sources in the HRI catalog(s) do not belong to the LRI frame and can therefore be skipped, *before* producing the stamps; if it is set to `false`, this stage will be performed after producing all stamps, before the convolution process (see Sect. 4.2.2)

4.2 Convolve

In this stage, high resolution cutouts (undependently of their origin) are convolved to low resolution via convolution with a kernel K. This can be either a global kernel, input via the keyword `kernelfile`, or a local/individual kernel if the keyword `mkext` is `TRUE` (see Sect. 4.7), or a locally registered kernel obtained on-the-fly from one of the previous two if a file listing X, Y shifts has been produced in a previous `dance` pass and the keyword `multikernels` is set to `TRUE` (see Sect. 4.2.1).

The convolution can be performed via straight pixel-per-pixel processing, or using a much faster FFT convolution, directly calling the `C` library `FFTW3`; the choice is specified via the relative keyword in the parameterfile. Each high resolution stamp is first padded adding half the size of the kernel to each side, and then bringing the dimensions up to the nearest power-of-two integer, which speeds up `FFTW3` computations. Finally, the resulting convolved template is cutted to its right dimensions after the convolution. *Note* that, as declared in its scientific distribution paper, "`FFTW` is typically faster than all other publicly available DFT software, including the well-known `FFTPACK` and the code from Numerical Recipes". See `www.fftw.org/benchfft/`.

After convolution, templates are transformed to the LRI pixel scale if the latter is different from the HRI one (as specified by the `relscale` keyword in the parameterfile). Templates are subsequently cut where necessary, so that they are all within the limits of the LRI; full templates can be stored anyway if desired. Also, they are normalized to total flux = 1.

As anticipated, real prior cutouts from the HRI can be obtained on-the-fly during the `convolve` routine, processing sources from scratch to the final template (i.e.: read the catalog; check if it will be processed; if so, DO NOT print out cutout but simply produce an array and convolve it). To do so, put the keyword `savecut` to `false` in the parameterfile, and explicitely exclude the stage `cutout` from the keyword `order` in the parameter file (i.e., do NOT put `order standard` but instead `order convolve, fit, diags, dance` etc.). In this way, one can choose whether to obtain the templates for a large HRI and only use a subset of them when processing a smaller LRI, or, on the other hand, to skip the cutout production stage if HRI and LRI are only used together and of similar dimensions. The computational time is nearly identical.

However, if mixed priors are used, cutouts for real priors will be automatically saved and stored, to avoid possible confusion.

4.2.1 Second pass with locally registered kernels

During the `dance` stage (Sect. 4.5), a list of X, Y shifts is produced (in HRI pixel scale). If the keyword `multikernels` is set to `TRUE` in the parameter file, T-PHOT reads the file produced during this stage, which lists the shifts for each source (the file is named after the `ddiagfile` keyword in the parameter file). During the convolution process, it creates a locally registered kernel on-the-fly, and uses this shifted kernel to produce a registered template.

NOTE: for unresolved priors, the shifts are re-cast in LRI pixel scale during the `positions` stange in pass 2.

NOTE: kernels are registered individually for each source since v2.0. Previously they were registered on "regions" of predetermined sizes, which could lead to large inaccuracies. The keyword `dzonesize` in the parameter file now gives the minimum size of the cells on which the cross-correlation between the collage model image and the LRI is made: if the template of a source is smaller than it the box is enlarged to that value. See Sect. 4.5.

4.2.2 Culling

In T-PHOT sources which are present in the input HRI catalog but are largely outside the LRI frame can be automatically excluded *before* the `priors` stage (to avoid producing stamps for the whole HRI when one is interested only on the sources subset matching the LRI field), or right before the convolution stage, to produce the stamps for a whole HRI for subsequent re-use but only convolve the relevant ones. In this case, the prediction of the sources to be culled out is performed *before* the convolution, since the dimensions of the kernel and therefore of the resulting templatescan be computed in advance.

4.3 Fit

In T-PHOT, a `C++` code is still used, largely based on the original TFIT fitting code, but including many new features, which are described below.[3]

4.3.1 The basic method

The search for the LRI fluxes of the objects detected in the HRI is performed creating a linear system

$$I = F_1 P_1 + ... + F_N P_N \tag{1}$$

where I contains the pixel values of the fluxes in the LRI, P_i are the normalized fluxes of the templates for the N objects in the (region of the) LRI being fitted, and F_i are the multiplicative scaling factors for each object. In physical terms, F_i represent the flux of each object in the LRI (that is, it is the unknown to be determined).

Once the normalized templates for each object in the (region of the) LRI have been generated during the `convolve` stage, the best fit to their fluxes can therefore be derived by minimizing a χ^2 statistic,

$$\chi^2 = \left[\frac{\sum_{m,n} I(m,n) - M(m,n)}{\sigma(m,n)} \right]^2 \tag{2}$$

where m and n are the pixel indexes,

$$M(m,n) = \sum_i^N F_i(m,n) P_i(m,n) \tag{3}$$

and σ is the RMS value in the pixel.

[3]The routine is parallelized; in the parameter file, the desired number of CPUs to be used can be specified with the keyword `nproc` (default=1); `nproc` should be an even number, however T-PHOT will adjust it to its optimal value if it is odd or it is higher than the disposable CPUs on the working machine. Then, a pool of workers is called by the PYTHON module `multiprocessing`; each CPU reads the whole set of data but then only works on a region of the image to be fitted and produces a catalog. At the end of the fitting procedure, catalogs are joint in a single one.

The output quantities are the best-fit solutions of the minimization procedure, i.e. the F_i parameters and their relative errors. They can be obtained resolving the linear system

$$\frac{\partial \chi^2}{\partial F_i} = 0 \tag{4}$$

for $i = 0, 1, ..., N$.

In practice, the linear system can be rearranged into a matrix equation,

$$AF = B \tag{5}$$

where the matrix A contains the coefficients $P_i P_j/\sigma^2$, F is a vector containing the fluxes to be determined, and B is a vector given by $I_i P_i/\sigma^2$ terms. The matrix equation is solved via one out of three possible methods (see Sect. 4.3.2).

4.3.2 Fitting options

- Three different methods for solving the linear systems are now implemented and they can be selected with a simple switch in the parameterfile: namely, the LU method (used by default in TFIT); the Cholesky method; and the Iterative Biconjugate Gradient method (used by default in CONVPHOT).

- A threshold can be imposed so that only pixels with a flux higher than it will be used in the fitting procedure; again, the value of the threshold must be defined in the parameterfile.

- Sources which end up with a large, unphysical negative flux ($f_{meas} < 3\sigma$) can be excluded from the fit, and in this case a new fitting procedure will be performed; to do so, switch the `clip` keyword in the parameterfile to `TRUE`.

- Along with the standard TFIT method (in which the LRI is subdivided into an arbitrary grid of cells, and the fit is performed - that is, a linear system is solved - independently in each cell; in the end, the best fit for each source is chosen as the one obtained in the cell in which the object is closer to the center) and the standard CONVPHOT method (in which the fit is performed on the whole LRI as a whole, in a single step, requiring large amounts of computational time and memory if the image is relatively large), a new *cells-on-objects* fitting method can be chosen. If this method is chosen, the sources are first ordered by decreasing HRI flux; then, a loop is performed on them, and a fitting cell is built around each one, initially having the dimensions of the object template; the cell is subsequently enlarged, and objects overlapping the central source or any other object already included in the cell are appended to the cell list, unless they are way fainter than the central source, *or* they only overlap for a tiny fraction of their area. The fitting procedure is then performed on the cell; at the end, only the central object is considered, its measured flux is stored, and *the object is subtracted*

from the LRI. In this way, there is no possibility that a bright object may contaminate the fit of any other source in the image. Large computational time saving is possible, depending on the level of blending/confusion in the LRI: if the latter is very high, most sources will be overlapping, and the cells will be very large, ultimately resulting in repeating many times the fit on regions with dimensions comparable to the whole image. On the other hand, if the confusion is not dramatic, a saving in computational time up to two orders of magnitude can be achieved. NOTE that using the arbitrary cells method can introduce potentially large errors to the flux estimates, due to wrong assignations of peripheric flux from sources located outside a given cell to sources within the cell. The single fit on the whole image should always be the option of choice, as it provides the most accurate estimate. However, this may not always be possible, depending on the size of the problem. In such cases, the *cells-on-objects* methods can provide an acceptable alternative, provided the blending in the LRI is not too dramatic.

4.3.3 Exclusion of high-RMS sources

To exclude from the fit sources belonging to regions with exceedingly large RMS values (e.g. flawed regions, or artificially enlarged borders) it is possible to include a keyword `rmscheck` in the parameter file; if the value is set equal to some $c_{RMS} > 0$, a check is performed on the RMS map and sources having their central pixels with a value higher than c_{RMS} are automatically excluded fro mthe list of the sources to be fitted. These sources will be re-included in an extended version of the final output catalog (see Section 5).

4.3.4 Background subtraction

From version 1.6.1, T-PHOT can subtract a constant background from the whole image, and/or a local background for each source, during the fitting stage. To subtract the constant global background, set the keyword `fitbackground` to `true` in the paramfile.

IMPORTANT NOTES:

- It is strongly recommended to use this option only with the whole image fitting method. If a cell method is used for the fit, the local background will be computed for each cell and this might lead to patchy solutions.
- The value of the background will only be outputted in the log file of the fitting routine, while the model and residual images will *not* be background subtracted; to visualize the results, the user must subtract the fitted value from the residual image. On the other hand, the fitted fluxes in the output catalog obviously take into account the background.

To fit a local background for each source, T-PHOT will build an additional template that will be fit simultaneously with the real object template. Set the

keyword `fit_loc_bkgd` in the paramfile to an integer giving the offset to be assigned to the IDs of the fake bakground templates (it should be larger than the maximum ID of real objects). IMPORTANT NOTE: The fitted value for these background templates may scatter significantly from a reliable average value. Also, including these templates will change the covariance matrix and hence affect the error budget of the measured fluxes. To cope with these issues, it is strongly suggested to build a model image including only the fitted background templates (see Sect. 4.4), subtract it from the real image (perhaps after smoothing), and repeat the fit in standard mode on this final background subtracted image.

4.3.5 Flux prioring

From version 1.6.4, it is possible to perform the fit with an option to constrain fluxes within a chosen interval around a fixed value.

To do so, the matrix $AF = B$ is modified as follows:

- when $i = j$, the element a_{ij} now changes to $a_{ij} = a_{ij} + 1/\sigma_i^2$, where f_i is the estimated flux for source i that you has to be used as prior for that source, and σ_i is its associated uncertainty;
- $B_i = B_i + f_i/\sigma_i^2$.

To enable this *flux prioring* option, a file must be prepared with the following format:

```
# id flag fluxprior fluxerror
```

in which `flag` is equal to 1 if the object is to priored and to 0 if it has to be fitted freely, `fluxprior` is the value f_i around which it has to be fitted, within an error `fluxerror` σ_i. Then, add the name of the file to the parameter file, with keyword `fluxpriorscat`.

4.3.6 Errors estimate

In this same stage, covariance output is computed. Errors to each source are assigned as the square root ot the diagonal element of the covariance matrix relative to that source. Errors are listed in the output catalog; the whole covariance matrixes are listed in a separate output text file, and are also graphically reproduced in `.fits` files in a subdirectory.

NOTE that using any cell method for the fitting will affect the uncertainty budget (the nominal errors outputted along the best fit fluxes), since a different matrix is resolved for each cell and the errors are computed via covariance matrix.

NOTE also that this error budget is a *statistical* uncertainty, and is not related to any possible *systematical* error, which can instead be tracked via flagging of potentially problematic sources (see Sect. 4.4.2).

4.4 Diags

This is a module used to extract the best solutions from the different ones found within different cells, assemble the final output catalogs, and produce the diagnostic images (collage and residual). NOTE: it is possible to input a file containing a list of IDs from the HRI catalog to be *excluded* from the collage and residual images (keyword `exclfile`). This feature can be useful to isolate objects removing neighbors (to do so, include in the list the ID of the object to be kept and see the residual image), or to remove bright foreground sources leaving background objects. The output catalog contains the following data listed for each detected source (as reported in the catalog file header): `id`, `x` and `y` positions, id of the cell in which the best fit has been obtained (only relevant for arbitrary grid fitting method), x and y positions of the object in the cell and distance from the center (always equal to 0 if the *cells-on-objects* method is adopted), fitted flux and relative error (these are the most important output quatities), flux of the object as determined by SExtractor (i.e., the same given in the input catalog), flux of the object as determined in the `cutout` stage (it should be equal to the previous one if the segmentation was not dilated), *flag* indicating a possible bad source (see below), number of fits for this source (only relevant for arbitrary grid fitting method, 1 in all other cases), maximum flag (old feature). The last two columns give ID of the object having the largest covariance with the present source, and the *covariance index*, i.e. the ratio of such maximum covariance and the variance of the object itself; this number can be considered an indicator of the reliability of the fit, since large covariances often indicate a possible systematic offset in the measured flux of the covarying objects (see Merlin et al. 2014).

4.4.1 Statistics on the residual image

From version 1.6.1, it is possible to obtain a set of diagnostic statistics computed on the residual image. To do so, set to `TRUE` the keyword residstats. A file `residual_stats.cat` will be produced, listing mean, median, RMS and kurtosis computed on the residual image at each source's positions, on all the pixels on which the corresponding template model is extended. Also, the same values computed only on an inner and outer regions (the limit of such regions is defined where the flux of the template is half the value of the peak) will be output.

NOTE that the module `tphot_resid_stats.py` can also be used as a stand-alone script, with command line

```
python tphot_resid_stats.py fluxcat templdir templcat residimg
```

where `fluxcat` is the catalog with the fitted fluxes output by T-PHOT, `templdir` and `templcat` are the folder and the catalog of the convolved templates, and `residimg` is the residual image.

4.4.2 Flags

There can be different possible causes for systematic offsets of the measured flux with respect to the true flux of a source. Some of these causes can be easily identificated and a flag can be given to the interested sources. T-PHOT assigns the following flags:

- +1 if the prior has saturated or negative flux
- +2 if the prior is blended with another one (the check is performed on the segmentation map)
- +4 if the source is at the border of the image.

4.5 Dance

In this stage, shifts in HRI reference pixels, along `X` and `Y` directions, are computed for each source, for subsequent use in a second, multikernel pass. NOTE that in previous versions the shifts were computed on a predefined grid of boxes, potentially leading to inaccuracies. There are three conceptual stages:

- x, y shift is computed for each object, cross-correlating the collage image and the LRI on the its template area[4]; if the size of the template is smaller than `dzonesize`, it is enalrged;
- to avoid spurious shifts due to local noise, the final values are smoothed over the closest neighbors. The keyword `nneighinterp` in the parameter file defines the number of neighbors over which the smoothing is performed; if it is set to 0, T-PHOT will include all and only the neighbors within a fixed area defined by the radius $3 \times \sqrt{A/N_{sources}}$, where A is the area of the image. If the keyword is set to a negative value, no smoothing will be performed. A reasonable default value can be 50 <`nneighinterp`< 100.
- finally, a catalog listing the shifts for each source are produced.

4.6 Second pass multikernel convolution

Having obtained the local shifts to register the sources, they can be used for a second pass, to obtain more astrometrically precise results. In this pass, the keyword `multikernel` in the parameterfile must be switched to `true`. T-PHOT builds on-the-fly local kernels to produce the new templates that will be used for the pass2 new fit.

If unresolved priors are used, the list of shifts generated in the `dance` stage will be used by the `positions` routine during the second pass to produce correctly shifted PSFs and generate new templates.

[4] FFT and direct cross-correlations are implemented, the latter being the preferred default choice giving mores precise results at the expense of a slightly slower computation.

4.7 Input external list of local kernels

From version 1.6.3, it is possible to input an *external* list of local or individual kernels, to be associated arbitrarily to the HRI cutouts, e.g. for varying PSFs in different regions of the LRI. To do so, just `cd` to the working directory and create a subdirectory named `KERNELS`, containing all the kernels in `.fits` format; *IMPORTANT: the kernels must be named* `id_KERNEL.fits`*, where* `id` *is the ID of the source they must be associated with*: e.g., the kernel that must be used to convolve the object with ID=1 must be named `1_KERNEL.fits`, and so on. This way, T-PHOT will associate the correct kernel to each source. Then, be sure that the keyword the keyword `mkext` exists and is set to `true`, *in the paramfile of both pass1 and pass2*. In pass2, the keyword `multikernels` can be switched to `TRUE` if the user wants to locally registrate the kernels using the shifts computed during the `dance` stage of pass1.

4.8 Archive

At the end of the second pass, if the stage `archive` is included at the end of the process, all results (catalogs, diagnostic images, log files etc.) will be stored in a new subdirectory, whose name is based on the LRI and the chosen fittin method.

5 Output

T-PHOT output files are as nearly identical to the TFIT ones. Namely, they are:

- the "best" catalog described above (named after the `tphotcat` keyword in the parameterfile), reporting source IDs, positions, measured fluxes (counts/s) and errors, plus other useful diagnostic quantities. NOTE: from version 1.6.6, if the final catalog has a different number of sources with respect the templates catalog (e.g. because of culling, or because of exclusion from the bit for having too high RMS), a new catalog with a `_COMPLETE` suffix will be created, in which the missing sources will be included with fitted flux equal to $-0.990000E+02$ and error $0.990000E+10$.

- two catalogs reporting statistics for the fitting cells and the covariance matrixes (named after the keywords `tphotcell` and `tphotcovar`;

- a model `.fits` image, obtained as a collage of the templates, named after the keyword `modelfile`;

- a diagnostic residual `.fits` image, obtained subtracting the collage image from the LRI;

- a subdirectory containing all the templates;

- a subdirectory containing the covariance matrices in graphic (`.fits`) format;
- a few files relative to the shifts of the kernel for the secon pass and a subdirectory containing the shifted kernels.

6 Single source processing

The T-PHOT `C` codes can be launched directly, to study one single source (or some of them of course).

These are the command lines:

- `cutout`:

  ```
  $ tphot_cutout HRI SegImage Cat OutDir OutCat SubBckg
  ```

  where: `Cat` is a `.txt` file in SEXTRACTOR shape, i.e.: `id x y xmin ymin xmax ymax bckgd flux`, with the values copied from the real SEXTRACTOR catalog for the desired source(s) (*important*: remember to add a blank line at the end!); `OutDir` is the output directory, create it if does not exist or just give `./` as input, and `SubBckg` is 1 if the value given in the 8th column of the input catalog must be subtracted.

- `convolution`:

  ```
  $ tphot_convolve InDir InCat Kern OutDir OutCat
       Xoff Yoff LRI 0 None 1 0 HRI SegImage PixRatio
       SubBckg 0 None 0 Normalize
  ```

  where: `InDir` is the directory in which the cutout thumbnail to process is found; if there is no cutout the convolution will be done using the `HRI` data, but `InDir` *must* exist (even if void), so just create it or print `./` as input; InCat is the same as in the cutout case; `Kern` is the kernel file, in `.fits` format if the last digit (before the `HRI` argument) is "0", or in `.txt` format if it is "1"; `OutDir` is the output directory, create it if does not exist or just give `./` as input; `Normalize` is equale to "1" to normalize the convolved template to unitary flux; `Xoff` and `Yoff` are the offset in X and Y directions of the LRI w.r.t. the HRI, to find them if you don't know them:

  ```
  $ python /your/path/tphot/lib/get_offset.py LRI HRI
  ```

  while `SubBckg` (1 = True) is used in case the cutouts must be created on the fly.

7 Validation and computational times

Please refer to Merlin et al. (2015).
Please also consider not to harm animals for your pleasure. :)