A Brief Guide to Data Reduction

Luca Tortorelli

October 2018, v1.0

This short manual is intended to give a brief overlook on how to go from the raw images to fully astrometrically and photometrically calibrated ones.

1 Software requirements

The data reduction is performed mostly in Python, but additional softwares are required such as ds9, Source Extractor, SCAMP, SWARP and/or DeepSkyStacker.

1.1 Python Installations

Python comes pre-installed on Linux and MacOS, while on Windows it is not prepackaged.

Windows users can download and install Python from its website https://www.python.org/downloads/windows/. My suggestion is either to install Python 2 with the latest 2.7 version or to install Python 3 with one of the 3.6.* versions. You can find an example of a step by step guide at https://www.howtogeek.com/197947/how-to-install-python-on-windows/.

The next step is to setup a Python environment. There are two ways of doing that: creating a virtual environment or installing Anaconda. The latter is easier to setup and install, but, especially for Linux and MacOS users, I strongly suggest the use of virtual environments.

1.1.1 Virtual environment

For **MacOS** users:

- Install homebrew as described at https://brew.sh/ (when you run the shown comand press RETURN to answer the first question).
- Run 'brew install python2' and 'brew install python3'.
- Execute

  ```
  $ which python2
  ```

  and

  ```
  $ which python3
  ```

  to check if they are installed into the folder '/usr/local/bin'.
- Run

```
$ pip2 install -U pip setuptools
```

- Run

  ```
  $ pip2 install virtualenv
  ```

- Alter your '~/.profile' so that the locale and language is set properly:

  ```
  $ cat .profile
  $ export LC_ALL=en_US.UTF-8
  $ export LANG=en_US.UTF-8'
  ```

- If you changed your '~/.profile' file reload the changes by either opening a new terminal or by executing:

  ```
  $ source .profile
  ```

For **Linux** users:

- Check if you can submit

  ```
  $ python2
  ```

  and

  ```
  $ python3
  ```

  from the command line. If not, use your linux distributions package manager to install python2.7 and a 3.6* version of python3. On Ubuntu this is

  ```
  $ sudo apt install python2
  $ sudo apt install python3
  ```

- Next check if you can run

  ```
  $ pip2 --help
  $ pip3 --help
  ```

  Again use the package manager to install the packages, e.g. on Ubuntu use

  ```
  $ sudo apt install python-pip
  $ sudo apt install python3-pip
  ```

- Next install virtualenv globally for Python 2:

  ```
  $ sudo pip2 install virtualenv
  ```

- Alter your '~/.bashrc' so that the locale and language are set properly:

  ```
  $ cat .bashrc
  $ export LC_ALL=en_US.UTF-8
  $ export LANG=en_US.UTF-8'.
  ```

- If you changed your '~/.bashrc' file reload the changes by either opening a new terminal or by executing: 'source ~/.bashrc'.

For both MacOS and Linux users, the actual activation of the virtual environment is the following:

- Python 2: run

  ```
  $ cd <folder_name>
  $ virtualenv venv
  ```

  By typing 'ls venv' the folders 'bin','include' and 'lib' should be listed.

- Python 3: run

  ```
  $ cd <folder_name
  $ python3 –m venv venv
  ```

To use the local interpreter and packages installed into this virtual environment, you have to activate it first:

```
$ cd <folder_name
$ source venv/bin/activate
```

By typing 'which python', the path to the bin folder into the virtual environment should be displayed, e.g., 'venv/bin/python'. This has to be done again every time you open a new terminal window or tab.

Every time you have to install a new python package, you simply type

```
$ pip install <package_name>
```

For **Windows** users: if you install Python2 >2.7.9 or Python 3 >3.4 from https://www.python.org/downloads/windows/, then 'pip' should be already installed. To check for that, type

```
$ pip —help
```

in your Command Prompt, if 'pip command not found' is displayed, then download 'pip' from http://pip.readthedocs.io/en/stable/installing/#do-i-need-to-install-pip and save it to your Desktop. Then in your Command Prompt, navigate to Desktop with

```
$ cd Desktop
```

and execute get-pip.py with

```
$ python get-pip.py
```

The actual installation of the virtual environment is the following:

- In your Command Prompt navigate to the folder where you want to install the virtual environment and then type

  ```
  $ virtualenv env
  ```

- On Windows, virtualenv creates a batchfile with the extension '.bat', so to activate the virtualenv you have to run the activate script in the Script folder, Example:

  ```
  $ C:\Users\'Username'\venv\Scripts\activate.bat
  ```

Every time you have to install a new python package, you simply type

```
$ pip install <package_name>
```

1.1.2 Anaconda

For all users, Anaconda can be download from https://www.anaconda.com/download/. Download and install either the 2.7 version or the 3.6* version. The installation procedure is very detailed, so simply follow the instructions on http://docs.anaconda.com/anaconda/install/. On the same website you'll find also guides on how to activate the environment and install new packages.

1.1.3 List of Python Packages

The following is the list of Python packages you need for the data reduction:

- numpy
- Astropy
- photutils
- astroscrappy
- astroalign
- pyfits (needed for Cosmics.py)
- Jupyter notebook (optional)

1.2 DS9

DS9 displays FITS images and it can be download and installed for all the platforms from http://ds9.si.edu/site/Download.html.

1.3 Source Extractor

MacOS users can easily install Source Extractor with 'brew' or 'Macports'. However remember that you must have your virtual environment deactivated when you use 'brew' or 'Macports'. In your terminal, run

```
$ brew install homebrew/science/sextractor
```

or

```
$ brew install brewsci/science/sextractor
```

since sometimes packages change directory. If you want to install it with Macports, then first you have to install it according to your operating system https://www.macports.org/install.php. Then simply type in your terminal:

```
$ sudo port install sextractor
```

For **Linux** users, the simplest way to have SExtractor up and running is to install the standard binary package the comes with your Linux distribution. Run, e.g.,

```
$ apt-get sextractor
```

(on Debian) or

```
$ dnf sextractor
```

(on Fedora) as root and SExtractor, as well as all its dependencies, will automatically be installed. Alternatively, you can follow the instructions at https://sextractor.readthedocs.io/en/latest/Installing.html.

Source Extractor can't be installed on **Windows**. In Windows 10, however, you can activate the Unix terminal. Maybe there it works, but I have never tested it.

The instructions on how to use it are at https://sextractor.readthedocs.io/en/latest/Using.html.

1.4 SCAMP

SCAMP computes the astrometric solution of astronomical images. Since I provide also a website where to do it, it is not stricly necessary that you install it.

MacOS users can easily install SCAMP with 'Macports'. Simply type

```
$ sudo port install scamp
```

in your terminal window.

Linux users have to follow the instructions on the website https://scamp.readthedocs.io/en/latest/Installing.html.

SCAMP can't be installed on **Windows**. In Windows 10, however, you can activate the Unix terminal. Maybe there it works, but I have never tested it.

1.5 SWARP

MacOS users can easily install SWARP with 'Macports'. Simply type

```
$ sudo port install swarp
```

in your terminal window.

Linux users have to follow the instructions on the website https://www.astromatic.net/software/swarp.

SWARP can't be installed on **Windows**. In Windows 10, however, you can activate the Unix terminal. Maybe there it works, but I have never tested it.

1.6 DeepSkyStacker

DeepSkyStacker performs the same steps as SWARP, but with a graphical interface. It works only on **Windows** as far as I'm aware of and it can be downloaded at http://deepskystacker.free.fr/english/index.html. The website containes several tutorials on how to use it.

2 Data Reduction Steps

In this section we go through the different data reduction steps, with some example code on how to perform the different tasks. The code can be run either in the terminal or inside a Jupyter notebook. Some glossary: 'raw science frame' refers to a single exposure image of the target galaxy or star without any post-processing applied to it, i.e. as it is taken at the telescope; 'dark for flat'

refers to a dark image taken with the same exposure time as the single flat field image, while 'dark for science' refers to a dark image taken with the same exposure time as the 'raw science frame'.

2.1 Dark Subtraction and Flat Fielding Correction

The dark subtraction and the flat fielding correction are applied to the 'raw science frame' according to the following:

$$\text{science frame} = \frac{\text{raw science frame} - \text{master dark}}{\text{master flat}/\langle\text{master flat}\rangle} \tag{1}$$

where 'science frame' refers to the 'raw science frame' after dark subtraction and flat fielding correction, 'master dark' refers to the median image of all the 'dark for science' images, 'master flat' refers to the median image of all the flat images after 'dark for flats' subtraction and $\langle$master flat$\rangle$ is the mean value of the 'master flat' image.

Images can be read with the astropy package. In the terminal, type

```
$ from astropy.io import fits
$ data = fits.getdata('path_to_image', ext=0)
$ head = fits.getheader('path_to_image', ext=0)
```

to get the electronic counts in each pixel and the metadata (header file) of an image. The 'data' variable stores the image in the form of a 2D array (i.e. a matrix), therefore operations such as sum, difference, ratio etc. can be done simply as

```
$ diff = data1 - data2
```

In order to create a median image, you can for example store each 'data' variable for each image as element of a list or create a 3D array having size as the number of images times the number of pixels for each side of the image. Then you simply run

```
$ import numpy as np
$ median_image = np.median(image_list, axis=0)
```

The variable 'median image' will contain the median of the electronic counts of the different images pixels. To create an image that can be displayed also in DS9, run

```
$ fits.writeto('out_image_path', median_image, head, overwrite=True)
```

Once you create the different median images, you can perform the dark subtraction and the flat fielding corrections as in 1.

2.2 Cosmic Ray Rejection

The cosmic ray rejection can be performed with Astroscrappy[1] or cosmics.py.

The syntax for Astroscrappy is the following:

```
$ from astroscrappy import detect_cosmics
$ from astropy.io import fits
$ data = fits.getdata('path_to_image', ext=0)
$ head = fits.getheader('path_to_image', ext=0)
```

[1] https://github.com/astropy/astroscrappy

```
$ mask, _clean = detect_cosmics(data, inmask=None, sigclip=4.0,
                                sigfrac=0.3, objlim=5.0, gain=1.15,
                                readnoise=6.5, satlevel=65536,
                                pssl=0.0, niter=4, sepmed=True,
                                cleantype='meanmask', fsmode='median',
                                psfmodel='gauss', psffwhm=2.5,
                                psfsize=7,
                                psfk=None, psfbeta=4.765, verbose=False)
$ fits.writeto('path_to_image_cr.fits', _clean, head, overwrite=True)
```

The values as just for reference. Read the documentation and figure out by yourself the correct ones. Gain can be read from the header of each image, while saturation and readout noise can be found in the CCD camera manual.

Cosmics.py can be downloaded from https://obswww.unige.ch/~tewes/cosmics_dot_py/. Put the file in your working folder. The syntax is the following:

```
$ import cosmics
$ data = fits.getdata('path_to_image', ext=0)
$ head = fits.getheader('path_to_image', ext=0)
$ c = cosmics.cosmicsimage(gal_raw, gain=1.15, readnoise=0, sigclip = 5.0,
                           sigfrac = 0.3, objlim = 5.0, satlevel=65535)
$ c.run(maxiter = 4)
$ cosmics.tofits('path_to_image_clean.fits', c.cleanarray, head)
```

The values as just for reference. Read the documentation and figure out by yourself the correct ones. Gain can be read from the header of each image, while saturation and readout noise can be found in the CCD camera manual.

2.3 Background Subtraction

The background subtraction can be performed using 'photutils'. First you estimate the background following https://photutils.readthedocs.io/en/stable/background.html for every 'science image'. Then you subtract the background image from each science image. Have a look at the image after background subtraction, the process is not trivial and you have to play with parameters in order to obtain a good result.

An example code for a 1D background subtraction is the following:

```
$ data = fits.getdata('path_to_image', ext=0)
$ head = fits.getheader('path_to_image', ext=0)
$ mask = make_source_mask(data, snr=2, npixels=5, dilate_size=11)
$ mean, median, std = sigma_clipped_stats(data, sigma=4.0, mask=mask)
$ backsub_data = data - mean
$ fits.writeto('path_to_image_backsub.fits', backsub_data, head,
                overwrite=True)
```

2.4 Astrometric Calibration

The astrometric calibration is performed through the website astrometry.net. You need to upload either a single image or a tarball (.tar, .gz), containing fits images to http://nova.

astrometry.net/upload. It will recognize stars in the image and it will give you back the same input images, but with the astrometric solution included.

The same steps can be performed using SCAMP. See https://scamp.readthedocs.io/en/latest/Using.html for a tutorial on how to use it. SCAMP needs a list of catalogues and a configuration file in order to work. First, you need to run Source Extractor on each image. Look for the 'default.param' file and uncomment the lines containing the following parameters: 'XWIN_IMAGE', 'YWIN_IMAGE', 'ERRAWIN_IMAGE', 'ERRBWIN_IMAGE', 'ERRTHETAWIN_IMAGE', 'FLUX_AUTO', 'FLUXERR_AUTO', 'FLAGS', 'FLAGS_WEIGHT', 'IMAFLAGS_ISO', 'FLUX_RADIUS', 'ELONGATION'. The, look for the 'defaul.sex' file, open it and change 'CATALOG_TYPE' in 'FITS_LDAC' and 'PARAMETERS_NAME' in the path to 'default.param'. The rest of the parameters is up to you to discover what they do and which value to choose. Now you can run Source Extractor:

```
$ sex image_path -c default.sex -CATALOG_NAME 'path_to_catalogue_name'
```

These catalogues that you have created for each image will be read by SCAMP. First, create an ASCII file where each line contains the path to the Source Extractor output catalogues, preceded by . Then look for the configuration file (usually it ends with '*.conf') and check that the 'SOLVE_ASTROM' parameters is set to 'Y'. Check the manual for the other parameters meaning and values. Then run SCAMP with

```
$ scamp catalogue_list -c scamp.conf
```

Alternatively, you can use a Python package called 'astroalign' (https://github.com/toros-astro/astroalign). You can install it with

```
$ pip install astroalign
```

This package finds similar 3 point asterisms in an input and reference image and computes the affine transformation between them. It may not work on images of extended objects with few point-like sources or in very crowded fields. However, also in these cases, one of the three methods will do the job. An example code is

```
$ from astropy.io import fits
$ import astroalign
$ data = fits.getdata('path_to_image', ext=0)
$ reference_img = fits.getdata('path_to_ref_image', ext=0)
$ reference_head = fits.getheader('path_to_ref_image', ext=0)
$ aligned_image = astroalign.register(data, reference_img)
$ fits.writeto('path_to_aligned_image.fits', aligned_image, reference_head,
                overwrite=True)
```

2.5 Stacking

Stacking consists of combining several astronomical images. In order to do that, images have to be aligned one respect to each other.

There are 3 options to perform this task:

- DeepSkyStacker: it has a graphical interface that makes things easy to perform, however it works only under Windows. Tutorials on how to use it can be found in the website.

- Python-based: first you align all the images to a reference one with 'astroalign'. Then those images can be combined by adding them

  ```
  $ final_science = aligned_image_1 + aligned_image_2
  ```

  if you want to obtain a deeper image (e.g., for galaxies), or they can be combined by averaging them

  ```
  $ list = [aligned_image_1, aligned_image_2]
  $ final_science = np.mean(list, axis=0)
  ```

  for example when you have star images, e.g., a standard star. This method does not work for images downloaded from 'astrometry.net', since it only computes the astrometric solution, but it doesn't change the pixel position of objects.

- SWARP: this is the most precise method, but it requires images having the astrometric solution in their headers. Therefore, it requires as input images downloaded from 'astrometry.net' or that have been processed with SCAMP. SWARP manual can be found at https://www.astromatic.net/pubsvn/software/swarp/trunk/doc/swarp.pdf. Look for the configuration file (it should end with '*.swarp') and check that the 'COMBINE' parameter is set to 'Y'. Check the manual for the other parameters meaning and values. If you have images from 'astrometry.net', you can create an ASCII file with image paths (as in SCAMP) and then you run SWARP with

  ```
  $ swarp image_list -c hpp.swarp
  ```

  to obtain as output a coadded (combined) image. Exposure times, gains etc. will be updated accordingly. If you have catalogues from SCAMP, then it will output a .head file that can be read by SWARP by setting the 'HEADER_ONLY' parameter to 'Y' (see manual for more details).

2.6 Photometric Calibration

The photometric calibration requires the image of a standard star that has gone through all the data reduction steps we mentioned so far. The theory behind the photometric calibration is very well explained in http://slittlefair.staff.shef.ac.uk/teaching/phy241/lectures/L07/index.html. Since you need to perform aperture photometry on your standard stars (or galaxies or HII regions, depending on the experiment), you can use the routines provided by the 'photutils' Python package https://photutils.readthedocs.io/en/stable/aperture.html. By following the two tutorials, it should be quite straightforward to obtain the magnitude zero-point of the image.

Credits

Credits to Uwe Schimtt for the virtual environment installation description and to Jason Fitzpatrick for how to install Python on Windows.

Python Software Foundation. Python Language Reference, version 2.7.

Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Anaconda, Nov. 2016. Web. <https://anaconda.com>.

Homebrew was created by Max Howell. Website by Remi Prevost, Mike McQuaid and Danielle Lalonde.

SAOImage DS9 development has been made possible by funding from the Chandra X-ray Science Center (CXC) and the High Energy Astrophysics Science Archive Center (HEASARC). Additional funding was provided by the JWST Mission office at Space Telescope Science Institute to improve capabilities for 3-D data visualization.

E. Bertin and S. Arnouts. SExtractor: Software for source extraction. A&AS, 117:393-404, 1996.

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

Credits to http://deepskystacker.free.fr/english/index.html.

van Dokkum 2001, PASP, 113, 789, 1420 http://adsabs.harvard.edu/abs/2001PASP..113.1420V.

Credits to https://github.com/toros-astro/astroalign.

This research made use of Astropy,[2] a community-developed core Python package for Astronomy (Robitaille et al. A&A 558, A33 (2013)).

Phoutils: DOI 10.5281/zenodo.1340699

Credits and many thanks to Stuart Littlefair http://slittlefair.staff.shef.ac.uk.

[2]http://www.astropy.org