Exercise 3: Two-dimensional aperture photometry (part 1)

Astronomical Data Processing Course, 2022, Copenhagen University, Lise Christensen, Luca Izzo, Christa Gall

Learning outcomes

After completing this exercise the student will be able to:

- Perform aperture photometry on two-dimensional imaging data
- Determine the zero-point of the instrument

Python codes needed for this exercise

Photutils (see https://photutils.readthedocs.io/en/stable/ where installation instructions are described at the beginning of the documentation at that webpage).

lmfit (which will be used for least square minimisation).

imexam (see https://imexam.readthedocs.io/en/0.9.1/index.html)

Additional packages must be installed if missing in the python distribution of each single user (one of the most frequent missing packages is astropy or scipy). During the execution of scripts/commands, these missing packages will be called by the same code.

Background reading material:

1) Handbook of CCD astronomy, Howell, Chapter 5, pages 102-123
2) The Statistics of Centroiding: Study note by Peter Jakobsen on Absalon
3) The Statistics of Aperture Photometry: Study note by Peter Jakobsen on Absalon

References to photometric standard star fields

- Landolt, A , 1992: https://ui.adsabs.harvard.edu/abs/1992AJ....104..340L/abstract
- Stetson, P.B. 2000: https://ui.adsabs.harvard.edu/abs/2000PASP..112..925S/abstract

Data:

Use the processed data from Exercise 2, namely the images in the three filters bands: B, V, and R.

The goal is to measure the instrumental magnitude for the standard star in the three filters B,V,R, and then estimate the zero-point of the instrument for each filter. The output from this exercise will be used in the following exercise.

A typical run

Photometry consists in the measure of the amount of flux emitted by an astronomical source as a function of the wavelength, and as observed by a detector, in this case a CCD camera. As you have already seen in previous exercises, astronomical images can be considered as 2-dim arrays. Python permits to manipulate easily 2-dim arrays (in principle, it can work on n-dim arrays) and we will use it to perform photometry on some test images.

The process of aperture photometry consists of few consequential steps, which must be executed in the following order:

1. Determine the seeing to get a value for the FWHM, useful to identify the best aperture to use for the photometry.
2. Identify the astronomical sources in a given image. For this step we will use pre-compiled *photutils* functions.
3. Performing the aperture photometry using *photutils*.
4. Optionally, we will also introduce the concept of differential photometry, using external catalogs.

For this step, we need to have a clear understanding of the World Coordinate System, useful for the definition of source positions and aperture dimensions.

As an exercise, you will be asked to perform aperture photometry using multiple aperture radii, to verify our initial choice for the aperture using the seeing value. The results obtained from the above procedure will be a table of instrumental magnitudes, which must be converted to physically meaningful magnitudes using the information from observations of a photometric standard star.

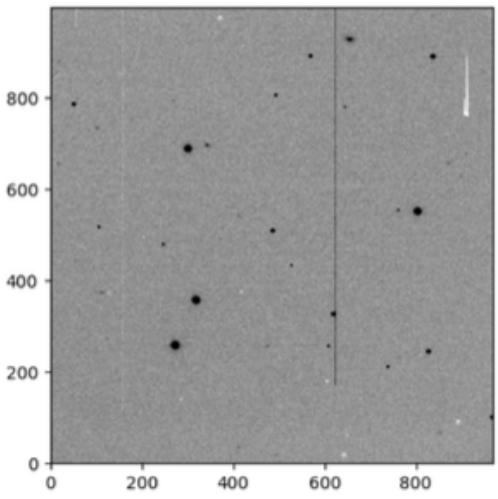

Fig. 1: The R-filter standard star image, as shown with *plt.imshow().* Using the interactive plotting window, you can determine a first guess for the standard star position in pixels. Alternatively, you can manually open the file with a ds9 window.

Step 1 - Determination of the FWHM

To estimate the FWHM of an image, we will use the package *Imexamine*, which we have already used in the previous exercise. *Imexamine* will allow us to determine the FWHM of all point sources detected in a given astronomical image by using the *line_fit* method. This step will then permit us to select the radius (in pixel) for the aperture that we will use for the determination of the magnitude of these same sources.

To perform the photometry measurement we must know the position (the centroid) of our astronomical source with a great accuracy. Then, we are going to implement a 2-dim fitting procedure that will permit to infer the FWHM value as well as the position of our desired source with a great precision. Of course we should provide an initial guess for the location of our standard star: this is feasible by plotting the standard image array with *plt.imshow*

```
>>>plt.imshow(std_reduced_image,cmap='Greys',origin='lower',clim=(300,
500))
>>>plt.show()
```

We have now all that we need to infer the FWHM of the sources in our image, e.g. the position of the standard star source. To infer the average FWHM value, we must:

1) fit the source with a given specific function (that in general is a 1-dim Gaussian or a 1-dim Moffat function);

2) estimate the FWHM, while functional form varies according to the function used to fit the detected sources in the image. In the case of a *Gaussian1D* function, we only need to use the Sigma parameter, which is related to the FWHM through the following formula

*FWHM = Sigma * np.sqrt(8.0 * np.log(2.))* [Eq. 1]

While in the case of a *Moffat1D* function, we must consider both alpha and gamma parameters, given that in this case the profile is described by the function:

*FWHM = gamma * 2 * np.sqrt(2**(1/alpha) - 1)* [Eq. 2]

The most time-expensive procedure is given by the first step above, the estimate of the fit-function parameters. To this aim we will still use the *Imexamine* package, given that it permits to implement several operations (and iterate them) on astronomical images. As an example, to fit one detected source, we need first to create a variable and initialise it on our standard image. Then, according to the pixel coordinate, stored in a variable defined by you, use a specific fit function to determine the main parameter(s) that will be used to estimate the FWHM. A starting point, to be used to develop further your code, is the following:

```
>>> from imexam.imexamine import Imexamine
>>> plots=Imexamine()
>>> plots.set_data(DATA)
```

For a single detected source, and a Gaussian fit function, the main code to use is:

```
>>>plots.line_fit(xcentroid,ycentroid,form='Gaussian1D',
genplot=False)
```

While for a Moffat fit function, we have:

```
>>>plots.line_fit(xcentroid,ycentroid,form='Moffat1D', genplot=False)
>>>plots.line_fit(xcentroid,ycentroid,form='Moffat1D', genplot=False)
```

The output from each *line_fit* command is the value of the Sigma parameter which is equal to the standard deviation (*stddev_0.value*) in the first case, and of the alpha and gamma parameters (*alpha_0.value* / *gamma_0.value*) in the second case of a Moffat function fit. The values can be extracted :

```
>>>plots.line_fit(xcentroid,ycentroid,form='Gaussian1D',
genplot=False).stddev_0.value
```

Ex. 1 - *Estimate the FWHM of one or more of the stars in the standard star field using both the Gaussian1D and the Moffat1D functional form. Do you find differences in the final FWHM ?*

Why is important to estimate the FWHM before measuring the photometry ?

In the next step, we will implement the aperture photometry method on our science image. The inferred magnitudes will strongly depend on the aperture (pixel or arcsec) used to collect the counts from each astronomical source for which we are going to measure its magnitude. Given that the sky condition are not the same, i.e. the background sky emission change with time during the night, and the seeing is then variable all nights, the estimate of the FWHM will permit us to determine the best aperture value to use for our aperture photometry.

Ex. 2 - *Determine the seeing of the night from the inferred FWHM value*

Step 2 - Aperture photometry on standard star image and zero-point determination

With a CCD, we are going to measure what is called the "instrumental magnitude" of a given astronomical source. This quantity is determined from a measure of the amount of photons (ADU) coming from our source of interest and it is related to the visual magnitude by the following equation:

$$m=-2.5\log_{10}(I)+ZP \qquad \text{[Eq.3]}$$

Where *I* is the amount of photons collected per second in your instrument. *ZP* is the zero-point of the instrument for the specific filter used, and it is a constant value that allows to infer the visual magnitude *m* of an astronomical source. It is computed through the measurement of an instrumental magnitude of standard stars, e.g. objects for which we know in advance the magnitudes in the specific filter used for the observations.

An additional correction is included to the above equation that includes effects of the atmospheric absorption and extinction, as well as a colour-correction to include for the effect of differences in filter transmission from your data, and that use for the reference photometric standards.

We will start this part of the exercise with the measurement of the instrumental magnitude for a standard star in a given filter for which we know its magnitude. This operation will permit us to infer the zero-point of the instrument for those filters.

To estimate the instrumental magnitude we will use the method of the aperture photometry. The concept at the base of the aperture photometry is that we are going to compute the total amount of ADU that are included within a specific, generally circular, region used for this purpose. The radius of the circular region is given by the FWHM value that we have found before. Consequently, we can have different aperture values for different nights, if the corresponding seeing is quite different. This is the motivation of the determination of the FWHM explained before.

In addition to the circular aperture, used for the source, we will also need to implement an additional aperture, commonly called an 'annulus' that will be used to measure and correct for the underlying background where the source is located. The reason of this additional aperture is due to the possibility that:

1. The background is not uniform, or it was not subtracted correctly. Note that it is not necessarily needed to first subtract the background flux in the reduced 2D image before continuing with the next step of measuring object fluxes.
2. The source we are going to measure the magnitude is on top, or embedded within an extended source such as a galaxy (this is very common when we measure the magnitude of a supernova)

In the Howell book, it is well explained how to estimate the background during an aperture photometry operation. It is also well explained that it is preferable to select as the aperture radius the value corresponding to the seeing, or 1xFWHM (2.35 x sigma, estimated in the previous paragraph). Choosing the aperture sizes equal to the seeing, will optimise the S/N ratio of the photometry, but it will **not capture all the flux from the wings of the stars PSFs.** To do that, we instead need to choose an aperture that is of a sufficient size (typically 3 x FWHM), or alternatively include the method of using aperture corrections, as described on page 119 in Howells book.

The area of the background annulus region must be ~3 times the area of the aperture region used to measure the instrumental magnitude. For a more accurate estimate of the background flux, we must apply a median filter and use a 3-sigma cutoff method (sigma clipping) for the background region. This will give a more homogeneous distribution for the background counts, for each aperture, which is centered on the median background value with all pixels contained within +- 3 sigma.

All these operations can be implemented in a python code/pipeline using the *photutils* package. For this part of the exercise, we need to import the following functions from the *photutils* class

```
>>> from photutils import CircularAperture, CircularAnnulus,
aperture_photometry
```

The first two functions will be used to define the source and background regions, while with *aperture_photometry* we will measure the instrumental magnitudes. These two functions have two main parameters to be configured (three for the CircularAnnulus) and they are

1. The coordinate of the source we want to measure the magnitude
2. The dimensions (in pixel) of the aperture radius, or the inner radius of the annulus, for the CircularAnnulus
3. For the CircularAnnulus alone, the outer aperture radius

As standard star, we will use the PG 1323+06 star field, whose magnitudes can be found in the Landolt (1992) tables. Note that these tables report the magnitudes in the filter V, and the colours (V-R, R-I, B-V) that must be used to infer the magnitudes in the desired filter. A page dedicated to this standard star-field can be found at the following link

http://fcaglp.fcaglp.unlp.edu.ar/~egiorgi/cumulos/herramientas/landolt/pg1323-086.htm

Ex. 3 - *From the above link, determine the centre positions in pixels of the 3 standard stars. (Avoid 'star A' because it is has been found to be variable). You are going to use this information for the definition of the apertures and the magnitude measurement.*

The main function that provides the estimate of the instrumental magnitude is the *aperture_photometry.* It needs few more inputs to work correctly:

1. The 2-dim array containing the reduced image with the star(s) for which we are going to measure the magnitude
2. The object containing the circular apertures that we have defined before. The object can also be a list of apertures, as we will see later

These are the main inputs, but given that we work in a digitised array, a circular aperture won't include perfectly all the pixels that contains the light from the star but that are partially included in the circular aperture. For this reason we must use some pre-defined function. There are several options (and you can see a description in the dedicated page of the function:

https://photutils.readthedocs.io/en/stable/api/photutils.aperture.aperture_photometry.html#photutils.aperture.aperture_photometry)

We will use the `method='subpixel'` where pixels are divided into a number of subpixels, which are included or excluded by the aperture based on their centers. In this case, we must define an additional keyword, which defines the number of subpixels needs to be set

3) `method = 'subpixel'`
4) `subpixels` = a scalar, generally 5 is a good choice
5) finally, you can also provide your error image (= square root of the variance image), so that the aperture_photometry can give an estimate of the systematical uncertainty associated with the measurement of the instrumental magnitude.

```
>>>aperture = CircularAperture(standard_coo, aperture_size)

>>>phot = aperture_photometry(inputimage, aperture, method='subpixel',
error=errorimage, subpixels=5)
```

where `standard_coo` the coordinates of the star in pixels (i.e. an array with two numbers [x,y]).

The main output of the *aperture_photometry* function consists in a table containing all the information we need, e.g. the updated position of the star(s), the total amount of counts (in ADU) in the circular aperture and its error if you supply an error image.

```
 id xcenter  ycenter aperture_sum
    pix      pix       adu
--- -------- ------- ------------
 1 802.0845 550.935    539590.76
```

Ex 4: *From the known position of the 3 standard stars, compute the sum of the flux in the apertures. Choose an aperture that is of a sufficient size to capture all the flux from the stars, that is in this exercise we will not compute aperture corrections.*

What is the area of your chosen aperture?

What is the exposure time of the input image ?
What is the count rate (ADU/second) for the 3 stars ?

Additional columns can be included if we provide the error image, or if we give additional apertures (i.e an array) as input to the *aperture_photometry.* We can also define our own columns as we will see right now.

So far so good, but we have not used yet the circular annulus **background**. As mentioned before, we want to apply a median filter to the background pixel values, in order to exclude possible image defects (bad pixels, cosmic ray hits, trails...) and render the background estimate more homogeneous. This procedure is a bit tricky, given that we will have to do with "masks". These are images where some of the pixel intensity values are zero, and others are non-zero: Wherever the pixel intensity value is zero in the mask image, then the pixel intensity of the resulting masked image won't be considered as part of the background. Then, we will multiply the mask by the science image and then we will select only pixels with value > 0. This will result in a 2-dim array composed only of the annulus background. Finally, we will apply a median sigma clipping algorithm to estimate the median value and cut the count distributions by excluding values that are > 3 sigma from the median value. Given that this part needs some more support, a short example for a single source is reported below:

We will first define the background annulus

```
>>> annulus_apertures = CircularAnnulus(standard_coo, r_in=r_in,
r_out=r_out)
```

With `standard_coo` the coordinates of the star in pixels (i.e. an array with two numbers [x,y]), `r_in` and `r_out` inner and outer radii of the annulus aperture.

Then, create the mask, multiply by the science image and select only counts > 0

```
>>> annulus_masks = annulus_apertures.to_mask(method='center')
>>> annulus_data = annulus_masks.multiply(DATA)
>>> annulus_data_1d = annulus_data[annulus_masks.data > 0]
```

finally, we will apply our median sigma clipping algorithm and will obtain a median value for the background

```
>>> from astropy.stats import sigma_clipped_stats
>>> _, median_sigclip, _ = sigma_clipped_stats(annulus_data_1d)
>>> bkg_median = median_sigclip
```

When we will measure the magnitudes from the M67 cluster, you will need to implement this technique to all the sources detected in the image (next exercise). More details can be found at the following link

https://photutils.readthedocs.io/en/stable/aperture.html#sigma-clipped-median- within-a-circular-annulus

The background value obtained with the above method must still be normalised to the area of the circular aperture, before being subtracted from the total number of counts measured for our source. Once you will subtract the median background, you will have a corrected number of counts from the star. It is convenient to include these numbers in the output table provided by the 'phot' object.

```
>>> phot['annulus_median'] = bkg_median
# flux from the background inside the aperture of the star :
>>> phot['aper_bkg'] = bkg_median * aperture.area
# Now you can subtract the background flux from the aperture:
>>> phot['aper_sum_bkgsub'] = phot['aperture_sum'] - phot['aper_bkg']
```

Ex. 5 - *With the knowledge acquired so far, determine the magnitude of the standard stars in all the filters (B,V,R). Use equation 3, and remember to compute the count rate, ie. counts/seconds. Initially, we do not know what the zero-point value is, but a reasonable number from modern optical instruments is around 25 magnitude, so assume in the first place that ZP = 25 in all filters.*

Measure photometry errors

Using the rules of error propagation in summed apertures, one can derive an equation that is used for measuring the errors of the magnitudes:

$$\Delta m = 1.0857 \frac{\sqrt{Var_{aperture}}}{\sum_{aperture} f} \qquad \text{[Eq. 4]}$$

where $Var_{aperture}$ in the nominator is the sum of the variance values in the aperture that is identical in size to that in the science image. The sum in the denominator is the aperture sum you measure from question no 4.

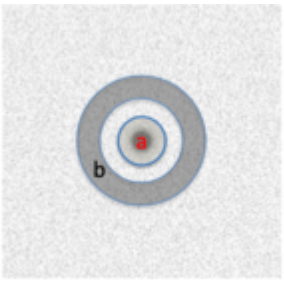

Fig. 1 : Illustration of the aperture and annulus. the aperture includes 'a' pixels, and the annulus used for estimating the background flux has 'b' pixels.

The Variance aperture in Eq. 4 is :

$$Var_{aperture} = \sum_{aperture,a} Var + \frac{n_a^2}{n_b} \langle Var \rangle \qquad \text{[Eq. 5]}$$

where n_a and n_b are the number of pixels in the aperture and the annulus and <*Var*> is the average variance level in the annulus.

Ex. 6 - *Compute magnitude errors for your 3 standard stars using equation 4 and 5. Use the error propagated image (variance image) or alternative make a simplified version of an error (or Variance) image. Use your apertures and annuli from the photometry above to propagate the uncertainty.*

Create a table with your measurements of the 3 standard star CCD magnitudes and magnitude errors for the 3 filters.

Photometric calibration

We are almost at the end, but before estimating the zero-point there are still a couple of further corrections to implement to our instrumental magnitude.

First, since the zero-point represents an absolute quantity, it must be normalised to a specific exposure time: not all the astronomical images have been obtained with the same exposure time, and the duration of an exposure affects the total number of counts. Imagine to observe the same star with the same detector, same filter but different exposure times: the aperture photometry on the longer exposed image will provide a larger number of counts, falsifying the zero-point value. For this reason, we must divide the total number of counts for the exposure time of the image.

Second, the magnitude estimated using the number of counts must be corrected for atmospheric extinction to get the correct instrumental magnitudes. The following relations are approximately correct:

$$b_{inst} = b_{CCD} - 0.25 * airmass \qquad \text{[eq. 6]}$$

$$v_{inst} = v_{CCD} - 0.13 * airmass$$
$$r_{inst} = r_{CCD} - 0.09 * airmass$$

The information about the airmass, as well as about the exposure time and the filter, can be found in the "header" of each science image. With astropy it is easy to get these values: you should already know how to do that, but probably you do not know what keyword refer to what. Some useful tips:

Quantity	Keyword
Exposure time	"EXPTIME"
Airmass	"HIERARCH ESO TEL AIRM START"
Filter	"HIERARCH ESO INS FILT1 NAME"

The zero-point is then immediately obtained using the following formulas (for the case of a R-filter image)

$$r_{inst} = -2.5 * log(counts / exptime) - 0.09 * airmass \qquad \text{[eq. 7]}$$

$$ZP_R = R_{STD} - r_{inst}$$

Ex. 7 *- It's your turn, now, to compute the zero-point for all the B,V,R filters. If you assumed a zero-point earlier on as in Ex. 5, then ZP_R will give a correction to your assumed zero-point.*

The results above are valid only if your observations are carried out with *exactly* the same filters that were used for to observe the stars in the photometric tables. This is rarely the case, and we must correct for small differences in between the filters. Hence, we need another correction to convert from the instrument magnitudes to magnitudes in the standard system. The general expression for converting from instrumental magnitudes to that in the standard system are for our 3 filters:

$$B_{std} = b_{inst} + c_{b2}(B-V)_{std} + c_{b3}$$
$$V_{std} = v_{inst} + c_{v2}(V-R)_{std} + c_{v3} \quad \text{[Eq 8]}$$
$$R_{std} = r_{inst} + c_{r2}(V-R)_{std} + c_{r3}$$

So replacing the instrument magnitudes from above gives:

$$B_{std} = b_{CCD} - 0.25*Airmass\_b + c_{b2}(B-V)_{std} + c_{b3}$$
$$V_{std} = v_{CCD} - 0.13*Airmass\_v + c_{v2}(V-R)_{std} + c_{v3} \quad \text{[eq.9]}$$
$$R_{std} = r_{CCD} - 0.09*Airmass\_r + c_{r2}(V-R)_{std} + c_{r3}$$

Here, capital letters refer to the magitudes in the standard star system, i.e. equal to those provided in reference tables that you can find online. The colours $(V-R)_{std}$ and $(B-V)_{std}$ are calculated from the standard stars magnitudes, if they are not directly reported in tables.

To find the best fit parameter ($c_{b2}, c_{v2}, c_{r2}, c_{b3}, c_{v3}, c_{r3.}$), you need to define a set of equations, and pass the measured magnitudes ($b_{CCD}, v_{CCD}, r_{CCD}$) for 3 stars, and their errors to a minimisation function. Here is an example (for fitting the B-band) of how this can be done (see also exercise3.py on Absalon).

Let 'data' be the photometric magnitudes of the 3 standard stars, so replace the '?' with the correct numbers.

```
# Standard star catalog
BV = np.array([ ?, ? ,? ])
VR = np.array([ ? , ? , ? ])
V= np.array([?,?,?])
data = (V,BV,VR) # a tuple of numpy arrays

# your derived CCD magnitudes and their errors for the 3 standard stars
mb= np.array([?,?,?])
mberr= np.array([?,?,?])

# define an object to pass parameter values to 'bfit' function in
exercise3.py

from lmfit import minimize, Parameters, report_fit

params = Parameters()
params.add('cb2', value=0.1)
params.add('cb3', value=0.1)

fitted_b = minimize(bfit, params, args=(mb, data, mberr))
report_fit(fitted_b)
```

Ex. 8 - *Compute the best-fit parameters and their errors for Eq. 9, which will complete the full equations to perform the photometric calibration.*

Create a table with your measured CCD magnitudes and their magnitude errors for the 3 standard stars.

Create a table with the standard star magnitudes (BVR magnitudes or colours).

Your python programme should include definitions of the 3 equations from [eq. 9]

Use the python module 'lmfit' with the option to perform a least square minimisation (with minimize) of the equation and give you best-fit values and errors for the parameters c_{b2},c_{b3}, c_{v2}, c_{v3}, c_{r2}, c_{r3}.

c_{b3}, c_{v3}, c_{r3} *are corrections to your assumed Zero-point. Your final zero-point will be* $ZP_b = 25 + c_{b3}$ *for the B-band.*

For your report, include a table with your best fit values. Compare also with the 'simple zeropoint' from Ex. 7.

Ex.9 (optional if you have more time): *As a final exercise, write a single pipeline that starts from the raw images, computes a single, average FWHM, perform aperture photometry, and provides the zero-point from each filter of your input images.*