lab_3_prelab

March 17, 2022

1 ASP3231 Laboratory 3: Shift, Scale and Combine images

In this week's laboratory we are going to shift, scale and combine images. Doing this is critical to produce combined images where the signal-to-noise is improved, poor quality data is rejected and object fluxes (measured in ADU) that can then be tied to calibrated magnitudes.

1.1 Prelab Questions

You are expected to read the lab notebook prior to the lab, and the Prelab Questions are designed to prompt you to consider what you have read. This will hopefully prepare you for the lab session and ultimately save time.

Question 1: Why do we subtract the background before combining images?

Question 2: Why is it often better to take 10 exposures of a science object and combine them instead of taking a single longer exposure?

Question 3: What is an advantage of a median combine over a mean combine?

Question 4: The traditional centroid is a flux weighted mean of the x-y pixels positions. Using online resources, find out what the astropy centroid_2dg function.

Question 5: Why is it important to visually inspect images with ds9? What sort of errors could visual inspection catch that may be missed by statistics calculated automatically?

Quation 6: We will be using for loops quite often in this notebook. Write a Python program to construct the following pattern, using a loop and just a single print statement: p py pyt pyth pytho python

Q1 - Because we have a reference for our background only valid for single science images, not combined science images.

Q2 - Because it's easier to take 10 separate exposures - the observation is less prone to disturbances like knocking/human error. If a 10 minute exposure is ruined, that's 10 minutes wasted and we need to take another 10 minute exposure. But if one of the 1 minute exposures is ruined throughout the course of taking 10 1 minute exposures, we can discard that exposure, waste 1 minute, and take another exposure.

This also transfers over to analysis after the fact - if you get back to the lab and discover that one of your 10 minute exposures has been ruined somehow, you can't discard 3 minutes' worth of that exposure and continue to analyse the data. If you have separate 1 minute exposures, you can still perform analysis.

Similarly, it ensures that all your data isn't ruined by anomalous events - like in Q3.

Q3 - A median combine is less prone to outliers. For example, in outside of visual wavelength observations, a streak of few pixels can be completely saturated by a cosmic ray of some sort. A median combine will discard the outlier pixels if those appear only in one exposure, whereas a mean will factor those in to produce a skewed combine.

At a less extreme scale, this logic also applies for pixels that have been disproportionately saturated by Gaussian noise purely as a result of probability.

Q4 - Calculate the centroid of a 2D array by fitting a 2D Gaussian (plus a constant) to the array - per the photutils documentation.

Q5 - Some large-scale qualitative human error like a telescope being knocked generating a junk image, or perhaps more subtly and likely - something like a dark exposure being marked as a science exposure due to a mistake in naming conventions or metadata assignment.

[ ]:
```
#Q6 - just for a bit of fun!
py_string = "python"

list(map(lambda i: print(py_string[:i+1]), range(len(py_string))));
```

```
p
py
pyt
pyth
pytho
python
```

1.2 End of Pre-Lab

2 Lab 3. Shift, Scale and Combine images

2.0.1 By the end of this lab, you should be able to:

- shift images **correctly**
- scale images and understand why you need to scale images
- understand several different methods used for combining images
- provide a combined image with improved signal-to-noise relative to the indiviudal exposures

2.0.2 Let's begin

Task 1. Import all the needed libraries. Can you identify a library or function we didn't use in the previous lab?

[ ]:
```
# Import various libraries and some specific functions. Note the additions␣
↪relative to the previous lab.

import numpy as np
import astropy
import ccdproc
from ccdproc import CCDData, Combiner
from astropy import units as u
from astropy.visualization import SqrtStretch
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from photutils import centroid_com, centroid_1dg, centroid_2dg
from photutils import CircularAperture
from photutils import aperture_photometry
from photutils import Background2D
from photutils import MedianBackground
from scipy.ndimage import shift
import gc
gc.enable()
```

Task 2. Our starting point is processed V-band images of NGC 2362 that have an exposure time of 30 seconds. Display one of these images using the ds9, and modify the scale using the minmax and zscale options.

Display that same image in this notebook and use the *vmin and vmax* parameters of function *imshow* to mimic the colour scale of ds9. Use the links below for more help:

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.imshow.html

https://docs.astropy.org/en/stable/visualization/normalization.html

[ ]:
```

```

Figure 1: proc_NGC_2362_V_30.000secs00009656.fits

Task 3. Let's create a collection of processed V-band images of NGC 2362 (called images).

[ ]:
```
images = ccdproc.ImageFileCollection(".",glob_include='proc_NGC_2362_V_30*')
```

Task 4. Let's now read these images into scim, using code adapted from the previous lab. (Note the use of PICTTYPE is somewhat redundant in this particular instance, but would catch misnamed images.)

[ ]:
```
scim
```

2.1 Shifting

Task 5. When we shift our images we do not want overwrite existing images, so we define new image names. Can you use the comments to describe what each line of the following code does?

```
[ ]: newname=[]
     for fn in images.files_filtered(PICTTYPE=1):
         newname.extend(["s"+fn])
     print(newname)
```

3 Don't forget to take notes in this notebook!

Task 6. The first image in our list, proc_NGC_2362_V_30.000secs00009656.fits, has a relatively bright star near x=2030, y=1360. We can make a copy of this image, subtract the background and then measure the position of the star using centroid_2dg on a section of the image. Run the following the code cell and comment on each line and in particular explain the last line in detail.

```
[ ]: temp=scim[0].copy()
     temp=temp-np.ma.median(temp)

     # Determine the centroid - note the python convention for x and y can be␣
      ↪flipped!
     x1, y1 = centroid_2dg(temp[1300:1400,2000:2100])
     print(x1+2000, y1+1300)
```

Task 7. Compare this bright star position from centroid_2dg with what you measure manually using ds9. Does your position agree within 1, 2 or 3 pixels of the output of centroid_2dg? If there's a difference, how could it result from how centroid_2dg and you measure the object position?

Task 8. Let's now implement this for all the processed V-band 30 second exposures of NGC 2362. What are xoffset, yoffset, xbox and ybox defining? Fill in the comments for the loop and run the code

```
[ ]: xoffset = 1960    # x edge of the box
     yoffset = 1300    # comment
     xbox = 100        # comment
     ybox = 100        # comment
     shiftx=[]         # comment
     shifty=[]         # comment

     for idx, thisimage in enumerate(scim):
         temp =
         temp =
         x1, y1 = centroid_2dg( temp[yoffset : yoffset + ybox, xoffset : xoffset +␣
      ↪xbox])
         print(x1 +xoffset, y1 + yoffset )
         shiftx.append(x1 + xoffset)
         shifty.append(y1 + yoffset )

     print(shiftx)    # What is this?
     print(shifty)    # What is this?
     print(shiftx[0]-shiftx) # What does this mean?
     print(shifty[0]-shifty) # What does this mean?
```

Task 9. Let's now determine the integer offsets that need to be applied to shift the star to the same x-y as those given in the first image. The first image has shifts of 0,0 by defintion. The second image has shifts of 2, 1 while the third image has shifts of 1,1.

Here's how you would write the x-y shifts for the first four images.

```
[ ]: shifts=[(0, 0), (2, 1), (1, 1), (43, -5)]
     print(shifts)
```

Task 10. Now implement the shifts for all the images. Can you use a loop to do this easily? (Hint, the np.rint function could be useful to implement integer shifts.)

```
[ ]:
```

Task 11. Let's implement the shifts and cross check the position of the star. Registration of the images to within 1 pixel will suffice for now (as the blurring by our atmosphere is much larger than one pixel). Run and comment on the code.

```
[ ]: # What is this code doing?
     xoffset=2000
     yoffset=1335
     xbox=60
     ybox=60

     # What is this code doing?
     for idx, thisimage in enumerate(scim):
         yxshifts=(shifts[idx][1], shifts[idx][0]) # Note the y-x convention being␣
      ↪used here and in the following command.
         temp = shift(scim[idx], yxshifts, order=0, mode='constant', cval=-1000)
         temp=temp-np.ma.median(temp)

         # What is this code doing?
         x1, y1 = centroid_2dg(temp[yoffset:yoffset+ybox,xoffset:xoffset+xbox])
         print(x1+xoffset, y1+yoffset, shifts[idx][0], shifts[idx][1])
```

Task 12. Once you are happy with the shifts, write out the shifted images to FITS files. While we are at it, we will subtract the sky background too (which will come in handy later). Run the code

```
[ ]: for idx, thisimage in enumerate(scim):
         yxshifts=(shifts[idx][1], shifts[idx][0])
         temp = CCDData(shift(scim[idx], yxshifts, order=0, mode='constant',␣
      ↪cval=-1000)-np.ma.median(scim[idx]), unit="adu")
         temp.header = scim[idx].header
         temp.write(newname[idx])
```

Task 13. Use ds9 to display these images, and check they have been successfully registered. A simple command line for this is:

ds9 sproc_NGC_2362_V_30.000secs000* &

The "blink frames" option in the frame meny could prove useful.

Also display the first and last frame in this notebook.

Once you are happy with these images, we can move onto the step to scaling the images before combining.

3.1 Scaling

Task 14. Create a collection of shifted images of NGC 2362 (called images) and read these images into scim.

[ ]:
```

```

Task 15. To determine scalings, we haved identified bright stars that are in all the images, and we will use the fluxes of these stars to see how the images have been dimmed (relative to a reference image) by passing clouds and scattering/absorption of light by our atmosphere.

Below we have defined the positions of three stars and defined circular 20 pixel apertures for photometry.

[ ]:
```
positions = [(1538.0,1488.0), (2269,1279), (1776,1047)]   #x-y notation
apertures = CircularAperture(positions, r=20.0)
```

Task 16. We can measure the counts of the three stars in the first image, and print them to the screen. This is aperture photometry, so it's adding up the flux (above the background) within some radius. Run the following code cell.

[ ]:
```
phot_table = aperture_photometry(scim[0], apertures)
print(phot_table)
```

Task 17. Now add aditional bright stars to this list, making sure they do not have bright neighbours and they aren't saturated in the CCD images (i.e. they don't peak at 60,000+ counts per pixel).

[ ]:
```

```

Task 18. Perform aperture photometry for the updated star list using the first image.

[ ]:
```

```

Task 19. Let's now do determine the counts (fluxes) for these stars in all of the images. Don't forget to use comments to describe what the following code is doing (and why it's being done).

[ ]:
```
phot_table=[]
for idx, thisimage in enumerate(scim):
    phot_table.extend([aperture_photometry(thisimage, apertures)])
    print(idx, phot_table[idx])
```

Task 20. To print the fluxes (no coordinates) for the stars in the first image (index=0), we can use the following command.

```
[ ]: print(phot_table[0]['aperture_sum'])
```

Task 21. We are going use the 7th image (index 6) as our reference image for the combine. What properties would we like a reference image to have (e.g. signal-to-noise, observing conditions)?

Task 22. What is the following series of commands doing?

```
[ ]: for idx, thisimage in enumerate(scim):
         print(idx)
         print(phot_table[6]['aperture_sum']/phot_table[idx]['aperture_sum'])
         print(np.ma.median(phot_table[6]['aperture_sum']/
       ↪phot_table[idx]['aperture_sum']))
```

4 Combine

Task 23. Happy with the scalings? How have you confirmed that they make sense (and is this documented here)? If you are happy with the scalings then let's combine the images together using two methods - averaging and median combine. Comment on the following code cell; what does minmax clip do?

```
[ ]: images = ccdproc.ImageFileCollection(".",glob_include = 'sproc_NGC_2362_V_30*')
     scim = [CCDData.read(fn) for fn in images.files_filtered()]
     for idx, thisimage in enumerate(scim):
         m = np.ma.median(phot_table[6]['aperture_sum'] /␣
       ↪phot_table[idx]['aperture_sum'])
         print(m)
         scim[idx] = scim[idx].multiply(m * u.adu)

     sci_average = ccdproc.combine(scim, method = 'average',dtype = np.float32,
                                   minmax_clip = True, minmax_clip_min = -500)
     sci_average.write("NGC_2362_V_average.fits")

     sci_median = ccdproc.combine(scim, method = 'median',dtype = np.float32,
                                  minmax_clip = True, minmax_clip_min = -500)
     sci_median.write("NGC_2362_V_median.fits")

     del(scim)
     collected = gc.collect()
     print('Check garbage collection', collected)
```

Task 24. How do the combined images visually compare?

Task 25. How does the photometry of the final product compare with the reference photometry? How do the statistics compare? Below we provide a sample code to print out the stats for the average stacked image. Perform the similar task for the median stacked image.

[ ]:
```
print('Average combine')
phot_table_average = aperture_photometry(sci_average, apertures)
print('Median pixel value', np.ma.median(sci_average))
print('Standard deviation', np.std(sci_average[400:500,400:500]))
print(phot_table_average)
print(phot_table[6]['aperture_sum']/phot_table_average['aperture_sum'])
print(np.ma.median(phot_table[6]['aperture_sum']/
 ↪phot_table_average['aperture_sum']))
```

Task 26. An alternative to the median and average combine is an average with min-max rejection, which rejects the n-highest and n-lowest pixel values before doing an average combine. Combine the images using this method and then visuall inspect them.

[ ]:
```
images = ccdproc.ImageFileCollection(".", glob_include = 'sproc_NGC_2362_V_30*')
scim = [CCDData.read(fn) for fn in images.files_filtered()]

for idx, thisimage in enumerate(scim):
    m = np.ma.median(phot_table[6]['aperture_sum']/
 ↪phot_table[idx]['aperture_sum'])
    print(m)
    scim[idx] = scim[idx].multiply(m * u.adu)

sci_minmax = ccdproc.combine(scim, method = 'average', dtype = np.float32,
                             minmax_clip = True,minmax_clip_min = -500,␣
 ↪clip_extrema = (3,3))
sci_minmax.write("NGC_2362_V_minmax.fits")
```

Task 27. Adapt the code from above to determine the statistics for the average combine with min-max rejection. How do the statistics for this combined image compare with the average and median compbined images?

[ ]:
```

```

Task 28. Don't forget to write your key **conclusions** in this notebook.

[ ]:
```

```