Observational Astronomy 2025
Assignment 1

Due: Friday 2 May 2025 at 17.00

In this assignment you will develop simple Python routines (based on `astropy`) to reduce simple optical imaging data. Refer to the "Lab 1" lecture notes for details on *why* each step is performed and what each step *does* to the data.

In this assignment (and this course), you'll want to become familiar with `astropy`, `https://www.astropy.org` and especially its FITS I/O implementation described at `http://docs.astropy.org/en/stable/io/fits/index.html`, and also the notes that are available from your ProgNum Course. Of course, NumPy will be invaluable throughout this course, so make sure you're comfortable with the way arrays work and can be manipulated in that package (see the help pages accessable through Jupyter Lab).

This assignment is to be done and submitted individually, *not with your group.* Talking through solutions with your colleagues is permitted, of course.

Submit a Jupyter notebook with your results. At each step, describe what you did in enough detail that someone else could follow it, show your code (or the appropriate command), and whenever a new image is generated, display a picture of the new image(s) in your notebook. You may need to change the limits in the display so that you (and we) can see (variations in) the images well.

1. First things first: copy the data for this assignment from `/net/vega/data/users/observatory/LDST/2222-22-22` to one of your local directories. This is a set of sample data taken at the Lauwersmeer Dark Sky Telescope (LDST). You almost certainly want to create a new directory. **NOTE:** Do **not** use your home directory for data reduction! Experience shows that you will run out of space quickly, and we can't run your code from there! Always use a **data disk** like `/net/virgo01/data/users/`<*your user name*>. Furthermore, use the full – not the relative – path to the files in your code.

2. (20 pts) To reduce CCD data, you need to know which frames are "bias" frames, which are "dark" frames, which are "flatfield" frames (and usually in which *filter* the flatfield was taken), and which are your "program" frames (i.e., the ones you really care about!). Make a table with the filename of each file, the object, the "observation type" (bias, dark, flat, etc.), and filter name. You'll have to list the appropriate **FITS header keywords**: i.e., OBJECT, EXPTIME, IMAGETYP, FILTER, etc... (Check which keywords are available in the data you are using and understand what they mean!)

   (Note: I urge you to read the header keywords for exposure time inside of your routine, as it will make your code easier to run on other data later!)

3. (20 pts) Next, make (and save) a *master bias* frame for these data using the *bias* frames you found in the previous step. As described in the accompanying lecture, your best bet here is to make a *median image* of the "stack" of bias frames. Display your master bias.

   Hint: the NumPy command `numpy.dstack` is your friend for median combining images.

4. For the Lauwersmeer Dark Sky Telescope we currently do not take *dark* frames, so in this case you do not need to make a *master dark*, as the exposures are short, and the dark current is very low. (For the Gratama telescope, you *do* want to make a *master dark* frame for your data, but not for this assignment.)

5. (30 pts) The last calibration images that you'll need to make is the *master flatfield* frame for each filter. You want to make a *normalized* master flatfield frame, in which the average value of the pixels is 1, because you need to take out the *relative* quantum efficiency ("gain") variations of each pixel. The steps to do this are slightly more complicated here: first subtract the master bias frame and the master dark *scaled to the exposure time of each flatfield frame* from each flatfield frame. Then divide this (bias- and dark-) corrected flatfield frame by its mean (or, better, median) to produce a normalized flatfield. Then you can median combine these frames to make a normalized master flatfield. Display your master flatfields for each filter present.

   Make sure that you inspect the individual flatfield frames, and discard any that are inappropriate (for example, that the pixels are saturated).

6. (30 pts) Almost there! Reduce your program images (everything left over, basically) by subtracting the master bias frame, and then dividing by the normalized master flat field (for the appropriate filter). Display your images for each filter.