Calibration Process for SMHASH data

Vicky Scowcroft

June 3, 2015

Draft last edited June 3, 2015

1 Python Scripts

For several of the calibration steps I have made python scripts available through the SMHASH github repository. Information about the scripts is given in the README.md file of the repository. I have indicated in the text when scripts are available and which one you should be using, and describe the steps you should do if you want to perform the calibration steps by hand.

The python install I am using is the Ureka distribution (the one that you probably installed with IRAF if you're using a Mac) from STScI. If you have questions or comments about the python scripts please let me know.

Throughout this document I will use the omegaCen_1 data as an example.

2 Initial Setup

The SMHASH data comes in two forms: The individual BCDs and the mosaicked images. It is more than likely that you will be working with mosaicked images. If the file names start with SPITZER_I and end with cbcd.fits then it is a BCD file. If it starts with a descriptive name, like Sgr or CSS then it is a mosaicked image.

In either case, the images you have received will be in flux units. For daophot to work properly you need to convert the images into units of 'counts'. This can be done using something like imarith in IRAF, or whatever program you choose. The equation you need to use is:

$$C = I \times T/F \tag{1}$$

where C is the image in counts, I is the original image, T is the value of the FRAME_EX keyword in the image header, and F is the value of the FLUXCONV keyword in the image header.

The new images are now in the correct units so that you'll get reasonable uncertainties out of daophot. You should run daophot, allstar and allframe on all your images.

Image type	Pixel size	A1	Inner sky	Outer Sky
Non–mosaicked	1.2″	3	3	7
Mosaicked	0.6″	6	6	14

Table 1: Aperture sizes required for calculating aperture correction

3 Calibration Starting Point

You should work such that you have each field in a separate folder. For example, if you have multiple fields of the same object (field 1, field 2) keep them in separate folders for now. The scripts I have developed do a lot of the calibration automatically and assume that everything in the folder is of the same field, and get confused if that isn't the case. Make it easy for them and yourself!

At this point you should have PSF magnitudes from allframe, done on the 'counts' images. We will refer to this as photometry file $\textcircled{1}$. We will start working on epoch 1 of your data.

Note: log always means log10.

4 Aperture correction

For this step we work on the original images that are in flux units, i.e. before we converted them using the fluxconv and exptime keywords.

Do aperture photometry on a selection of uncrowded stars in the image. The best stars for this are probably the PSF stars. See Table 1 for the appropriate aperture sizes.

The photometry from the flux image is photometry file $\textcircled{2}$ Now use daomatch and daomaster to match the results from the counts file to the flux file. Do a first pass with daomatch, then edit the match file. Copy and paste the first line to the second line (No transformation should be required, they're the same image but you're looking at different stars) and edit the file name of the second line. Then run daomaster, and get the raw output.

Calculate the aperture correction required:

$$\Delta M = M_{\textcircled{2}} - M_{\textcircled{1}} \tag{2}$$

Aperture correction script: calculate_aperture_correction.py

```
python calculate_aperture_correction.py omegaCen_1__e1_3p6um_dn.raw
```

For this script the input file (a .raw file from daomaster) must have the aperture photometry in column 4 and the allframe photometry in column 6. This means that there should have been two files in the .mch file that went into daomaster, with the aperture photometry on line 1 and the allframe photometry on line 2.

Apply aperture correction script: apply_aperture_correction.py. (*This script also does the zero–point correction described in Section 5 and the correction to the standard aperture described in Section 6.*)

Pixel Size	[3.6]	[4.5]
0.6 arcsec	18.80	18.31
1.2 arcsec	17.30	16.81

Table 2: Zero Points used in images with different pixel sizes.

```
python apply_aperture_correction.py omegaCen_1__e1_3p6um_dn.alf A C
```

where A is the aperture correction calculated by the previous script, and C is the channel (1 or 2).

The output of this script will be a file called omegaCen_1__e1_3p6um_dn.apc which is calibrated to the 10–native–pixel IRAC standard aperture and is on the correct zero–point (i.e. all calibration steps through Section 6 have been applied).

If you are not running the scripts:

Apply the aperture correction to **all the stars** in allframe file $\textcircled{1}$.

$$M_{\textcircled{3}} = M_{\textcircled{1}} + \Delta M \tag{3}$$

5 Zero–point correction

Zero point correction script: This step is performed by the aperture correction script. I provide details of the steps here for reference, but if you've run apply_aperture_correction.py you **do not** need to do this step.

The next step is to put the photometry on the correct zero–point.

Daophot assumes an instrumental magnitude scale such that

$$M = 25 - 2.5\log(Counts) \tag{4}$$

To put the IRAC photometry on the correct scale we must use the correct zero–points, by subtracting the 25 in Equation 5 and adding in the the corresponding value in Table 2.

6 Conversion to the standard aperture

Standard aperture conversion script: This step is performed by the aperture correction script. I provide details of the steps here for reference, but if you've run apply_aperture_correction.py you **do not** need to do this step.

The IRAC standard aperture is a 10 native pixel (12″) aperture with the sky measured between 12 and 20 native pixels. There is a standard conversion between the small aperture we have used and the standard aperture which is taken from the IRAC handbook.

First convert your $M_{\textcircled{3}}$ magnitudes to fluxes:

$$Flux = 10^{(-M/2.5)} \tag{5}$$

Then multiply the flux given in equation 6 by 1.128 (channel 1) or 1.127 (channel 2). Convert this result back to magnitudes using equation 6

$$M_{\textcircled{4}} = -2.5\log(Flux \times A_0) \tag{6}$$

At this point, your epoch 1 data is almost calibrated.

7 Calibrate all epochs

In this step all epochs are calibrated to the same zero–point. Create a .mch file containing the epoch 1 .apc file and the .alf files for all other epochs for a *single channel*. Run this through daomatch and daomaster, with the following input parameters for daomaster (Assuming 12 epochs):

- Minimum number, minimum fraction, enough frames: 12,1,12
- Maximum sigma: 99
- Degrees of freedom: 4

and requesting the following output:

- Assign new star IDs? Yes
- A file with raw magnitudes and errors? Yes
- A file with the new transformations? Yes
- Simply transfer star IDs? Yes

The new transformation file may not be strictly necessary as you already have a good transformations file from allframe, but I find it good practice to make an updated version.

The .raw and .mtr files will be used as input to the next script, which calculates the offset between the first epoch photometry and every other one.

Calibrate all epochs script: calculate_each_epoch_offset.py

```
python calculate_each_epoch_offset.py omegaCen_1_3p6um.raw C
```

where omegaCen_1_3p6um.raw would be replaced by the name of the .raw file you just created and C is the channel number (1 or 2). Don't worry that the .raw file is formatted such that the magnitudes are spread over three lines for a 12 epoch data set – the program is designed for this. The output from this script is a .off file for each channel/epoch combination.

Here we are essentially repeating the process where you calculated the aperture correction, this time calculating the offset between epoch 1 and all the other epochs. Epoch 1 is your reference. The vast majority of the stars in your frame will not vary so you can use them as local standards. Calculate the average difference between the stars in epoch$_N$ and epoch 1 and apply this correction to epoch$_N$. Repeat for all epochs.

You now have (almost) calibrated photometry for all epochs.

8 Location Correction

The final step is the location correction. This step MUST be performed last as it is image dependent. It is a very small effect. I also haven't found a smart way to regex the correction image names from the input photometry file names yet, so you have to run this for each image separately.

Location correction script: location_correction.py

```
python location_correction.py omegaCen_1__e1_3p6um_dn.off
                                         omegaCen_1__e1_correction_3p6um.fits
```

where the .off file is the output from calculate_each_epoch_offset.py and the .fits file is the location correction image that will be in the folder with the image data.

For each magnitude in the file, the script converts it back to flux using equation 6. At the position of each star, it pulls out the pixel value of that position in the correction image. Each correction image is slightly different so make sure you are using the correct one.

The flux is multiplied by the correction value:

$$F_{loc-cor} = F \times f_{location} \tag{7}$$

then is converted back to magnitudes. This script must be run separately for all images. You now have calibrated magnitudes for all stars in all images!

The output from this script is a .cal file. This is your final, calibrated photometry!