Palomar 200" LFC Reduction Cookbook

Micaela Bagley

December 2014

1 Introduction

This describes a set of Python codes that are written for the purpose of reducing imaging data observed with the Large Format Camera (LFC) on the 200" at Palomar Observatory. This Cookbook explains how to reduce the data, fit them with WCS solutions, stack the images, calibrate the photometry, and create a final catalog of sources matched to the WISP catalogs.

2 Requirements

Untar the directory of Python scripts in a convenient directory:

```
unzip palomarLFC.zip
```

Add the following to your ~/.bashrc and source it:

```
export PYTHONPATH="$PYTHONPATH:[path_to_scripts]/palomarLFC/"
export PATH="$PATH:[path_to_scripts]/palomarLFC/"
```

The following Python packages are required and can be installed automatically by running:

```
pip install -r requirements.txt
```

- `numpy`
- `pyraf` (requires `IRAF`)
- `astropy`
- `scipy`
- `matplotlib`
- `PyPhot`[1]

Additional requirements:

- `astrometry.net` (see below)
- `Source Extractor`

3 Reduction

3.1 Organize the data

The first step is to sort the data from a night of observations. The `extract_header` module in `sort_images.py` will produce a file listing all images in a directory as well as their image type, object name, exposure time, filter, airmass, and binning (1×1 or 2×2).

```
usage:  sort_images.py [-h] dir
```

`dir` − the directory for which to extract information from all image headers.

[1]Some of the IDL AstroLib photometry algorithms translated to Python. Written by David Jones, Johns Hopkins University

You should have a directory for each night of science data (e.g. `23mar12/`) and a directory for each of the types of calibration images to be used in reducing the science data (a directory each for biases, flats, and if necessary, darks). The directories for calibration images may be wherever you like (inside the directory of science images, some random place on your computer, etc.) as long as the calibration files have their own directories.

Inspect all calibration images and put any biases you don't want to use inside a directory called `unused/` within the biases directory. The same goes for the flats (and darks). The naming convention for directories is necessary if you wish the reduction pipeline to automatically produce a log file.

3.2 Reduce a night of data

`run_reduction.py` is the wrapper for all reduction modules. There are several arguments and options available.

```
usage:  reduction.py [-h] [--biasdir BIASDIR] [--darkdir DARKDIR] [--flatdir FLATDIR]
              [--SaveSteps] [--LogSteps LOGSTEPS] [--DarkSubtract] [--UseBias USEBIAS]
              [--UseDark USEDARK] [--UseFlat [USEFLAT [USEFLAT ...]]]  directory binning
```

`directory` – Directory to be reduced. Ex: 23mar12

`binning` – Binning of data to be reduced. ('binned' or 'unbinned')

`--biasdir BIASDIR` – Directory containing the biases. Default is `biases/`.

`--darkdir DARKDIR` – Directory containing the darks. Default is `darks/`.

`--flatdir FLATDIR` – Directory containing the flats. Default is `flats/`.

`--Overwrite` – Overwrite files after each reduction step. If not set, images are saved as separate files, with strings ('bs', 'ds', 'ff') appended to filenames to indicate reduction steps taken.

`--LogSteps LOGSTEPS` – Create a log file to record the creation of calibration files and the science images included in reduction.

`--DarkSubtract` – Dark subtract the data. Dark subtract is not performed by default.

`--UseBias USEBIAS` – Use a previously-created Master Bias.

`--UseDark USEDARK` – Use a previously-created Master Dark.

`--UseFlat` – Use a previously-created Master Flat. List a Flat for as many filters as necessary.

Paths to the directories containing the calibration files should be *relative* paths. For example, consider a three night observing run in March of 2012. All data for this run are in a directory called `march2012/`, and we wish to reduce the data in `march2012/23mar12/`. Biases and flats from this run are in `march2012/bias/` and `march2012/flats/`, respectively.

To reduce binned data and output a logfile called `23mar12.log`:

```
run_reduction.py --biasdir march2012/bias/ --flatdir march2012/flats --LogSteps
march2012/23mar12.log march2012/23mar12 binned
```

Combined calibration images (e.g. `Bias_unbinned.fits`, `Flatg_unbinned.fits`) are written to their respective directories. To reduce unbinned data, use a previously-created Bias, and output a log file:

```
run_reduction.py --UseBias march2012/bias/Bias_unbinned.fits --flatdir march2012/flats
--LogSteps march2012/23mar12.log march2012/23mar12 unbinned
```

To reduce binned data, use previously-created Bias and Flats, and output a log file (if there is more than one flat, list each flat that you want to use separated by a space):

```
run_reduction.py --UseBias march2012/bias/Bias_binned.fits --UseFlat
march2012/flats/Flatg_binned.fits march2012/flats/Flati_binned.fits --LogSteps
../march2012/23mar12.log march2012/23mar12 binned
```

4 Astrometry & Alignment

After reducing the data, sort all images by WISP parallel field with one field per directory. Output files will be named after the directory. For example, images in `Par300/` will be aligned and combined to form `Par300_[filter].fits`. There are several fields that were observed on multiple nights. To avoid overwriting files, give each FITS file a unique name (include the date observed, for example).

Astrometry and image alignment is performed in 4 steps:

1. `astrometrynet.py`
2. `fix_astrometry.py`
3. `combine.py`
4. `align_images.py` (`imalign.pro`)

4.1 `astrometrynet.py`

The first step is to get a rough (though very good) approximate WCS solution for each image in a WISP parallel field using the *Astrometry.net* software package. `astrometry.py` builds the command that will run the *Astrometry.net* software on each image in a directory.

`usage: astrometrynet.py [-h] [--useSE] wispfield`

`wispfield` – the WISP parallel field whose images are to be fit with WCS solutions. This must match the name of the directory that contains all relevant FITS files.

`--useSE` – option to use *SExtractor* to detect sources in each image rather than *Astrometry.net*'s bundled "images2xy" program.

Solved files are renamed `<base>_solved.fits`. These are the original FITS images with headers updated to contain the WCS solutions. The original FITS image as well as all of *Astrometry.net*'s auxiliary output files are moved to a directory called `astrometric_solns`. See Appendix B for a list of the output files.

Astrometry.net software can take a long time if it must run through every available index file in order to find one that covers the image. In order to expedite the process, `astrometrynet.py` tells *Astrometry.net* to:

- skip running the FITS files through a "sanitizer", which cleans up non-standards-compliant images
- query only index files within 1° of the RA and Dec in the image headers
- expect image pixel scales in the range $0.1 - 0.45$"/pix

The range in the last step is enough to cover both binned and unbinned images and ensures that only index files of the proper scale are used in determining the fit. (This step probably does not affect much as that's a rather large range.)

`astrometrynet.py` prompts you to check your `/tmp` directory for any `[tmp.sanitized.*]` files. *Astrometry.net* can create loads of these files if it runs into any problems solving for an image. `/tmp` is hard-wired into *Astrometry.net*, but `/tmp` is pretty small on the physics network.

4.2 `fix_astrometry.py`

The *Astrometry.net* software does a great job with the astrometry near the center of the chip. Most WISP fields were observed on the top 1/3 of the chip. At the bottom and edges of the images, the WCS gets further off. `fix_astrometry.py` uses IRAF's CCMAP task to calculate a second order solution that takes care of the small distortions and rotations at the edges. To do so, it compares the positions of objects in the Palomar images with those in the SDSS catalog. Download the SDSS catalog in fits format[2] and put it in the directory `wispfield`. The LFC detector covers a lot of area, so a search radius of 5 arcmin (or more) is best. `fix_astrometry.py` expects the file to be called `result.fits`.

This script uses an IRAF task. Create a symbolic link to your `login.cl` file before running `fix_astrometry.py`.

`usage: fix_astrometry.py [-h] wispfield`

`wispfield` – the WISP parallel field whose images are to be fit with improved WCS solutions. This must match the name of the directory that contains all relevant FITS files.

`Required files`:
`result.fits` – the SDSS catalog for the WISP field.

`fix_astrometry.py` first runs *SExtractor* on all images and outputs a catalog and segmentation map named `<base>_astr_cat.fits` and `<base>_astr_seg.fits`, respectively. For each image, it then matches the sources detected in the Palomar image with the SDSS catalog. A file to be used with CCMAP, `SDSS.match`, is created listing the Palomar x, y coordinates with the SDSS RA and Dec for each source. Finally, it launches CCMAP, which allows the user to interactively delete obvious residuals in the astrometric solution. The `<base>_solved.fits` headers are updated with the new and improved WCS.

4.3 `combine.py`

Once all images have corrected WCS solutions, images from like filters are stacked using IRAF's IMCOMBINE task. Before combining, the user should inspect all images and remove any for which the seeing is significantly worse. Use IRAF's IMEXAM to check the FWHM of a few point sources across each image. Make sure to check the observing logs as well for any mention of clouds or other problems. `combine.py` will prompt the user to stop and perform this inspection if they haven't already.

`usage: combine.py [-h] [--rejection none,minmax,ccdclip,crreject,sigclip,avsigclip,pclip]`
`[--combine average,median,sum] wispfield`

`wispfield` – the WISP parallel field for which to stack image. This must match the name of the directory that contains all relevant FITS files.

`--rejection` – the rejection algorithm to use. Default is `crreject`, which rejects only positive pixels and is appropriate for cosmic ray rejection.

`--combine` – the type of combining operation to perform. Default is `median`.

See the IMCOMBINE help page for more information about parameter options.

[2]Download the catalog from `http://skyserver.sdss3.org/dr10/en/tools/search/radial.aspx` in fits format

For each filter, `combine.py` determines which images need to be combined and prepares all parameters and files for use with IMCOMBINE. Multiplicative scale factors are calculated for each image from the airmass (`[wispfield]_[filter].scale`). The scale factor $= 10^{0.4\ \kappa_F\ X}$, where κ_F is the airmass coefficient calculated for the Palomar site[3] and filter F. Additive zero level shifts are calculated from the median value for each image. All zero corrections are done with respect to the first image, so the images are sorted such that the lowest sky is listed first.

The readnoise is necessary for IMCOMBINE's sigma clipping routines, and can be estimated by binning the counts from an averaged bias frame. The mean of the resulting histogram is the bias offset and the width of the distribution is σ_{ADU} = readnoise/gain = FWHM. The user should include in each WISP field directory the master bias from each night the field was observed, and `combine.py` will estimate an average readnoise to use for all images.

Finally `combine.py` reports the number of images available in each filter and asks the user to choose the number of low/high pixels to reject or the number of pixels to keep (depending on the rejection algorithm used).

IMCOMBINE uses the WCS in the headers to calculate and apply offsets to each image so they are properly aligned before stacking. Combined images are called `[wispfield]_[filter].fits`. Sigma images are also created (`[wispfield]_[filter]_sig.fits`), and the log file for the combination is called `[wispfield]_imcombine.log`.

4.4 `align_images.py`

If there are 2 filters observed that night, `combine.py` will import `align_images.py` to align those images using the WCS in the headers and IRAF's task WREGISTER.

5 Calibration

For each field, the Palomar photometry is calibrated against the SDSS catalog (`result.fits`). `calibrate_sdss.py` calculates both a zero point shift and a color term and saves them for use in calibrating the final catalog.

```
usage:  calibrate_sdss.py [-h] wispfield
```

`wispfield` – the WISP parallel field for which to calibrate photometry. This must match the name of the directory that contains all relevant FITS files.

SExtractor is run in dual image mode on the combined Palomar images with 1σ detection and analysis thresholds and a minimum area for detection of 5 pixels. Such low thresholds are used to ensure that as much light as possible is included in each source's automatic magnitude. The Palomar catalogs are then match with the SDSS catalog using a $0.5''$ matching threshold.

The zero point shift, $zp = m_{SDSS} - m_{Palomar}$ is calculated for all sources, and those outside of 68% of the mean zp are excluded. There is a residual slope visible in a plot of zp that depends on color (top two rows of Fig. 1). This dependence is especially prevalent in the g band. Color terms to correct for this dependence are determined by fitting a line to the array of zero point shifts. Outliers (plotted in red) that are too far from the line in the y-direction are removed, and a new line (plotted in black) is fit to the remaining points.

[3]Taken from `http://www.ifa.hawaii.edu/~rgal/science/dposs/ccdproc/ccdproc_node3.html`, where $\kappa_g = -0.152$, $\kappa_r = -0.094$, and $\kappa_i = -0.07$.

The Palomar photometry is then calibrated by:

$m_{calibrated} = m_{Palomar} + \alpha_0\ (g-i)_{Palomar} + \alpha_1$,

where α_0 is the slope of the best-fit line (the color term) and α_1 is the offset (the zero point shift). Several plots are then created to check the quality of the calibration (rows $3-4$ in Fig. 1). The last row shows the residuals of the calibrated data. For perfectly calibrated photometry, there would be no remaining color dependence and the solid line would have a slope of zero.

5.1 1 Palomar Filter, 1 WFC3 filter

Fields that were observed with only one SDSS filter require a second filter for determination of the color terms. For this we use the WFC3 UVIS filters that are available for the field. Possible color combinations include ($g-$F814W), (F606W$-i$), ($g-$F600LP), and (F475X$-i$). Use `calibrate_sdss_hst.py` in place of `calibrate_sdss.py`.

6 Final Catalog

`make_final_catalog.py` runs *SExtractor* a final time with detection and analysis thresholds of 2.2σ so that all sources are detected at $\geq 5\sigma$. The zero point shift and color terms calculated by `calibrate_sdss.py` are applied to all Palomar photometry. Palomar sources are matched to the WISP catalog with a 0.5" matching radius. The final catalog includes both Palomar and WISP RA and Dec and photometry from all available filters.

```
usage:  make_final_catalog.py [-h] wispfield
```

`wispfield` – the WISP parallel field for which to construct the final catalog. This must match the name of the directory that contains all relevant FITS files.

A Installing *Astrometry.net*

Designate a directory for installation of *Astrometry.net* and its requirements. In the following example, the installation directory is `/home/bagley/Software`.

(Thanks to Michael Gordon (UMN-MIfA) for these installation instructions, which are also provided as a text file (`astro_setup.txt`)

A.1 CFITSIO

CFITSIO is a library of C and Fortran subroutines for reading and writing data files in FITS data format, available from NASA's High Energy Astrophysics Science Archive Research Center (HEASARC).

```
# mkdir ~/Software/cfitsio
# cd ~/Software/cfitsio
# wget ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/cfitsio_latest.tar.gz
# tar xzvf cfitsio_latest.tar.gz
# cd cfitsio
# ./configure --prefix=/home/bagley/Software/cfitsio
# make
# make install
```

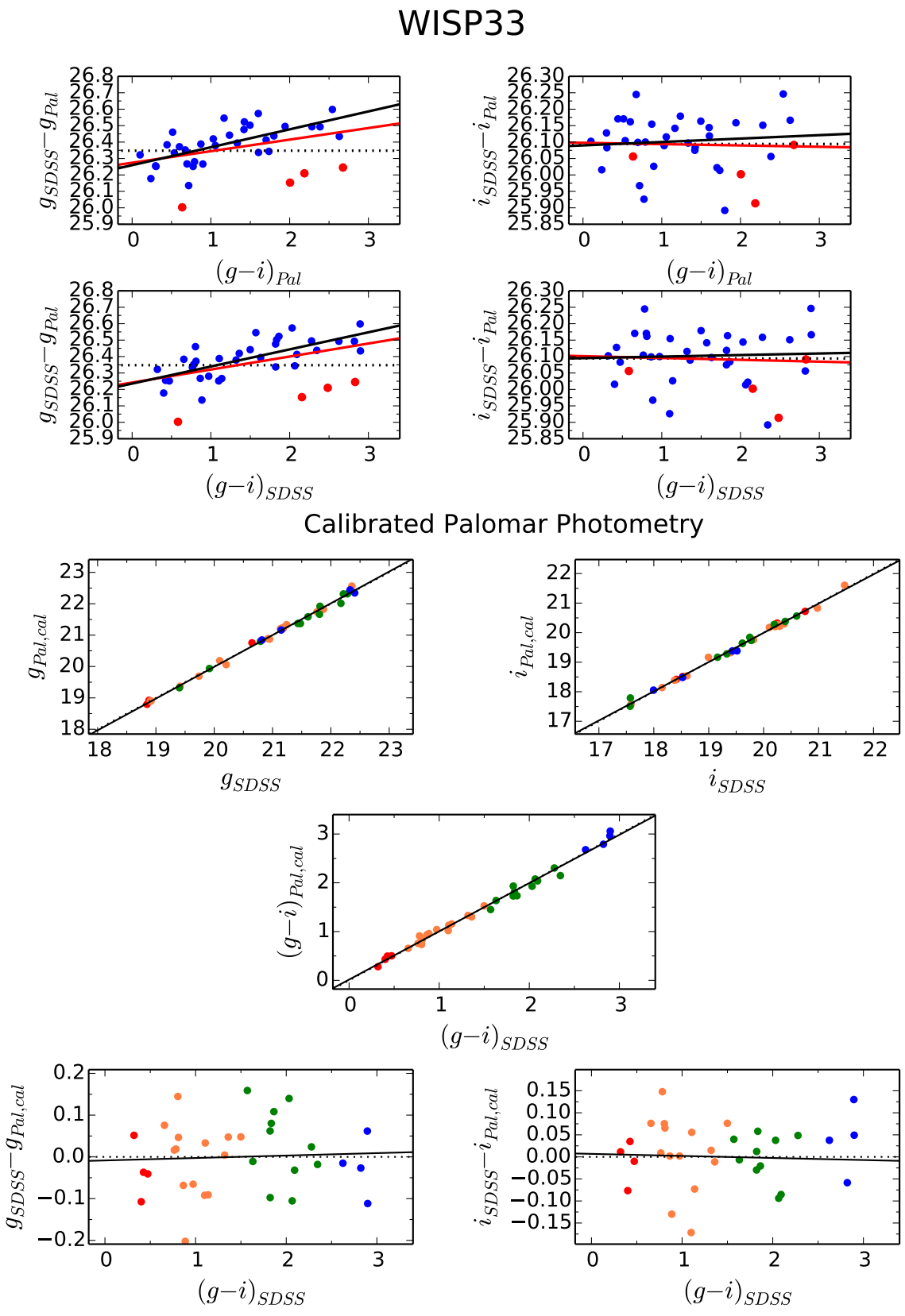

Figure 1: An example set of plots created by `calibrate_sdss.py` used to check the quality of the calibration. Zero point shifts as a function of Palomar instrumental color (top row) and SDSS color (2^{nd} row). In the bottom three rows, points are color-coded by their SDSS $(g-i)$ color. The bottom row shows the residuals of the calibrated photometry.

Add the following to your ~/.bashrc and source it:

```
# export PKG_CONFIG_PATH=/home/bagley/Software/cfitsio/lib/pkgconfig
```

Run the following commands, if they print something, you're good:

```
# pkg-config --cflags cfitsio
# pkg-config --libs cfitsio
```

A.2 *Astrometry.net*

If you are running Python in a Virtual Environment, make sure you are in your virtualenv before completing the next steps.

```
# mkdir ~/Software/astrometry
# cd ~/Software/astrometry
# wget http://astrometry.net/downloads/astrometry.net-0.46.tar.bz2
# tar xjvf astrometry.net-0.46.tar.bz2
# cd astrometry.net-0.46
# make
# make py
# make extra
# make install INSTALL_DIR=/home/bagley/Software/astrometry
```

Add the following to your ~/.bashrc and source it:

```
# export PATH="$PATH:/home/bagley/Software/astrometry/bin"
```

In your virtualenv, run astrometry.net as:

```
# solve-field --no-plot img.fits .....
```

A.3 Index Files

Astrometry.net requires index files, processed from an astrometric reference catalog such as USNO-B1 or 2MASS. Pre-cooked index files built from the 2MASS catalog are available here. Use the `wget` script to download the full catalog of index files, requiring about 10G of space. Alternatively, use the healpix png's to determine in which tiles your fields of interest reside and download only the relevant index files.

The software looks for index files in `~/Software/astrometry/data`. If you don't have enough space there, symlimk the `data` directory to their actual location.

```
# cd ~/Software/astrometry
# rmdir data
# ln -s /other/thing/data data
```

B *Astrometry.net* Default Outputs

`<base>.wcs`	–	a FITS WCS header for the solution
`<base>.new`	–	a new FITS file containing the WCS header (renamed `<base>_solved.fits`)
`<base>-indx.xyls`	–	a FITS BINTABLE with the pixel locations of stars from the indx
`<base>.rdls`	–	a FITS BINTABLE with the Ra,Dec of sources extracted from the image
`<base>.axy`	–	a FITS BINTABLE of the sources extracted, plus headers that describe the job (how the image is going to be solved)
`<base>.corr`	–	a basic FITS header
`<base>.solved`	–	exists and contains (binary) 1 if the field solved
`<base>.match`	–	a FITS BINTABLE describing the quad match that solved the image

These may be useful if the user ever wishes to check the WCS solutions. To keep *Astrometry.net* from creating these files, add `--wcs 'none'`, etc., to the command. Output options are: `--wcs`, `--new-fits`, `--keep-xylist`, `--rdls`, `--pnm`(?), `--corr`, `--solved`, `--match`, respectively. (I have not tried suppressing the creation of these files. See *Astrometry.net*'s README for more information.)