Palomar 200" LFC Reduction Cookbook

Micaela Bagley

December 2014

1 Reduction

1.1 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] [--extract_header] directory
```

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

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.

2 Astrometry & Alignment

WCS

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

2.1 `astrometry.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:  astrometry.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 **??** 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, `astrometry.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.)

2.2 `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.

`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[1] 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.

2.3 `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.

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

`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.
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[2] 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`.

2.4 `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.

3 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.

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

Table 1: *SExtractor* Parameters

Parameter	Value
DETECT_MINAREA	5
THRESH_TYPE	RELATIVE
DETECT_THRESH	2.2
ANALYSIS_THRESH	2.2
DEBLEND_NTHRESH	32
DEBLEND_MINCONT	0.005
WEIGHT_TYPE	None
PHOT_APERTURES	30
PHOT_AUTOPARAMS	2.0, 3.5
PHOT_AUTOAPERS	0.0, 0.0
GAIN	GAIN × (total exptime)
BACK_SIZE	64
BACK_FILTERSIZE	3
BACKPHOTO_TYPE	LOCAL
BACKPHOTO_FILTERSIZE	24

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. **??**). 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.

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. **??**). 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. The Palomar photometry is then calibrated by:

$$m_{calibrated} = m_{Palomar} + \alpha_0 \ (g-i)_{Palomar} + \alpha_1, \tag{1}$$

3.1 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`.

4 Final Catalog

If both filters are present, SE is run in dual image mode with the i band used for detection. SE parameters are listed in Table 1. SE is run with a zero point magnitude of 0.0, so that the zero point calculated in §**??** can be applied to the photometry.

`AUTO` magnitudes are calibrated according to eqn. (1). We do not perform an aperture correction on the `AUTO` photometry. We checked the curve of growth for a variety of sources and could not determine a single correction factor for all types of sources. The percentage of light enclosed by the Kron radius depends on the Sersic index. (See, for example, Graham & Driver (2005).)

The Palomar and WISP catalogs (fin_F110.cat, fin_F140.cat, fin_F160.cat) are matched with a matching threshold of 075.

`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.