Manual for the illumination correction in THELI

Dominik Klaes

Argelander Institute for Astronomy
Auf dem Hügel 71
53121 Bonn
Germany

Contact: dklaes@astro.uni-bonn.de

March 10, 2014

Contents

1 Main script **4**
1.1 Filtering 7

2 Fitting data **9**

3 Calculations after fitting **10**

4 Checkplots **12**

5 Applying illumination correction **13**

6 Duration **15**

1 Main script

This manual is describing the illumination correction code in THELI in detail in chronological order. All scripts contain detailed information about the required command line arguments and the history of changes. On github, every single change is also covered with a small comment. As far as possible, all steps are using multiprocessing features and are instrument independent (required information are taken from the THELI camera config file).
The needed command line arguments for the main script (illum_correction.sh, written in bash) are (in this order):

MAIND:	Main directory
STANDARDD:	Directory to files used for zeropoint calculation / illumination correction from main directory
FILTERNAME:	Used filtername, e.g. r_SDSS
SOLUTION:	In THELI three different fits for zeropoint estimation are available (the number in brackets corresponds to the given argument): **(1)** 3 parameter fit (zeropoint, extinction and color). **(2)** 2 parameter fit (zeropoint and color, extinction fixed to -0.10mag). **(3)** 1 parameter fit (only zeropoint, extinction fixed to -0.10mag, color fixed to 0.05mag).
EXTENSION:	In THELI all files have an extension depending on reduction steps. This is needed to find all files.
FILTER:	Used filter in reference catalog, e.g. r.
MODE:	Depending on the zeropoint calibration, an illumination correction can be done by using all data from one run ("RUNCLAIB") or each night individually ("NIGHTCALIB").
COLOR:	Used color index, e.g. gmr for $g - r$.

Table 1.1: List of command line arguments for main script

Example: ./illum_correction.sh /data/KIDS_V0.5/run_1/ STANDARD_r_SDSS r_SDSS 2 OFCS r RUNCALIB

Then, according to section 1.1, several filtering limits can be given.

MINOBJECTS:	Minimum number of objects on one chip. If less, then a warning will appear. Default value: 0

CUTS:	Gives the order and kind of filtering. Default value: "RESMEAN"
LOWERCUTPERCENT:	Gives the percentage of objects not to be used in the lower range. A value of e.g. 0.1 corresponds to 10% of objects in the lower range not being used. Default value: 0.1
UPPERCUTPERCENT:	Gives the percentage of objects not to be used in the upper range. A value of e.g. 0.1 corresponds to 10% of objects in the upper range not being used. Default value: 0.1
LOWERCUTRESABS:	Gives the lower residual cut. For a value of e.g. -0.2 this means that all objects with a residual less than -0.2mag are not used. Default value: -0.2
UPPERCUTRESABS:	Gives the upper residual cut. For a value of e.g. 0.2 this means that all objects with a residual larger than 0.2mag are not used. Default value: 0.2
LOWERCUTMAG:	Gives the lower magnitude cut. For a value of e.g. 10 this means that all objects with a magnitude less than 10mag (meaning brighter) are not used. Default value: 10
UPPERCUTMAG:	Gives the upper magnitude cut. For a value of e.g. 25 this means that all objects with a magnitude larger than 25mag (meaning fainter) are not used. Default value: 25
SIGMAWIDTH:	Gives the width of sigma clipping. A value of 1 means that a 1σ-area around the mean is used. Default value: 1
LOWERCUTRESMEAN:	Gives the lower magnitude cut w.r.t. the mean of the dataset. For a value of e.g -0.2 and a mean of 0.1(mag) this means that all objects with a residual smaller than -0.1mag are not used. Default value: -0.2
UPPERCUTRESMEAN:	Gives the upper magnitude cut w.r.t. the mean of the dataset. For a value of e.g. 0.2 and a mean of 0.1(mag) this means that all objects with a residual larger than 0.3mag are not used. Default value: 0.2

Table 1.2: List of filtering arguments for main script

Furthermore several (sanity) checks are done in the beginning and during filtering.

They cover all problems occurred during development but might not be complete!

- **Number of command line arguments:** In the current configuration, exactly eight arguments have to be given (cf. table 1.1). If more or less are given, the script ends with an error.

- **Runmode:** As written in table 1.1, the script needs to know in which mode (RUN or NIGHT) the illumination correction has to be done. If this arguments does not match with one of the two just mentioned, the script will end with an error.

- **Folder presence:** If an old illumination correction is detected, it is deleted with a warning message.

- **Camera information:** It is checked if the `CHIPGEOMETRY` is set in the camera configuration file. From this several information are extracted: `ROWMAX` (the number of chips in x-direction), `COLUMNMAX` (the number of chips in y-direction), `PIXX` (the number of pixels in x-direction), `PIXY` (the number of pixels in y-direction), `PIXXMAX` (the maximum number of camera pixels in x-direction, calculated via `ROWMAX`$\cdot$`PIXX`) and `PIXYMAX` (the maximum number of camera pixels in y-direction, calculated via `COLUMNMAX`$\cdot$`PIXY`). If `CHIPGEOMETRY` is not set, the script ends with an error.

- **Catalog presence:** Checks if there are LDAC catalogs available at all. These contain only those objects which were matched with the reference catalog. If not available, the script will end with an error.

- **First filtering:** The first filtering keeps only objects with a magnitude less than 99mag and larger than -9999mag in all corresponding bands (observed filter and both color filters). Here, the convention is used that the magnitude of objects with no match in the reference catalog is set to 99mag. Furthermore the colorterm has to be larger than -10mag but smaller than 10mag. If after this step no objects are left, the script will end with an error.

- **BADCCD flag filtering:** All objects which have the BADCCD flag set (BADCCD=1), e.g. due to a problematic chip, are sorted out. Afterwards it is checked if still enough objects are present. If not, the script will end with an error.

- **Photometric calibration file presence:** If the zeropoint calibration file is missing (for the entire run in "RUNCALIB" mode or for one of the nights in "NIGHTCALIB" mode), the script ends with an error.

- **Second filtering:** After filtering as described in section 1.1, the number of objects per chip is checked again with respect to "MINOBJECTS" as described in table 1.2. If too few objects are left, the script will raise a warning, naming also the chip number.

After these pre-checks and filtering, the filtered LDAC catalog is used as input by the fitting algorithm (illum_correction_fit.py, written in Python). A detailed explanation can be found in section 2.

After the fitting procedure, several values such as e.g. mean, variance and standard deviation are calculated. More information can be found in section 3.

Afterwards, several checkplots as described in section 4 and FITS correction files are created (illum_correction_contourplot_fitfunction.py, written in Python).

All scripts are highly optimized and, where possible and reasonable, they also use multiprocessing features. E.g. filtering objects and creating statistics is done chip by chip (so they run in parallel), but the needed time is about a few seconds in total so with multiprocessing this step would not speed up significantly. As a counterexample, creating all checkplots in section 4 takes about six minutes with multiprocessing (splitting up each file on one processor calculating there subplot by subplot). Furthermore, all scripts are easily expandable for new features e.g. other filtering techniques or checkplots.

1.1 Filtering

With given residuals, it is necessary to keep only objects within a certain range and to reject outliers. These can be caused by e.g. wrongly matched objects in the reference catalog and result in wrong magnitudes. For this purpose, the residual mean (equation 1.1), variance (equation 1.2) and standard deviation (equation 1.3) are calculated:

$$\bar{\epsilon} = \frac{1}{n}\sum_{i=1}^{n}\epsilon_i \tag{1.1}$$

$$\mathrm{Var} = \frac{1}{n}\left(\sum_{i=1}^{n}\epsilon_i^2 - \frac{\left(\sum_{i=1}^{n}\epsilon_i\right)^2}{n}\right) \tag{1.2}$$

$$\sigma = \sqrt{\mathrm{Var}} \tag{1.3}$$

It is possible to choose between the following techniques or a sorted combination of them:

- Fixed residual limits: Fixed limits are a specific residual range, e.g. only objects with $-0.2\mathrm{mag} < \epsilon < 0.2\mathrm{mag}$ are kept. Asymmetric boundaries are also possible. Especially non-photometric nights can be excluded by this (keywords are LOWERCUTRESABS and UPPERCUTRESABS).
- Residual limits w.r.t. the mean: These limits are a specific residual range which are calculated with respect to the mean, e.g. for a value of -0.2 and a mean of 0.1(mag) only objects with $-0.2\mathrm{mag} + 0.1\mathrm{mag} < \epsilon < 0.2\mathrm{mag} + 0.1\mathrm{mag}$, so $-0.1\mathrm{mag} < \epsilon < 0.3\mathrm{mag}$ are kept. Asymmetric boundaries are also possible.

Especially non-photometric nights can be excluded by this (keywords are LOWERCUTRESMEAN and UPPERCUTRESMEAN).

- Magnitude cut: An upper and lower magnitude cut can be given to exclude e.g. very faint and/or very bright objects. With this, only objects with a minimum and/or maximum S/N can be chosen (keywords are LOWERCUTMAG and UPPERCUTMAG).
- Percentage: Only a fixed percentage of objects is kept, e.g. for a value of 0.2 the upper and lower 20% are deleted. Again, asymmetric boundaries are possible (keywords are LOWERCUTPERCENT and UPPERCUTPERCENT).
- Sigma clipping: Using the calculated standard deviation, only objects with a residuum smaller than this value with respect to the mean $\bar{\epsilon}$ are kept (keyword is SIGMAWIDTH).
- Iterative sigma clipping: An iterative sigma clipping can be achieved via using the SIGMA argument several times. At the moment for all sigma clippings the same sigma width has to be used.

In some special cases, e.g. if one of the observations was taken during non-photometric conditions, there will be a lot of objects outside these ranges. Depending on the data, the illumination correction is also able to correct for this.
Also, it is very important how the cuts are chosen. The overall best solution for KiDS observations seems to be a residual cut of $\pm$0.2mag w.r.t. the mean (about 10% of the stars are now marked as "outliers"). For comparison, a 10% cut (lower and upper) plus afterwards a 3σ clipping is used in the estimation of the zeropoint. The reason for this is that in the latter case only the stars with precise magnitude measurements are kept to calculate a precise zeropoint. To get a usable illumination correction, not only these precise measurements are wanted, additionally also the magnitudes that are incorrect because of the non-uniform illumination of the camera are wanted. This means that the conditions for the ZP estimation are stricter than those for the illumination correction. In both cases non-photometric observations have to be excluded.

2 Fitting data

The Python script illum_correction_fi.py reads the filtered data from the previous step, fits a model via a χ^2 method and saves the resulting model prefactors to a text file. The coveriance matrix is also saved to a text file.
The needed command line arguments are:

-i:	Input file containing the data to be fitted
-t:	LDAC table name containg all data
-p:	Output path where the coefficient and coveriance matrix files shall be saved to.

Table 2.1: List of command line arguments for fitting script

This script uses a χ^2 method for fitting. As fitting function,

$$\epsilon = Ax^2 + By^2 + Cxy + Dx + Ey + F\,[chip] \tag{2.1}$$

is used, implemented in the code as

$$\epsilon = Ax^2 + By^2 + Cxy + Dx + Ey + \sum_{i=1}^{N} \delta_{iz} Fi. \tag{2.2}$$

Here, x represents a NumPy array with all x-coordinates, y an array with all y-coordinates and z an array with the corresponding information on which chip the object is located. That means for a given x, y, z triple, that F will only count where i=z, e.g. for z=1 (object is on the first chip), equation 2.2 reads as

$$\epsilon = Ax^2 + By^2 + Cxy + Dx + Ey + \underbrace{\delta_{11}}_{=1} F1 + \underbrace{\delta_{21}}_{=0} F2 + \underbrace{\delta_{31}}_{=0} F3 + ... \tag{2.3}$$

$$= Ax^2 + By^2 + Cxy + Dx + Ey + F1. \tag{2.4}$$

Unfortunately, Python does not have an implemented δ-function, so this had to be programmed by hand. This was done via the condition

$$\mathrm{int}\,(i - z) == 0 \tag{2.5}$$

As initial guess, all prefactors are set to 0.0 and all data points get an error calculated via Gaussian error propagation. At the moment the magnitude error of the observed objects and its counterpart from the reference catalog, the errors on the magnitude zeropoint, the extinction coefficient, the color term coefficient and color term are used. The error on the airmass is set to 0.0 because for this error no value is available or can be guessed.

3 Calculations after fitting

The fitting results are first used to correct the data on catalog basis via

$$m_{app,corr} = m_{app} - \epsilon \tag{3.1}$$

with ϵ defined as in equation 2.1. Also the residuals after fitting, including errors (via Gaussian error propagation), are calculated accordingly.
Additionally, the center position of the illumination correction can be obtained by the derivative of equation 2.1 via

$$\frac{\partial \epsilon}{\partial x} \stackrel{!}{=} 0 = 2Ax + Cy + D \tag{3.2}$$

$$\frac{\partial \epsilon}{\partial y} \stackrel{!}{=} 0 = 2By + Cx + E \tag{3.3}$$

Solving equation 3.2 for y and 3.3 for x:

$$x = -\frac{2By + E}{C} \tag{3.4}$$

$$y = -\frac{2Ax + D}{C} \tag{3.5}$$

Inserting result 3.5 into equation 3.4 and this afterwards again into 3.5 yields:

$$x_{max} = \frac{CE - 2BD}{4AB - C^2} \tag{3.6}$$

$$y_{max} = \frac{2AE - CD}{C^2 - 4AB}. \tag{3.7}$$

Afterwards, several statistics of the residuals before and after fitting are calculated for each individual chip and for all chips combined:

- mean according to equation 1.1
- minimum
- maximum
- number of data points
- variance according to equation 1.2
- standard deviation according to equation 1.3

- number of objects that are compatible with 0 within a 1σ-error
- percent of objects that are compatible with 0 within a 1σ-error (from 0 to 1)
- center position of the illumination correction (maximum of the polynomial) according to equation 3.6 and 3.7 including errors.

The following values are calculated only for the entire camera for the ellipse:

- length of the minor axis
- length of the major axis
- ellipticity
- numerical ellipticity
- Rotation w.r.t. the y-axis, counted clock-wise (CW)
- number of pixels that have a correction value between (at the moment) -0.015 and 0.0mag. These values are chosen such that in all illumination corrections, at least for KiDS observations, a certain number of pixels fulfil these requirements.
- last value, given in percent (from 0 to 1).

4 Checkplots

5 Applying illumination correction

With the FITS files created in the last step, they have to be applied to the reduced data like SCIENCE, SCIENCESHORT and/or STANDARD images. This task is done by another bash script (illum_apply.sh). Also, this script contains detailed information about the required command line arguments and an history of changes. On github, every single change is covered with a small notice. The needed command line arguments are (in this order):

MAIND:	Main directory
APPLYD:	Directory where the files are that need to be corrected
STANDARDD:	Directory to files used for zeropoint calculation / illumination correction from main directory
FILTERNAME:	Used filtername, e.g. r_SDSS
EXTENSION:	In THELI all files have an extension depending on reduction steps. This is needed to find all files.
MODE:	Depending on the zeropoint calibration, an illumination correction can be done by using all data from one run ("RUNCLAIB") or each night individually ("NIGHTCALIB").
NPROC:	Number of maximum parallel processes. This number should not be too high due to too high I/O. Recommended value: 16
ILLUMDIR:	If the illumination correction was done before and saved in another directory, as it is done with bias and flat images, this argument is different from STANDARDD.

Table 5.1: List of command line arguments for applying script

Furthermore several (sanity) checks are done in the beginning and during filtering. They cover all problems occurred during development but might not be complete!

- **Number of command line arguments:** In the current configuration, exactly eight arguments have to be given (cf. table 5.1). If more or less are given, the script ends with an error.
- **Folder presence:** If already corrected files are detected, the script requires to delete those files and ends with an error.
- **Runmode:** As written in table 5.1, the script needs to know in which mode (RUN or NIGHT) the illumination correction was done. If this arguments does not match with one of the two just mentioned, the script will end with an error.

- **Illumination correction presence:** It is checked if for all nights FITS files are present. If not, the script will end with an error.

Afterwards a list with files is created and distributed to NPROC number of processes. The old files are moved to a backup folder named EXTENSION_IMAGES.

6 Duration

Important for such a correction is the duration, especially for testing different magnitude cuts. The script needs about 10 minutes for a single average run to filter, plot, fit and create the correction images per band. The time to correct the images is very sensitive to the number of images and the speed of the used hard drive. Tests showed that the average time with 32 parallel processes on a raid is about one image per second per process. For a representative run, 1632 files (51 images with 32 chips each) have to be corrected, resulting in a duration of about a minute. In total, the illumination correction needs less than 15 minutes to correct an entire run (for one band) from the beginning until the end. If the correction mode is changed from run based to night based, the first part scales linearly while the second part stays constant.