GCX User's Manual

Radu Corlan

Version 0.9.7 November 25, 2005

Contents

Chapter 1

Introduction

The previous version of GCX , CX was written to control the newly designed `cpx3m` ccd camera. Once the basic camera control functions were running, it was easy to add some LX200 control functions, so that the telescope could be pointed at various objects without having to switch applications.

Having telescope control and image acquisition integrated into one program makes the following step obvious: after entering goto/get commands over several cold nights, one wants to automate the process—especially if he observes a large number of fields every night (as when doing variable star work).

The fact that the author's telescope doesn't point precisely doesn't help automation. So the ability to check/correct the pointing becomes essential. `cx` first got the ability to read star information from the GSC and overlay it on the images; that eases visual checks (one doesn't need maps anymore) but still is one step short of full automation.

Finally, when reliable field matching was implemented in GCX , it became possible to make the program fully automatic. In the current version, GCX can run through a list of observations completely unattended, and only stops if clouds roll in.

As it happens, field matching and image processing are also essential steps for CCD photometry. Over the time, the photometry functions of GCX have expanded continuously up to the point where they contribute the largest part of the program. It is currently possible to reduce photometric data frames in a completely automatic fashion, and perform color transformations, transformation coefficient fitting and all-sky reduction with relative ease.

1.1 Features

Image handling

- Open/save 16-bit FITS image files;

GCX uses floating-point images internally, so other FITS formats are easy to add;

- Zoom/Pan images, adjust brightness/contrast/gamma in an intuitive way, appropiate for astronomical images;
- Convert FITS files to 8-bit PNM after intensity mapping;
- Show image statistics (both global and local);
- Maintain a noise model for the image across transformations; //
- Maintain bad pixel information;
- Perform ccd reductions (dark/bias/flat);
- Automatically align (register) and stack images.

Catalogs and WCS

- Read field star information from GSC1/2 and Tycho2;
- Read object information from `edb` and native files;
- Read recipe files;
- Detect sources (stars) from images;
- Overlay objects on the image;
- Edit objects' information;
- Match image stars to catalog positions;
- Calculate world coordinates for image objects.

Camera Control

- Control cameras over a TCP socket using a simple protocol;

  The control proces (`cpxcntrl`) presently supports the cpx3m camera. It can be easily modified to support other cameras.
- Acquire images under script control;
- Set binning/windowing/integration times/temperature;
- Dark frames;
- All acquired frames are fully annotated in their FITS headers;
- Auto-generate descriptive names for files.

Telescope control

- Support LX200 protocol over serial;
- Point telescope under script control;
- Point telescope by object name (if edb catalogs are installed);
- Refine pointing by comparing image star positions with catalogs;

Aperture Photometry

- Do sparse field stellar photometry using fixed circular apertures for stars, annular apertures for sky estimation;
- Aperture sizes fully programmable;
- Multiple sky estimation methods;
- Uses a complex error model thorughout, that takes into account photon shot noise, read noise, noise of the callibration frames and scintillation;
- Report noise estimates for every result;
- Take photometric targets (program and standard stars) from recipe files, or directly from the image;
- Produce a comprehensive report.

Multi-Frame Reductions

- Fit color transformation coefficients from multiple frames;
- Fit extinction coefficients;
- Perform all-sky reductions;
- Generate various plots for data checking;

Interfacing

- Uses plain-ascii files for configuration files, reports and recipies;
- Implements import filters and an output converter to interface with tabular formats;
- Most functions available in batch mode, so the program can be made part of a script.

1.2 Free Software

Gcx is free software, distributed under the GNU General Public License. Users can modify it to add features, reduction algoritms, support for other cameras/telescopes, file formats. It is written in C. The GUI uses the Gtk+ 1.2 toolkit. Some GNU-specific libc functions are used, but nothing fancy. It should compile and run on any system that has GNU tools, glibc and Gtk+ 1.2. GCX is maintained on a GNU/Linux system.

1.3 Contributing

The most important contribution you can make to GCX is to try it out, and don't give up immediately if something goes wrong. Complain to the author about it—he will try to help you.

The next most important contribution is to extend the hardware support of the program. When interface library are available for cameras (many manufacturers do have such libraries), it is relatively straightforward to add support for a camera, as GCX has cleanly defined camera interface. Likewise, many mount/telescope manufacturers use the LX200 protocol, so essentially what is needed for other telescopes/mounts is testing and maybe a little tweaking. The program only uses a few LX200 functions, so interfacing to even a custom mount should be easy.

Third, there's the bane of free software: documentation. Any help in documenting or checking the documentation of the program is greatly appreciated, and will go a long way towards keeping GCX users happy.

And finally, the fun part: the code itself. There are many clever algorithms that can be added to the program, and which will benefit from the general infrastructure and integration provided by GCX .

1.4 About this Manual

This manual is work in progress. It starts with a tutorial introduction, so people can get a taste of what GCX is all about. The focus in that chapter is on operations that don't involve particular hardware (image viewing and data reduction).

The next chapters describe the main data-reduction functions of the program. In general, each chapter stars with a general description of the algorithms and methods used, then proceeds to describing how the respective methods are implemented in practice. The chapters are written roughly in the order in which data reduction procedes.

Finally, the appendices contain either technical details of the program or general aspects that invlove slighlty more complex mathematics.

The manual is maintained in LaTeX.

1.5 Related Projects

`cpxctrl` the camera server used by GCX . Currently it supports the cpx3m camera, but should be easy to modify to control different ones;

`cpx3m` a free CCD camera design;

`avsomat` a batch variable star reduction program; more portable than GCX, it shares some code but uses a different field-matching algorithm.

`xephem` The well known planetarium program by Elwood Downey. GCX can read the same object database format as xephem, namely `.edb`, and uses compatible WCS annotation FITS fields. The star search algorithm is also inspired from xephem.

`libnova` A library for celestial mechanics and astronomical calculations; GCX uses some sidereal time and equatorial-to-horizontal coordinates transformation routines from libnova.

`wcstools` A suite of utilities for setting and using the world coordinate system in FITS headers; GCX uses the same FITS header fields for specifying the WCS as wcstools. Also, the coordinate transformation (projection) routines are taken from wcstools.

Chapter 2

Getting Started

This section is a tour of gcx's features that don't require any data files other than the ones provided with the distribution, or any special hardware. It should best be read while playing with the program.

2.1 Building and Running gcx

If you're lucky (meaning that you have an i386 GNU/Linux system with compatible libc and `gtk+-1.2` is installed on your system), the precompiled binary supplied with the distribution will just work. To test, cd to the toplevel distribution directory (gcx-x.x.x) and run:

```
src/gcx
```

If all goes well, you should get an empty window with a menu. Type **ctrl-Q** or *File/Quit* to exit the program. It is recommended that the program is installed in `/usr/local/bin` for example.

If the above doesn't work,[1] you have to recompile the program. Make sure `gtk+-1.2` is installed on the system (if you have Gnome, you also have gtk), then in the toplevel directory type:

```
./configure ; make clean ; make
```

Configure takes some options. See the INSTALL file supplied with the distribution for more details.

If the above step completes successfully, become root and do a

```
make install
```

This will place the program in /usr/local/bin, and may also install data files in future versions.

The installation is now complete.

[1] Or even if it does.

2.2 Starting with gcx

The `data` subdirectory of the distribution contains an example fits frame (`uori-v-001.fits.gz`), and an example recipe file for the frame (`uori.rcp`). These will be used throughout this section.

First, start the program:[2]

```
gcx
```

You should be presented with a empty window, with a menu at the top.

To load the example frame, type **ctrl-O** or use *File/Open Fits*; select the example fits file (`uori-v-001.fits.gz`) in the `data` directory an click *Ok*. The program will load and display the frame.

Alternatively, the fits file name can be supplied on the command line. Something like:

```
gcx data/uori-v-001.fits.gz
```

will star the program and load the frame at the same time.

Two status bars are displayed at the bottom of the window. The left one shows the current display parameters: the zoom level, the *low cut* and the *high cut*. The low cut corresponds to black on the monitor, while the high cut corresponds to 100% white. The values are expressed in the same units the FITS file is.

The right-side status bar shows the various status and error messages. When loading an image, global statistics for the image are displayed. This will be referred to as the "status bar" throughout this manual.

On most errors, a beep is sounded and an error message is printed in the status bar. Sometimes though, a command may appear to do nothing. Checking the terminal from which the program was launched will sometimes give an extra hint as to what happened.

2.3 Navigating the Image

To pan around the image, either use the scrollbars, or place the cursor over the point that you want in the center of the image and press the spacebar or the center mouse button.[3]

You can pan back to the center of the image using **ctrl-L** or select *Image/Pan Center* from the menu.

To zoom in, place the cursor over the point you want to zoom in around, and press the = key (same key that has the `'+'` symbol). To zoom out, press `-`. The *Image* menu also has *Zoom In* and *Zoom Out* options.

[2] If the program wasn't installed in `/usr/local/bin` or similar, you may have to type the full path to the binary; from the distribution toplevel directory type: `src/gcx`

[3] The image will pan only up to the point where it's edge is at the edge of the window.

When loading a frame, the image cuts are automatically selected for a convenient display of astronomical frames. The background is set at a somewhat dark level, and the dynamic range is set to span 22 times the standard deviation of the intensity across the frame. You can always return to these cuts by pressing **0** or selecting *Image/Auto Cuts*.

Pressing **1** – **8** will select various predefined contrast levels. **1** is the most contrasty: the image spans 4 sigmas, while **8** spans 90 sigmas. **9** will scale the image so that the full input range is represented (the cuts are set to the min/max values of the frame). Selecting *Image/Set Contrast/...* from the menu will accomplish the same effect.

To vary the brightness of the background, use **B** (*Image/Brighter*) and **D** (*Image/Darker*).

Another, sometimes more convenient way of making contrast/brightness adjustments is to drag[4] the pointer over the image. Dragging horisontally will change the brightness, while dragging vertically will adjust the contrast.

The key presses mentioned above are displayed in the menus alongside the respective options. **F1** or *Help/Show Bindings* will show on-line help about mouse actions.

It is important to know that all the ajustments described only apply to the display. The internal representation of the frame (and of course the disc file) is never changed in any way.

2.4 Examining the FITS Header

Select *File/Fits Header* from the menu. A new window will display the optional FITS header fields from the loaded frame.[5]

2.5 Stars

GCX maintains a list of objects it can overlay on the display and run various processing steps on. They are called "stars" or sources. The stars can be extracted from the image, or loaded from catalogs or star files.

Ctrl-click on a star image. A round circle will appear around it (you cannot mark very faint or saturated stars). You don't need to click precisely on the peak - the program will search around, find a star and create an object (a *user star*) positioned at the centroid of the star image.

Click inside the circle. Information about the star will be displayed in the status bar: the start type (field star), the pixel coordinates (counting from the top-left corner), and the world coordinates if possible. Since the frame we loaded contained WCS information, but it couldn't be verified by the program, the status bar will show world coordinates, but will mark

[4] move the mouse while holding the left button pressed

[5] The fields that are always required, like `naxis`, `SIMPLE`, `BITPIX`, etc are not displayed.

them as "uncertain" and disable all operations that depend on these objects' WCS. More on validating the WCS below.

Right clicking on a star will pop up a specific menu. As our WCS isn't validated yet, only the 'delete' option is active at this point.

Now press **S** or select *Stars/Detect Sources*. The program will search the whole frame, and mark stars. There is a limit as to how many stars will be marked. The limit can be changed by selecting *File/Edit Options*, clicking on the "+" next to *Star Detection and Search Options* and increasing the number in the *Maximum Detected Stars* field.

There is also a limit on how faint the detected stars can be. Decreasing the value in the *Star Detection SNR* field will make the program look for fainter stars. Note that a very low value of SNR will increase the run time of the detection routine considerably. Don't go below 2 or so.

To remove the detected stars from the display, use *Stars/Remove Detected Stars* or press **shift-S**.

Automatically detected stars and manually marked (user) stars are displayed with different symbols and deleted with separate commands, but otherwise equivalent. The program considers automatically detected stars somewhat expendable, but tries not to remove user stars unless specifically requested.

A second class of stars handled by GCX are catalog stars. They can be loaded from catalogs if installed on the system, or from star files.

Installing catalogs will be described later in this manual. For the moment, we will load the example recipe file from the `data` directory of the distribution.

Select *File/Load Recipe* from the menu, then select the example recipe file in the `data` directory (`uori.rcp`) and click ok.

Three types of stars will show up. Diamond-shaped ones are field stars. They are used to fit and validate the WCS. Target-shaped symbols are the standard stars. Their magnitudes are used to photometrically calibrate the frame. Cross symbols are "variable" or "target" stars - stars that we want to measure, but we don't know their magnitude in advance.[6]

To find out more about a star, right-click on a star symbol, and select *Edit Star* from the pop-up menu. This will open a dialog and display information about the star, which can be edited. The name, coordinates and comments fields should be obvious. Two types of magnitudes are shown: standard magnitudes are obtained from the catalog or recipe file; instrumental magnitudes are measured by the program.

A magnitude entry looks like this:

```
<band_name>(<system>)=<magnitude>/<error>
```

[6]The symbols used to depict various star types can be set by the user, so their appearance can vary. These are the default shapes.

The error and system fields are optional. The *band name* is the name of the filter ('v', 'b', etc). The *system* describes the source of the data. For instance, v(aavso) means 'v' magnitudes taken from aavso charts, while b(landolt) would be used for 'b' magnitudes of landolt standards.

For more information about stars, please see Chapter 4

2.6 World Coordinate System

Each time a frame is loaded, the program keeps track of the relation between the the positions within the frame, and the "true" positions of the objects. This relation is called the "WCS" inside the program.

If no information is known about the position of the field, the WCS is called "invalid". This can happen if the frame doesn't have WCS information in the header. When some information is available, we say the we have an "initial WCS". The program will treat wcs information from the header as approximate. If we have an initial WCS and some field stars, we can match the positions of the field stars with stars detected from the frame. If the program finds a good-enough match, it will decide that the WCS can be reliably used, and mark the WCS as 'valid'.

Our example frame already has an initial WCS. We have field stars loaded from the recipe file (or we could have some from GSC). We will first press **S** to detect starts from the frame. Select *Wcs/Auto Pairs* (or press **P**). This will match the stars and create pairs, which are drawn with dotted lines. Next, press **Shift-W** (or *Wcs/Fit Wcs from Pairs*), and the program will fit the WCS so that the pairs overlap, and display the mean error of the fit in the status bar. If enough pairs are fitted and the error is small enough, the fit will be validated.

Pressing **M** or *Wcs/Auto Wcs* will do all the above steps in one operation (detect stars, load field stars from GSC if possible, find pairs and fit the WCS). Pressing **shift-M** or *Wcs/Quiet Auto Wcs* will do the same, but will remove the detected stars and field stars after the fit. It will do nothing if the WCS is already valid.

The fitting algorithm can be tuned by changing parameters under *WCS fitting options* in the options dialog.

Once we have a valid WCS, we have new uses for the detected and user stars. Clicking on them will print their true coordinates on the status bar. It is also possible to mark them as variable stars, so they can be measured, or as standard stars, so they can participate in the photometry solution (for example when inputting data from a paper chart).

Choose a few detected stars, right click on them and choose *Edit Star*. Now check the "variable" flag. The star will be transformed into a variable, and its symbol changed to a cross.

2.7 Aperture Photometry

Now that we have our valid WCS and we know which stars we want to measure and which standards to use, the actual photometry is easy: just press **shift-P** or *Processing/Quick Aperture Photometry*.

A quick result for the first variable star is printed in the status bar. All stars' magnitudes are updated, and can be examined using the *Edit Star* function.

The reduction process has a number of parameters, which can be accessed through the options dialog, under *Aperture Photometry Defaults*. For more details about the photometry process check Chapter 7.

All the clicking in this section can be eliminated with one command. From the toplevel directory, run:

```
gcx data/uori-v-001.fits.gz -P data/uori.rcp
```

The program will load the frame, load the recipe, fit the WCS and run the photometry. A report will be written to standard out (all debugging messages are printed to stderr, so redirecting stdout to a file will write just the report to that file. For example:

```
gcx data/uori-v-001.fits.gz -P data/uori.rcp >outf
```

will write the report to outf.

2.8 Going Further

GCX has many more features and options than the ones described above. To find out about them, read below, browse the menus, or ask the author.

Chapter 3

Image Files

The basic format used by GCX for image files is FITS. Internally, images are represented using 32-bit (single-precision) floating-point values. The images are read and saved as 16-bit integer files. It it relativey easy to modify the program to read outher FITS formats (like 8-bit or floating point), should the need arise.

GCX will read and write fits files compressed in the gnuzip format (ending in `.fits.gz`) transparently. A zipped file name can be used whenever a regular fits file name is required.[1]

While the way image data is represented in the FITS files is well specified, additional information from the fits header can vary between different programs. Since the data reduction functions of GCX make use of quite a few fits header values, it is important to understand how they are interpreted.

Some camera control programs generate broken 16-bit FITS files that have `BZERO` set to 0 and all values stored as unsigned numbers. When loaded, values larger than 32768 appear as negative numbers. To accomodate these files, GCX provides the *File and Device Options/Force unsigned FITS* option, which will make the program interpret all values in frames with `BZERO`=0 as unsigned. Note that files are always saved in the standard format.

3.1 FITS Header fields

The default names of the fits header fields have been set to what the author considers the most common. It is however possible to change any of the field names by editing the options or the `~/.gcxrc` file.

After loading a fits file, you can examine it's header by selecting *File/Show Fits Header*.[2] When new frames are created by GCX from images captured by a CCD camera, the frame is annotated from one of the following sources:

[1] This function uses the `zcat` utility, which must be installed on the system.

[2] Note that the required FITS fields like `SIMPLE, BITPIX, NAXIS, NAXIS2, BSCALE, BZERO` are not shown by this operation.

- The *General Observation Setup Data* options;
- Object information from the currently selected target object;
- Frame information from the camera.

The table below details the most important fits header fields set and read by GCX :

Name	Type	Meaning	Comments
CRPIX1/2	Real	Coordinates of reference pixel	Set to the center of the frame; Used to set the initial WCS.
CDELT1/2	Real	Scale of image in degrees per pixel	Set initially from the observation setup focal length, pixel size and binning; updated after WCS fitting.
CROTA1	Real	Rotation of image in degrees	Set initially to 0; updated after WCS fitting.
CRVAL1/2	Real	World coordinates of the reference pixel	Set from the target object coordinates; update after WCS fitting.
FILTER	String	Name of filter used	Set from the current observation data or filter wheel status; used for photometric reductions.
ELADU	Real	Image electrons per AD unit	Set from camera parameters; used by the photometry routines in the error model.
RDNOISE	Real	camera read noise in AD units	Set from camera parameters; used by the photometry routines in the error model.
EXPTIME	Real	integration time in seconds	Set from camera paramters; used by the scintillation noise evaluation routine
JDATE	Real	Julian date of the start of integration	Used, among other things to calculate the frame airmass
MJD	Real	Modified Julian Date of the start of integration	An alternative to JDATE
DATE-OBS	String	The date/time of start of exposure in the format specified by the fits standard	An alternative to JDATE

Name	Type	Meaning	Comments
TIME-OBS	String	The time of day portion of the date/time, used when the DATE-OBS field only sets the date	An alternative to JDATE
APERT	Real	Telescope aperture in cm	Set from general observation options; Used to evaluate scintillation noise
LAT-OBS	String	Latitude of observation site in DMS format	Set from general observation options; Used to calculate airmass
LONG-OBS	String	Latitude of observation site in DMS format; Eastern latitudes are negative	Set from general observation options; Used to calculate airmass
AIRMASS	Real	Frame airmass	If present, disables airmass calculation and sets the airmass value
OBJECT	String	Target object name	Used to set the initial WCS if other information is not available
OBJCTRA	String	Target right ascension in HMS format	Used to set the initial WCS if other information is not available
RA	String	Target right ascension in HMS format	Used to set the initial WCS if other information is not available
OBJCTDEC	String	Target declination in DMS format	Used to set the initial WCS if other information is not available
DEC	String	Target declination in DMS format	Used to set the initial WCS if other information is not available
SECPIX	Real	Image scale in arcseconds per pixel	Used to set the initial WCS if other information is not available

3.2 Viewing Image Files

To pan around the image, either use the scrollbars, or place the cursor over the point that you want in the center of the image and press the spacebar or the center mouse button.[3] You can pan back to the center of the image using **ctrl-L** or select *Image/Pan Center* from the menu.

To zoom in, place the cursor over the point you want to zoom in around, and press the = key (same key that has the '+' symbol). To zoom out, press **-**. The *Image* menu also has *Zoom In* and *Zoom Out* options.

When loading a frame, the image cuts are automatically selected for a convenient display of astronomical frames. The background is set at a somewhat dark level, and the dynamic range is set to span 22 times the standard deviation of the intensity across the frame. You can always return to these cuts by pressing **0** or selecting *Image/Auto Cuts*.

Pressing **1** – **8** will select various predefined contrast levels. **1** is the most contrasty: the image spans 4 sigmas, while **8** spans 90 sigmas. **9** will scale the image so that the full input range is represented (the cuts are set to the min/max values of the frame). Selecting *Image/Set Contrast/...* from the menu will accomplish the same effect.

To vary the brightness of the background, use **B** (*Image/Brighter*) and **D** (*Image/Darker*).

Another (sometimes more convenient) way of making contrast/brightness adjustments is to drag the pointer over the image. Dragging horisontally will change the brightness, while dragging vertically will adjust the contrast.

It is important to know that all the ajustments described only apply to the display. The internal representation of the frame (and of course the disc file) is never changed in any way.

Further adjustments to the way the image is displayed can be made by bringing up the *Image/Curves&Histogram* dialog. This dialog shows a portion of the frame's histogram, overlapped by the currently set intensity transfer function. The image cuts are placed at the red vertical bars in the histogram window.

The shape of the transfer function can be altered by changing the Gamma and Toe parameters. Gamma controls the overall shape, while the Toe controls what happens in the leftmost portion of the curve. Increasing the toe will prevent the transfer function from having a very high slope near zero, as would be implied by the gamma setting.

3.3 Saving and Exporting Images

The image currently displayed can be saved in the FITS format by selecting *File/Save FITS As*. When frames are saved as FITS, their values aren't

[3] The image will pan only up to the point where it's edge is at the edge of the window.

affected by the display settings. Another option is to export the file to a different format (presently, only the 8-bit pnm format is supported). When exporting, the intensity mapping used for displaying the image is used to convert from the original frame to the 8-bit output image.

Chapter 4

Stars and Catalogs

In addition to images, GCX handles lists of astronomical objects generally referred to as *stars*. There are several types of stars, and the various types have different kinds of information attached. They can be grouped in two classes:

1. *Frame stars* only have positions within a frame (pixel coordinates). They come into existence by detection, either automatic, or user-directed. There are three sub-types of frame stars:
   - detected stars;
   - user stars;
   - aligment stars.
2. *Catalog stars* have world coordinates (right ascension and declination) attached to them, and can also hold photometric information. Catalog stars are either read from a catalog or recipe file, or created interactively by editing another star. There are four sub-types of catalog stars:
   - fields stars;
   - catalog objects;
   - standard stars;
   - photometry targets (AP targets).

Stars are drawn on top of the displayed images. The apearance of the various types of stars can be changed using the options under *Star Display Options*.

4.1 Star Detection

A relatively straight-forward star detection algorithm is implemented in GCX. It will search for local intensity peaks that satisfy the following conditions:

1. The peak is higher than the local background plus a specified number of standard deviations (usually 6-9). The number of standard deviations is specified in the *Star Detection SNR* option;

2. There are at least 4 pixels adjacent to the peak which are above the threshold;

3. The peak is not too close to another, higher peak;

4. The star radius (are in which the star is above the background) isn't too large;

The *Maximum Detected Stars* parameter limits the number of stars that are detected. The whole frame is searched and the bightest stars are kept.

The star detection routine is called by selecting *Stars/Detect Sources*, or automatically by other operations (WCS fitting and frame alignment). The detection routine will produce *detected stars*, except when called on an alignment reference frame, in which case it will produce *alignment stars*.

Another way of creating frame stars is to control-click on or near an otherwise unmarked star image. This will initiate a spiral star search in a region around the the cursor, with the SNR parameter set to a low value (3.0). In this way, even very faint stars that are located near the cursor can be marked. This procedure creates *user stars*.

4.2 Loading Catalog Stars

As their name implies, catalog stars are generally loaded from catalog files. GCX supports three kinds of catalogs: object catalogs, from which the stars are loaded by name (like GCVS, NGC, Messier, IC); field star catalogs, from which the stars are loaded by region of interest; and star files (including recipe files), which are loaded in their entirety, as they presumably contain stars in a region of interest.

As catalog stars are located by world coordinates, the program cannot display any of them if it doesn't have at least an approximate idea of what the image's coordinates are in the real world. The frame must have at least an *initial WCS*.[1]

If the frame WCS is unset, the program will refuse to load field stars; it will set the initial WCS so that the loaded object is at the center of the frame if a single object is loaded from an object catalog; and it will refuse to load a star file, unless the star file is a recipe that contains target coordinates, in which case the initial WCS will be set from those coordinates.

[1] See the "World Coordinates" section below.

4.2.1 Object Catalogs

Two formats for object catalogs are currently supported: the `.edb` format, also used by XEphem, and the `.gcx` format. The two are treated differently: `.gcx` files can be loaded in memory and the whole set searched by name; `.edb` files are searched directly (without loading). The particular edb file to search is selected depending on the object name.

There are two ways of loading stars from object catalogs: Selecting *Stars/Add From Catalog* will prompt for an object name and load it. If the frame has the `OBJECT` FITS field set, selecting *Stars/Show Target* will try to load the object specified in that field. Stars loaded from object catalogs are of the *catalog object* type.

4.2.2 Field Star Catalogs

Currently, the program supports two field star catalogs directly: GSC (and GSC-ACT) and TYCHO2. It can also load GSC2 objects in the form of star files (see below).

To load stars from GSC or Tycho, the frame has to have at least an initial WCS; then select *File/Load Field Stars/From GSC Catalog* or *File/Load Field Stars/From Tycho2 Catalog*.

Options under *Star Detection and Search Options* control the maximum number of stars loaded and the limiting magnitude. When more stars than the specified limit are found in the catalog, only the brightest ones are loaded. The region searched is slightly larger than the frame's size in world coordinates.

4.3 Star Files

GSC-2 files A file containing GSC2 stars in a specified region can be obtained from

`http://www-gsss.stsci.edu/support/data_access.html`

It can be loaded into GCX by selecting *File/Load Field Stars/From GSC-2 file*. Stars loaded this way are marked as *field stars*.

Native format files GCX defines a native format for star files. It is used for object catalogs, photometry recipe and report files, among other things. Stars from any of these files can be read by selecting *File/Load Recipe*. The native format contains star type information, which is maintained on loads.

Other star files Many star files are available from various sources, usually in some form of tabular format. GCX has a command-line only import function to convert these to recipe files (see `gcx --help` for more details).

While it can read a limited range of formats, it is relativey easy to either convert a given file to a supported format, or modify an existing conversion routine.[2]

4.4 Setting up Catalogs

The program expects the various catalogs to be set up in a certain way. Here are the requirements for the various catalogs.

4.4.1 Setting up GSC

The program reads the GSC or GSC-ACT[3] catalogs in the compact (binary) form. It can be downloaded in this format from:

```
ftp://cdsarc.u-strasbg.fr/pub/cats/I/255/GSC_ACT/
```

Place the downloaded files in a directory (for instance `/usr/share/gcx/gsc-act`) and set *File and Device Options/GSC Location* to point to that directory. Make sure all the directory and file names under the gsc dir use all lower case letters.

4.4.2 Setting up Tycho2

The Tycho2 catalog can be downloaded from

```
ftp://cdsarc.u-strasbg.fr/pub/cats/I/259/
```

We need to download all the `tyc2.dat.nn.gz` files, unzip them and concatenate together to create one (big) tycho2.dat file, perhaps like this:

```
zcat tyc2.dat.??.gz >tycho2.dat
```

After that, set *File and Device Options/Tycho2 location* to the full path of the file, like for instance:

```
/usr/share/gcx/tycho2/tycho2.dat
```

GCX must have write permissions to the directory where the `tycho2.dat` file is located, as it needs to create some indexes when the catalog is first accessed.

[2]The conversion routines are located in `recipe.c`; The function called by the import function is `convert_catalog`.

[3]GSC-ACT is a recalibrated version of the GSC. While the random astrometric errors are about the same, the systematic errors have been reduced considerably.

4.4.3 Setting up Object Catalog Files

To tell the program what native format catalog files to load, set the *File and Device Options/Catalog files* option to a comma-delimited list of files. The list can contain wildcards and is tilde-expanded. An example entry would be:

```
/usr/share/gcx/catalogs/*.gcx:~/catalogs/gcvs.gcx
```

All `.edb` files that are searched must be located in the same directory, specified in *File and Device Options/EDB files*. Depending on the searched object's name, GCX will look in a specific file:[4]

Objects with names starting in "M"	`Messier.edb`
Objects with names starting in "NGC"	`NGC.edb`
Objects with names starting in "UGC" or "UGCA"	`UGC.edb`
Objects with names starting in "IC"	`IC.edb`
Objects with names starting in "SAO"	`sao.edb`
Objects with ending in a constellation name	`gcvs.edb`
Other objects	`YBS.edb`

[4]This make-shift arrangement is likely to change in future versions.

Chapter 5

World Coordinates

World coordinates are the "real" equatorial coordinates of objects in catalogs: right ascension, declination and their epoch.[1] Given an image frame, we reffer to the transformation between x and y pixel coordinates and their world coordinate counterparts as the *World Coordinte System* (WCS for short) of the frame.

The transformation between the spherical equatorial and the "flat" image coordinates cannot be done without choosing a projection system. GCX uses the plane-tangent projection system, which is appropiate for relatively narrow fields.[2]

5.1 World Coordinate System Parameters

In the plane-tangent system, the WCS is specified by the following values:

1. The frame coordinates of a reference pixel in the image (usually the center of the frame) in the `CRPIX1` and `CRPIX2` fits header fields;
2. The world coordinates (r.a. and dec) of the reference pixel in the `CRVAL1` and `CRVAL2` fields;
3. The epoch of the coordinates in the `EQUINOX` header field;
4. The horisontal and vertical scale of the image in degrees per pixel in the `CDELT1` and `CDELT2` fields;
5. The rotation of the frame in the `CROTA1` field.

A slightly different form of these parameters is presented in the WCS editing dialog: the scale parameters are expressed in the more friendly arc seconds per pixel units, and the coordinates are expressed in the HMS and DMS formats.

[1] Whenever the epoch of some coordintes is not specified, GCX assumes J2000.

[2] This is the same system used by the popular *wcstools* package.

5.2 World Coordinate System States

A given frame's WCS can be in one of the following states:

Unset When the WCS is unset, the program has no idea about the WCS. It will refuse to do any operation that requires the WCS.

Initial An initial WCS is an approximate set of values for the WCS parameters. It enables the program to load catalog stars and display them on the image (more or less around their true positions). It also provides a starting point for WCS fitting. GCX will not use an initial WCS for any operation that requires precise coordinates (like aperture photometry).

Fitted The WCS has been successfully fitted, but the quality of the fit was not enough to allow it to be validated. A *fitted* WCS is treated very much like an initial WCS.

Valid If a fit was good enough (enough stars were fitted, and the error was low enough), the WCS is deemed *valid*. All operations that use the WCS are enabled in this situation.

5.3 Obtaining an Initial WCS

When a frame is loaded, the WCS is initially unset. The header of the frame is searched for information about the initial WCS. The following fields are searched, in order:[3]

1. `CRVAL1/2, CDELT1/2, CROTA1, CRPIX1/2, EQUINOX`. The bare minimum set consists of`CRVAL1, CRVAL2` and one of the `CDELTs`.

2. `RA` or `OBJCTRA`, `DEC` or `OBJCTDEC`, `PIXSCALE` or `SECPIX`. If neither of the scale fields is found, a default scale values is taken from *Wcs Fitting Options/Default image scale*;

3. `OBJECT` If this field is present, the object's name is searched in the catalog, and its coordinates used. The image scale is set from *Wcs Fitting Options/Default image scale*;

When neither of the above fields are found, the WCS is left in the *unset* state. An initial WCS can be set in this case by either entering the parameters in the WCS edit dialog (*Wcs/Edit Wcs*), loading a catalog object using *Stars/Add From Catalog* or loading a recipe file that has the target object or field center specified. In the last two cases, the default scale is used.

[3]The actual names of any of the fields can be changed in the options page. The names below are the defaults.

5.4 Fitting the WCS to an Image

By WCS fitting we understand the process of comparing the positions of stars extracted from the image frame versus the projected positions of catalog stars, and the subsequent adjustment of the WCS for the best match.

The fitting process consists of the following steps:

1. Detecting frame stars. This step is described in section 4.1;

2. Obtaining catalog stars for the match. These can come from either a recipe file or one of the field stars catalogs. The program will load stars from the Tycho2 and GSC catalogs. All the stars from a loaded recipe file that have the "astrimetric" flag set will also be used for WCS fitting;

3. Finding star pairs. This step tries to find similar asterism in the detected and catalog sets and match the corresponding stars.

   The algorithm tolerates frame rotation and changes in scale. If some bounds can be placed on initial errors (for instance if we know that only a limited rotation range is expected) it is possible to pass that information to the algorithm in order to narrow the search.

4. Fitting the solution. This is an iterative step consisting of calculating the required offset, scale and rotation in the frame coordinates, then adjusting the WCS accordingly. After that, the image coordinates of the catalog stars are recalculated and the step repeated until there is no significant change in the WCS. The iterative approach is necessary because the projection operation is non-linear. At the end of the fitting step, a *rms* position error is calculated, and compared to the value of the *Max error for WCS validation*. If the error is lower and enough pairs have been used in the fit (more than *Min pars for WCS validation*), the WCS is marked "valid".

The *Scale tolerance* option sets the maximum initial error of the image scale for the pairing alogorithm. A value of 0.1 specifies that the scale of the initial WCS has an error of at most ±10%. The *Rotation tolerance* specifies how much field rotation is expected by the pairs matching algorithm. A value of 180 will let the algorithm match frames of any rotation. A third important parameter is *Minimum number of pairs*. This specifies the number of pairs at which the algorithm decides it has found a match. The default values for these parameters almost never generate a bad match, even for quite dense fields. If one increases the scale tolerance, there is an increased risk of having a bad match, and the minimum pairs should be increased as well.

The pairing algorithm requires the initial WCS to have the correct mirroring. When the initial WCS's scale comes from the `CDELT1/2` fields, their

signs will determine the mirroring: when both have the same sign, the frame is "normal", i.e. W is to the right when N is up. If the signs are different, the field is flipped.

When the initial WCS's scale comes from a single scale parameter, the mirroring will be set by the program according to the value of the *General Observation Setup Data/Flipped field* option.

5.4.1 WCS Fitting Commands

The WCS fitting steps can be performed one at a time, or all together. The *Wcs/Auto Wcs* operation will do the following steps: *Stars/Detect sources*, *File/Load Field Stars/From Tycho2 Catalog*, *Wcs/Auto pairs*, *Wcs/Fit Wcs from pairs*. The *Wcs/Quiet Auto Wcs* variant will also delete the detected and field stars at the end of the fit.

Selecting *Wcs/Reload from frame* will revert the WCS to the parameters before the fit. The pairs will remain marked.

In the unlikely event that the pairing algorithm fails,[4] it is possible to create pairs "by hand". Select a detected star, then right-click on the catalog star you want to pair it with and select *Create Pair* from the pop-up menu. When at least 2 pairs have been marked, we can fit the wcs with *Wcs/Fit Wcs from Pairs*. Note that the fit will not be marked as "valid" unless at least *Minimum number of pairs* have been marked.

[4]The author would very much like to receive any frames for which the algorithm fails.

Chapter 6

CCD Reduction

Ideally, an image taked by a CCD camera through a telescope will give accurate information about the light flux distribution over a portion of the sky. Unfortunately, this is not generally the case. Instrument imperfections and the discrete nature of light itself concur to introduce errors in the measured data. The errors (differences between the measured values and the "true" ones) are the result of several factors, some random in nature, and some deteministic.

The goal of the CCD reduction process is to eliminate (or at least minimise) the contribution of deterministic factors in the errors, in other words to remove the *instrument signature* from the data.

A second, but not less important, goal is to preserve information about the noise sources, so that users of the reduced data can evaluate the random errors of the data.

We begin this chapter by describing the way the general CCD reduction process works, and in the process define bias, dark and flat files. The second part of the chapter is devoted to the practical implementation of the reduction tasks in the program.

6.1 CCD Camera Response Model

Raw pixel values for a CCD frame can be calculated as follows:[1]

$$s(x,y) = B(x,y) + tD(x,y) + tG(x,y)I(x,y) + \text{noise} \tag{6.1}$$

where $B(x,y)$ is the *bias* value of each pixel, t is the integration time, $D(x,y)$ is the *dark current*, $G(x,y)$ is the *sensitivity* and $I(x,y)$ is the light flux reaching the pixel. We cannot predict the instantaneous values of the noise component but some statistics about it can be calculated (Appendix A).

To estimate the flux values reaching the sensor from the raw frame, we need to estimate B, D and G. After that, Equation 6.1 can be solved for

[1] Neglecting non-linear terms in the CCD response.

I. B, D and G are calculated starting from calibration frames taken under controlled conditions: *bias*, *dark* and *flat* frames.

6.2 Bias and Dark Frames

If we take very short exposures without opening the camera's shutter (*bias frames* $t = 0$ and $I(x, y) = 0$; Equation (6.1) becomes:

$$b(x, y) = B(x, y) + \text{noise} \tag{6.2}$$

To obtain an estimate of B, we simply use b:

$$\widetilde{B}(x, y) = b(x, y) \tag{6.3}$$

We use the tilde to denote that we can only estimate B, because of the noise. If we average several bias frames together we can get arbitrarily close to B, as the relative noise contribution decreases with the square root of the number of frames averaged.

$$\widetilde{B}(x, y) = \frac{1}{N} \sum_i b_i(x, y) \tag{6.4}$$

We will call this the *master bias frame*.

Dark frames If we now take longer exposures with the shutter closed, we obtain *dark frames*:

$$d(x, y) = B(x, y) + tD(x, y) + \text{noise} \tag{6.5}$$

From this, we can simply subtract the bias and divide by the exposure time, and we get our dark current estimate:

$$\widetilde{D}(x, y) = \frac{d(x, y) - \widetilde{B}(x, y)}{t_{\text{dark}}} \tag{6.6}$$

Of course, to reduce the noise contribution we can also average several dark frames:

$$\widetilde{D}(x, y) = \frac{1}{t_{\text{dark}}} \frac{1}{M} \sum_i d_i(x, y) - \widetilde{B}(x, y) \tag{6.7}$$

It is convenient to work with a different form of the dark current frame:

$$\widetilde{D}'(x, y) = t_{\text{dark}} \widetilde{D}(x, y) = \frac{1}{M} \sum_i d_i(x, y) - \widetilde{B}(x, y) \tag{6.8}$$

which will be called the *bias subtracted master dark frame*.

If we have a data frame with an integration time of t_{data}, the first two terms in (6.1) are estimated by the *master dark* frame $\widetilde{D}_M$:

$$\widetilde{D}_M(x, y) = \widetilde{B}(x, y) + t_{\text{data}} \widetilde{D}(x, y) = \widetilde{B}(x, y) + \frac{t_{\text{data}}}{t_{\text{dark}}} \widetilde{D}'(x, y) \tag{6.9}$$

Noise contribution of bias and dark frames. A detailed description of noise sources in CCD cameras is provided in Appendix A. If the dark current contribution is not very large, the *noise* terms of (6.2) and (6.5) are both equal to the camera *read noise*.

If we use our estimated $\widetilde{B}$ and $\widetilde{D}$ to reduce a data frame with an integration time of t_{data}, the bias and dark subtraction will contribute a noise level of:[2]

$$\sigma_{DB} = N_R \sqrt{\frac{1}{N} + \frac{1}{M}\left(\frac{t_{\text{data}}}{t_{\text{dark}}}\right)^2} \tag{6.10}$$

where N_R is the camera read noise, N is the number of bias frames averaged, M the number of dark frames averaged and t_{dark} the integration time used for the dark frames.

We generally want to keep the square root in (6.10) between 1/3 and 1. A value lower than 1/3 will provide a negligible improvement in the overall signal/noise ratio, while for values larger than 1, this term will dominate the camera read noise and become significant.

6.2.1 Working without Bias Frames

It is easy to observe from (6.9) that we can obtain our master dark frame by simply averaging dark frames taken with the same integration time as our data frames. In this case, we don't need the bias frames at all:

$$\widetilde{D}_M(x,y) = \frac{1}{M}\sum_i d_i(x,y) \tag{6.11}$$

As a bonus, the noise contribution of the master dark frame is reduced to:

$$\sigma_D = N_R \sqrt{\frac{1}{M}} \tag{6.12}$$

In general, people using large telescopes and LN_2 cooled cameras preffer using bias frames, as they require less time than dark frames; The dark current of these cameras is very low and stable, and a single set of darks can be used to reduce many observations. Users of thermoelectrically-cooled cameras, which have more significant dark currents, are more likely to use dark frames exclussively.

6.3 Flat-field Frames

With B and D out of the way, we need a way to estimate G in (6.1) before we can recover the incident flux. To do this, we apply a flat-field (even)

[2] Neglecting the noise the bias frames add to the dark current estimate.

illumination to the camera[3] and acquire several *flat-field frames*

$$f(x,y) = B(x,y) + t_{\text{flat}} D(x,y) + t_{\text{flat}} G(x,y) L + \text{noise} \tag{6.13}$$

L is the light flux reaching each pixel, assumed equal across the frame.

We then calculate a master dark frame for the flat fields $D_M^F(x,y)$ and subtract it from the flats, obtaining:

$$f'(x,y) = f(x,y) - D_M^F(x,y) = t_{\text{flat}} G(x,y) L + \text{noise} \tag{6.14}$$

Again, to reduce the noise contribution, we usually average several flat frames, to arrive at a *master flat* frame

$$\widetilde{F}_M(x,y) = \frac{1}{N} \sum_i f'(x,y) = \frac{1}{N} \sum_i f(x,y) - D_M^F(x,y) \tag{6.15}$$

If we knew F, we could solve the above equation for $G(x,y)$. However, the aboslute value of F is not known, and in many cases can vary between different flats. So, instead of calibrating the absolute value of $G(x,y)$, we only try to remove it's variation across the frame. We write:

$$G(x,y) = \bar{G} g(x,y) \tag{6.16}$$

where the average of $g(x,y)$ across the frame is 1. (6.15) becomes:

$$\widetilde{F}_M(x,y) = t_{\text{flat}} \bar{G} g(x,y) L \tag{6.17}$$

We take the average of $\widetilde{F}_M(x,y)$ across the frame:

$$\bar{F} = \sum_x \sum_y t_{\text{flat}} \bar{G} g(x,y) L = t_{\text{flat}} \bar{G} L \sum_x \sum_y g(x,y) = t_{\text{flat}} \bar{G} L \tag{6.18}$$

Dividing equation 6.17 by $\bar{F}$, we obtain:[4]

$$\frac{\widetilde{F}_M(x,y)}{\bar{F}} = g(x,y) \tag{6.19}$$

[3]For the purpose of this discussion, the telescope illumination non-uniformity is folded into the camera response; the sensitivity we estimate will correct both the camera and the telescope's response nonuniformity.

[4]We can in principle replace the master flat frame with this normalised frame, by which we can divide the data frames directly. However, a normalised frame doesn't represent well in the very common 16-bit integer format. So we instead choose to keep the master flat frame scaled to it's original scale, and take care of the normalisation in the flat-field division routine.

6.4 Reducing the Data Frames

Armed with our master dark and master flat frames, we can proceed to reduce our data frame. Starting from:

$$s(x,y) = B(x,y) + t_{data}D(x,y) + t_{\rm data}\bar{G}g(x,y)I(x,y)) + \text{noise} \qquad (6.20)$$

we subtract the master dark frame (6.9) and divide by the normalised master flat (6.19) and obtain:

$$\frac{\bar{F}}{\widetilde{F}_M(x,y)}[s(x,y) - \widetilde{D}_M(x,y)] = t_{\rm data}\bar{G}I(x,y) + \text{noise} \qquad (6.21)$$

$$\widetilde{I}(x,y) = \frac{1}{\bar{G}\cdot t_{\rm data}}\frac{\bar{F}}{\widetilde{F}_M(x,y)}[s(x,y) - \widetilde{D}_M(x,y)] \qquad (6.22)$$

Or, in terms of the master bias and bias-subtracted master dark frames:

$$\widetilde{I}(x,y) = \frac{1}{\bar{G}\cdot t_{\rm data}}\frac{\bar{F}}{\widetilde{F}_M(x,y)}[s(x,y) - \frac{t_{\rm data}}{t_{\rm dark}}\widetilde{D}'(x,y) - \widetilde{B}(x,y)] \qquad (6.23)$$

We have obtained an estimate of the incident light flux up to a constant ($\bar{G}t_{\rm data}$) which represents the average sensitivity of the camera multiplied by the integration time, which is the best we can do without a reference source calibrated in absolute units.

6.5 Frame Combining Methods

Earlier in this chapter we mentioned that it is useful in many cases to "average" several frames in order to reduce the noise. It turns out that while arithmetic averaging does indeed reduce the resulting noise, it doesn't go very far in removing the effect of devinat values that are far from the mean. When combining CCD frames, the most common causes of deviant values are cosmic ray hits on all frames and unwanted star images on sky flats.

When combining N CCD frames, we start with N values for each pixel and want to arrive at a combined pixel value that uses as much as possible of the available information (so that we obtain the maximum noise reduction), while rejecting any values that are affected by artifacts. GCX implements four combining methods, described below.

Average Average is the "baseline" method of frame combining. The values are added together and the result divided by the number of values (arithmetic mean). For Gaussian-distributed data, averaging produces the values with the least variance (so it has the maximal *statistical efficiency*). Averaging is independent of the individual frame scaling, and computationally efficient.

The deviant value rejection of averaging is only modest; deviant values are reduced by a factor of N. For this reason, it is recommended that averaging only be used when we know that deviant values are not present (for example, when combining frames for which the outlier rejection has already been done).

Median A far more robust method of obtaining a combined value is the median (selecting the values which has an equal number of values greater and smaller than itself). The median is easily calculated and very little influenced by deviant values. It requires all frames to have the same intensity scale.

On the down side, the median's statistical efficiency for Gaussian distributed values is only 0.65 of average's. Also, when combining integer values, the median does nothing to smooth out the quantisation effects; the median of any number of integer values is also an integer.

Mean-Median Mean-median is a variant of the median method intended to improve the statistical efficiency of the median, and get around the quantisation problem. In the mean-median method, we compute the standard deviation of the pixel values around the median, and discard all the values than are farther away than a specified number of standard deviations (usually 1.5 or 2). The remaining values are averaged together.

Mean-median is fast, and works well for large sets. With small sets, the fact that the deviant pixels increase the calculated standard deviation limits it's efficiency. It requires all frames to have the same intensity scale.

Kappa-Sigma Clipping $\kappa\sigma$-clipping is the most ellaborate combining method provided by GCX . It starts by calculating the median and the standard deviation around it. The values with large deviations relative to the standard deviation are excluded. Then, the mean and standard deviation of the remaining values are computed. Again, the values that are away for the mean are excluded, and the process is repeated until there is no change in the mean or the iteration limit is reached.

$\kappa\sigma$-clipping is very effective at removing cosmic rays and star images from sky flats, even with a reduced number of frames. It is also capable of removing all the star images from a number of frames of different fields, leaving only the sky background, which makes it possible to create a flat frame from the regular data frames (see below for *super-flat*).

$\kappa\sigma$-clipping is the most computationally-intensive method of frame combining. It requires all frames to have the same intensity scale.

6.6 CCD Reduction with gcx

Like most tasks in GCX CCD Reductions can be performed either interactively or using the command-line interface. We'll discuss the interactive way first.

6.6.1 Loading and Selecting Image Frames

To open up the CCD reduction dialog, select *Processing/CCD Reduction* or press **L**. In the *Image Files* tab, click on *Add* and in the file selector select any number of files and click *Ok*. The selected files will appear in the file list.

To display any of the frames in the list, click on it (it will become selected) and then click *Display*. The image will show in the main window. The loaded files can be viewed in sequence by clicking *next* or pressing **N** repeteadly. If for some reson we want to exclude a frame from being processed, clicking on *Skip* or pressing **S** will mark it to be skipped.

Skipped frames appear in the list with their names enclosed in square brackets. If we want to remove the skip mark, clicking *Unskip* while the frames are selected will accomplish that. Note that all reduction operations apply to all the frames in the file list that don't have the skip mark, regardless of which are selected.

Whenever a frame is selected, a status line at the bottom of the dialog shows the file's name and any operations already performed on it.[5]

To revert some frames to their original status, select them and click on *Reload*. Frames can be completely removed from the list by selecting them and the clicking on *Remove*.

6.6.2 Creating a Master Bias or Master Dark Frame

We will create a master bias frame by stacking several bias frames. If we preffer to work without bias frames, we can use the same procedure to create a master dark frame. First clear the file list (*Select all* then *Remove*) and add the bias frames to be stacked to the list.

Then, in the *CCD Reduction* tab, check that all the "enable" ticks are off (we don't want to apply any operation to the bias frames). Same for the *Aligment* tab.

In the *Stacking* tab, select the desired parameters for the stacking operation (the method and method parameters). A good initial setting is kappa_sigma, 1.5 sigmas and an iteration limit of 4. Set background matching to Off, and check "Enable stacking".

[5] All processing is done on a copy of the frame held in memory—the original frames are not altered in any way.

Finally, in the *Run* tab type in a file name[6] [7] (for example "master_bias") and click *Run*. The progress of the stacking operation can be followed in the text window. After stacking is complete, the resulting frame is shown in the main window.

6.6.3 Creating a Bias-Subtracted Master Dark Frame

Suppose now that we have a number of dark frames taken with the same integration time and we want to create a bias subtracted master dark frame from them. We'll use the master bias frame we just created.

Like above, clear the frame list and load the dark frames. Then, in the *CCD Reduction* tab enter the name of the master bias frame we just created (or click on the "..." button and select it in the file selector). The "enable" mark for bias subtraction should be on.

Then make sure that stacking is still enabled, enter an output file name and click on *Run*. The program will subtract the master bias frame from each of the dark frames, and then combine the results. As before, the result will be shown in the main window.

Sometimes we need to scale the bias-subtracted dark frame, so it can be used to reduce data frames taken with a different integration time. We can also do that now. Suppose our dark frames used 60 seconds on integration time, and we want to reduce 30-seconds data frames. We will need out bias-subtracted master dark to be scaled by a factor of 0.5. In the *CCD Reduction* tab check the "multiply enable" box and enter 0.5 in the multiply entry. Then click on *Run* again,[8] of course not before changing the output file name into something meaningful, like "bmdark-30" or something similar.

6.6.4 Creating a Master Flat Frame

To create a master frame, we need either a master dark frame of suitable integration time for the flats, or a master bias and suitably scaled bias-subtracted master dark frame.

Clear the file list and add the flat frames to it. In the *CCD Reduction* tab, set the bias and dark frame file names,[9] and make sure their "enable"

[6] If the output file name field is left blank, the result of the stacking operation will not be saved. It will however display in the main window, and can be saved from there.

[7] The *Output file / dir name* field must contain a file name if the result of the current reduction set is a single frame (stacking is enabled) or a directory name if the result of the current reduction set consists of several frames (stacking is disabled).

[8] Note that once a reduction operation has been performed on a image frame in the list, it will not be performed again unless the image frame is reloaded—which discards any changes to it. If we want to apply different sets of reduction operations to some frames, like creating frames with different scales, the frames have to be reloaded (selected in the *Image Files* list and *Reload* clicked) before changes in parameters (like the multiplication factor or bias file name) can take effect.

[9] Or just the dark file name if we work without biases.

boxes are checked. Go to the *Stacking* tab and enable stacking, set your algorithm and parameters. Except if you are very sure that the flat frames have the same intensity (such as when using a known stable light box), enable multiplicative background matching. Finally, select an output file name, and run the flat generation.

Making a Superflat If we have enough data frames with a significant sky background and different fields, they can be combined and the sky background used as a flat. We proceed the same as for normal flats, except that we select data frames with a good background. It is recommended that frames with low sky level are excluded from the superflat set.

We probably want to multiply the frames by a constant to make sure the relatively low level flat resulting is not affected by quatisation when the frame is saved in an integer format.

6.6.5 Reducing the Data Frames

Reducing the data frames should is easy now; load them into the file list, specify the bias, dark and flat to be used, select an output directory name and click on *Run*.

GCX will never overwrite the original frames, so we will need to create a second set of reduced files. However, unless we have some use for the reduces frames, we can avoid saving them and just run the reduction every time we use the frames. For instance, we can align and stack the frame, or run aperture photometry on the reduced frames by just specifying the bias/dark/flats together with the required operation.

6.6.6 Aligning and Stacking Frames

When combining data frames, it is often required that they are aligned before their values are combined. GCX contains an automatic alignment algorithm that works well with frames taken with the same setup (for instance multiple exposures). In the current version, the alignment routine only performs image translations (it does not rotate or scale the images).[10]

To align frames, they need to be in the file list. Then, in the *Alignment* tab select an alignment frame file name. This is the frame used as a reference—all others will be shifted to match this one. Clicking the *Show Alignment Stars* button will detect suitable stars from the alignment frame and display them in the main window.[11]

[10] The matching algorithm does provide rotation and scale information, so this option is planned to be added in a future version.

[11] The stars will only show if an image frame is currently being displayed.

Clicking on *Run* will perform the actual alignment. Stars are detected from each frame, and their position matched to the reference frame.[12] The frame will then be translated the appropiate amount. It is also possible to apply a gaussian smoothing filter here. Usually small FWHMs work best (0.1-0.5 pixels).

After the frames are aligned, we can use the *Display* and *Next* buttons in the file list tab to browse the aligned files. Stars should show well centered inside the alignment star markings. This is also a good time to exclude any bad frames (such as frames with tracking problems).

When satisfied with the aligned frames, we can stack them and produce a final image. Just enable the stack check box and click on *Run* again.

6.6.7 Running CCD Reductions from the Command Line

All CCD reduction operations are also accessible from the command line, which makes it easy to integrate GCX with other applications. The command-line options are described by calling

```
gcx --help
```

Here we'll provide some examples of use. Let's assume that we have a number of 20-second data frames, some 20-second dark frames, some 1-sec sky flats, and some bias frames. We want to reduce the data frames and align and stack them together.

First, we check that the stacking method and parameters are properly set in the *CCD Reduction Options* page or the `/.gcxrc` file.

We combine the bias frames to create a master bias file (`m-bias.fits`):

```
gcx -s -o m-bias bias*
```

Then we combine the dark frames to create the master dark frame for the data frames (`m-dark-20.fits`):

```
gcx -s -o m-dark-20 dark*
```

We create a bias-subtracted master dark frame and scale it to be used on the flats (`dark-1s.fits`):

```
gcx -b m-bias -s -M 0.05 -o dark-1s dark*
```

And then the master flat frame (`m-flat.fits`):

```
gcx -b m-bias -d dark-1s -F -o m-flat flat*
```

[12] On fields with a large number of stars, it is sometimes possible that the default settings result in a bad match. If this is the case (a good indication of this is an unusually hish shift displayed in the Run report) increase the *Wcs Fitting Options/Minimum number of pairs* value by a few units.

Assuming that `red` is a directory, we reduce all frames and save them there:

```
gcx -d m-dark-20 -f m-flat -o red data*
```

We align and stack the data files (with a 0.1 pixels FWHM gaussian blur):

```
gcx -a red/data001 -s -G 0.1 -o stack1 red/data*
```

As an alternative, we align and stack directly from the original files, this time without any blur:

```
gcx -d m-dark-20 -f m-flat -a data001 -s -o stack1 data*
```

The order of the command-line options is not important. CCD reduction operations are always performed in the order: bias, dark, flat, multiply by a constant, add a constant, align and gaussian blur, stack. If no extension is provided on the file names, GCX will append `.fits` or `.fits.gz` automatically on saved image files, and will look for one of `.fits` or `.fit` with or without a `.gz` suffix.

Chapter 7

Aperture Photometry

The basic funtion of the aperture photometry routine in GCX is to measure the flux of a number of stars in the image (which will be expressed as an *instrumental magnitude*), and estimate it's expected error.

As an additional function, if some of the stars are *standard stars* of known magnitude the program will calculate the standard magnitude of the measured stars using the standard stars as a reference (ensemble photometry).[1]

7.1 Measuring Apertures

To measure the flux of a star, we add together the intensity values from a circular region around the target star (the central aperture), and subtract the estimated background contribution. The background is estimated from values in a annular region surrounding the star at a distance.

Normally, we choose the size of the central aperture to be large enough to include most of the star image. Common ranges are between 3 and 5 times the FWHM of the star image.[2]

The annular sky aperture is chosen to be far enough from the star so that it includes an insignificant amount of flux from it. The default values of the measurement aperture radiuses are 6 pixels for the central aperture and 9/13 pixels for the sky aperture. These values are appropiate for star images

[1] Without taking the color of the stars into account. Computing and applying color transformation coefficients requires using the information from multiple frames, taken with different filters. The output of the aperture photometry operation can be fed into the multi-frame reduction routine which takes care of that.

[2] Choosing too large an aperture has two drawbacks: it is more likely for the aperture to include unwanted stars; and the signal to noise ratio is degraded, especially for faint stars. A common way around this is to emply variable-size apertures, and correct the data for the amount of light that falls outside the aperture. While this method can give good results, it requires that the star images be well-sampled and uniform is shape across the frame. It also needs a good model for the light sensitivity distribution inside a pixel. Neither of the above conditions are met by common amateur setups, so great care is required when working with variable apertures.

between 2.5 and 3.5 pixels FWHM. If one obtains consistently tighter star images, reducing the central aperture would help improve the SNR of faint stars. The three radiuses are specified by options under *Aperture Photometry Options*.

Assuming there are N pixels inside the central aperture, the total flux (star + background) is:

$$F_T = \sum_i I_i \tag{7.1}$$

Where the sumation is taken over the pixels in the central aperture. If $\widetilde{B}$ is the estimated background level, the star's flux is taken to be:

$$F = \sum_i I_i - N\widetilde{B} \tag{7.2}$$

and the star's instrumental magnitude is:

$$M_I = -2.511886 \log\left(\sum_i I_i - N\widetilde{B}\right) \tag{7.3}$$

The estimated error of the instrumental magnitude is calculated taking most known random error sources into account. A detailed description of the error model and the way the instrumental magnitude error is calculated can be found in Appendix A.

7.2 Sky Estimation

To calculate the instrumental magnitude above we used an estimate of the sky background near the star. This value is calculated from the pixels in the annular ring.

Given the relatively large size of the sky annulus, it is very likely that we will find unwanted stars in at least some of the annuli. We must therefore use a robust algorithm to obtain the expected sky value.

The program offers a number of algorithms: *average, median, mean-median, κ-σ and synthetic mode*. The first four are described in Section 6.5.[3]

It is generally not recommended to use average, as it is not robust. The others, while not having a problem with robustness, will not produce the best estimate (which is the *mode*[4] of the sky annulus pixel values) when the distribution of the sky values is skewed. In this case (which arises whenever the sky level is relatively low), the synthetic mode is the best algorithm. The synthetic mode is calculated as follows:

The histogram of the sky values is created. Then, the histogram is clipped using a κ-σ algorithm in order to eliminate the effect of unwanted

[3] mean-median has been deprecated as a sky estimation method in recent versions.

[4] The most frequent value

stars and other defects. The mean and median of the clipped histogram are computed, and the synthetic mode is defined as:

$$\widetilde{B} = \text{mode} = 3 \cdot \text{median} - 2 \cdot \text{mean} \tag{7.4}$$

If the distribution is not skewed, the mean equals the median, and the mode would be equal to both.

The desired sky estimation algorithm is selected by the *Aperture Photometry Options/Sky method* option. The rejection band for the κ-σ and synthetic mode algorithms is set by *Aperture Photometry Options/Sigmas*.

Methods that rely on outlier rejection work best when a relatively large sky aperture is used.

7.2.1 Region Growing

Spurious stars in the sky annulus present a problem. While their peaks are easily rejected by the clipping algorithm, their "tails" are not, and will affect the estimated sky value. To avoid this, GCX can grow the area of rejected pixels, by including all pixels within a given radius of an outlier. This option can be enabled by setting the *Aperture Photometry Options/Region growing* option to a value larger than 0. Region growing is only applied when the κ-σ and synthetic mode algorithms are used.

7.3 Placing the Apertures

In GCX all photometry targets are specified using their world coordinates (right ascension, declination and epoch). The targets and standards are generally taken from a particular star file called a *recipe file*. The WCS of the frame is fitted, then the coordinates of the standards and targets are transformed to frame coordinates. The resulting positions are used as initial positions for the measuring apertures.

If the *Aperture Photometry Options/Center apertures* option is set the program will try to detect stars in the immediate vicinity of the initial positions, and center the apertures on the detected stars. The maximum distance from the initial position to the detected star is specified by *Aperture Photometry Options/Max centering error*. If this value is exceeded, the star is marked with the *not found* flag and the aperture is not moved. Otherwise it is marked with the *centered* flag.

If the apertures were centered, the amount by which each star was moved is indicated by a line extending from the center of the star symbol in the direction in which the star was moved. The length of the line is a factor of *Star Display Options/Plot error scale* longer than the star's displacement.

7.4 Finding the Ensemble Photometry Solution

If we have the instrumental magnitudes of the target stars and at least one standard star, we can calculate the standard magnitude of our targets[5] by simply adding the standard magnitude to the difference in instrumental magnitudes between the target and the standard. This is the simplest form of differential photometry.

We can however obtain significant advantages using more than one standard in the reduction:

- The errors of the standard stars will average out. This reduces both the contribution of their instrumental magnitude error and that of their standard magnitude error. It also reduces the contribution of the conformity error that is caused by the stars having different colors.
- Even more importantly, using several standards in reduction will provide valuable information about the quality of the frame. The instrumental magnitudes of the standard stars have to follow their standard magnitudes within limits set by the expected errors. If this doesn't happen we know that there was a problem with the frame. If it does happen, we are almost certain that the frame has good data.

We try to find the best estimate of the *frame zero point*, i.e. the value which is added to the instrumental magnitudes to obtain standard magnitudes. If we had no errors, all the standard stars' instrumental magnitudes would differ from their standard magnitudes by exactly the zero point value. This of course is never the case in practice. The differences will be dispersed above and below the zero point. We call the difference between a standard star's standard magnitude and the sum of it's instrumental magnitude and the zero point the star's *residual*.

We want to choose the zero point is such a way that the residuals are minimised. More specifically, we try to minimize the sum of the residuals' squares. It is easy to see that the residuals' sum of squares is minimised if the zero point is chosen so that the average of the residuals is zero.

There are two problems with this approach: First, by using many standards, we have a good chance that a few of the have "bad" values. They could be affected by a cosmic ray hit or a speck of dust that wasn't there when the flat was taken, or the catalog value may be in error. Or one of the standards may turn out to be variable. Secondly, if we use both bright and faint standards, the errors of the brighter ones are known to be lower. We would like the faint stars to have less influence on the resulting zero point than the bright ones.

[5] This will still not be the "true" magnitude of the star, because we haven't taken the colors of the standard and target into account. It is however the best we can do from a single frame. See Chapter 8 for fitting color transformation coefficients and other multi-frame reduction operations.

The algorithm used takes care of both these problems. It assigns weights to each standard star according to it's estimated error, and iteratively down-weights stars that have residuals that are larger than expected. For a detailed description, see Appendix B.

The algorithm produces it's best estimate of the frame's zero point, and a "diagnostic" value called the *mean error of unit weight*, usually abbreviated to *meu* or *me1*. The mean error of unit weight is a number that shows how well the spread of the residuals matches the estimated errors. It should have a value close to unity. A larger value shows that we have some error sources we didn't take into account. A consistently smaller value indicates that our error estimating parameters are overrated, and the estimated errors are too large.

Finally, the standard magnitude of the target stars is calculated by adding their instrumental magnitude to the estimated zero point. The error is the quadrature sum of the target's instrumental magnitude error and the zeropoint error.[6]

7.5 Annotations

The instrumental magnitude obtained is given the name of the filter the frame was taken with. The filter name is obtained from the `FILTER` field. If the field is not present, or the *Aperture Photometry Options/Force iband* option is set, the filter name is taken from *Aperture Photometry Options/Instrumental band*. If any pixel within the central aperture exceeds *Aperture Photometry Options/Saturation limit* the star is marked with the *bright* flag.

Relevant information from the fits header and recipe header is carried on to the observation report. The fields include:

object	the name of the target object, taken from the fits header or recipe.
ra, dec	World coordinates of target object.
equinox	Equinox of world coordinates and star coordintes in the report.
mjd	Modified Julian Date of integration start from `JDATE` or `MJD` fits fields.
exptime	Integration time from the `EXPTIME` field.
airmass	Frame airmass, from either the `AIRMASS` field or calculated from the geographical coordinates and time.
aperture	Telescope aperture from the `APERT` field.
telescope	Telescope name from the `TELESCOP` field.
filter	Filter used from the `FILTER` field or as set by the user.

[6] When using multiple standard stars, we have the "luxury" of using the actual spread of their magnitudes to calculate the zeropoint error, as opposed to the estimated errors.

latitude	Location of the observing site from the `LAT-OBS` field.
longitude	Location of the observing site from the `LONG-OBS` field.
altitude	Altitude of the observing site from the `ALT-OBS` field.
observer	Name of observer from the `OBSERVER` field.
sequence	A string describing where the sequence in the recipe originated, from the `sequence` field of the recipe.

7.6 Running Aperture Photometry

To run the aperture photometry routine on a frame, load the frame into gcx (*File/Open Fits*), then load a recipe file or another star file that contains standard and target stars.[7]

Then fit the frame's WCS using *Wcs/Auto Wcs* and finally run the aperture photometry routine with *Processing/Aperture Photometry to File*. A report file will be created, that lists all the standard and target stars with their instrumental and standard magnitudes, general information about the frame and fit information. More details about the report format can be found in Appendix C.

When reducing a large number of frames, it is more convenient to invoke GCX from the command line, perhaps from a script. To reduce frame `frame.fits` using the recipe file `vs.rcp` and append the format at the end of the `rep.out` file, we can use:

```
gcx -P vs.rcp -o rep.out frame.fits
```

In addition, if we have a master dark frame `mdark.fits` and a master flat frame `mflat.fits`, we can combine CCD reduction for the frame with the aperture photometry, like this:[8]

```
gcx -d mdark -f mflat -P vs.rcp -o rep.out frame.fits
```

Aperture photometry reports from several frames can be combined by simply concatenating the files together. The combined file can be used for further refining the data reduction with the multi-frame reduction routine (Chapter 8).

Selected information from the (combined) report file can be set out in a tabular format using the report converter function of GCX. The format of the table is specified in the *File and Device Options/Report converter output*

[7] A recipe file can be created "on the fly" by marking stars and then editing them to enter the standard magnitudes and types (standard and target). A valid WCS is required before being able to edit frame stars. In a pinch, using *Wcs/Force Validate* will make the program think it has one. Of course, all the coordinates will be off in the report file. It is much better to create a proper recipe file first (Section 7.7).

[8] This way we don't need to keep CCD-reduced version of the data frame, but rather work directly on the original frames.

format option. Possible values for the format are described in Appendix E and the on-line help. After setting the format,[9] invoke the report converter using:

```
gcx -T rep.out -o rep.txt
```

Which will convert the report file `rep.out` to a table named `rep.txt`.

7.7 Creating Recipe Files

Having a recipe file is central to running aperture photometry in GCX. Fortunately, creating one is relatively straightforward.

Let's create a recipe file for the `uori-v-001.fits.gz` frame, which is included in the GCX distribution. Open the frame and match the WCS (using *Wcs/Auto Wcs*). The WCS matching command leaves the GSC field stars and the detected stars visible.

7.7.1 Target Stars

First, we add our target: select *Stars/Add from Catalog*, and enter it's name at the prompt (uori). An object symbol will appear on the screen (around the bright star near the center, which is U Orionis). Select it, and bring up the star editing dialog using *Stars/Edit* or right-click on the star and select *Edit Star* from the pop-up menu. Change the star's type to "AP Target" and click *Ok*. The symbol on the image should change to a big cross, indicating the star is a target.

If we don't have GCVS installed, we can identify the star from a star chart and edit a field star or even a detected star and make it the target. We normally want to change the star's name to something descriptive and check that the coordinates are correct.

If there is no star at the desired position (which can happen if we prepare a recipe for a very faint variable), just edit any star, change the coordinates to the desired ones and the type to target. A recipe can have any number of targets; more can be added in the same way.

7.7.2 Standard Stars

Now we need some standard stars. If we have a chart we want to use as the base of the recipe, we can create it on-screen similarly with the target (by editing field stars). The difference is that the standards are marked as "Standard Star", and we need to enter their standard magnitudes. Several magnitudes can be entered in the "Standard magnitudes" field of the edit dialog. A magnitude is given as:

[9] The output format can be set from the command line using the `-S` option.

`<band>(source)=<magnitude>/<error>`

where *<band>* is the name of the standard band,[10] *(source)* is an optional field describing where the magnitude value originated, *<magnitude>* and *<error>* are the star's magnitude and error, respectively. The error field is optional, and it's absence means that we don't know the error of the magnitude. A few examples of magnitude entries are:

`v(aavso)=12.5`	A typical example for a value taken from a paper aavso chart; the error is unknown.
`v=12.53/0.05 ic=11.2/0.03`	A star for which we know the magnitudes in two bands.
`b=13.2 v=12.7/0.1 r=12.2`	We know three magnitudes, but only one error.

Another way to get standard stars is to use the tycho catalog. Remove all field stars, then select *File/Load Field Stars/From Tycho2 Catalog*. The Tycho stars will show up as field stars. All we have to do is to mark the ones that we want to use as standards.

7.7.3 Creating the Recipe File

Now we can finally create the recipe file. Select *File/Create Recipe* and enter a recipe name (or press the "..." button and select a file name). Then select which stars we want to include in the recipe file. We will most certainly want the standard and target stars, but we may include objects and field stars to be used for WCS matching if we envision using the recipe on a machine that doesn't have catalogs installed.

To verify our newly created recipe, remove all stars (*Stars/Remove All*) and load the file we just created (using *File/Load Recipe*). Run the photometry routine (*Processing/Aperture Photometry to File*) and check the output.

7.7.4 Working without an Image Frame

In the above examples, we have used a frame of the field as a backgound on top of which we loaded the stars. This is not required. If we select *Stars/Add from catalog* without having a frame loaded, the program will create a blank frame with the size set by *File and Device Options/New frame width* and *height*, and set it's WCS with the center of the frame pointing at the selected object, and the scale as set by *File and Device Options/New frame scale*.

[10] The band names are case-insensitive.

7.7.5 Creating Recipies from the Command Line

If we want to create many recipies at a time, it can be more convenient to use the command line. To create a recipe from the Tycho2 catalog, use:

```
gcx --make-tycho-rcp 20 -j uori -o uori.rcp
```

This will create a recipe using Tycho stars situated within a 20 minutes radius from U Orionis and save the result to `uori.rcp`.

If we have a sequence file in a format supported by GCX , such as the "`.dat`" files made available by Arne Henden at:

```
ftp://ftp.nofs.navy.mil/pub/outgoing/aah/sequence
```

it can be converted to a recipe file using the following command:

```
gcx --import henden <uori.dat --mag-limit 15 |\
gcx -p - --set-target uori >uori.rcp
```

The first part of the command reads the `uori.dat` file and converts it to a GCX star file, keeping only stars brighter that the 15^{th} magnitude, and writes the star file to the standard output. The second parts of the command reads the the file from the standard input, adds "uori" as a target, and writes the resulting rcp file to uori.rcp.

Chapter 8

Multi-Frame and All-Sky Reduction

If we want to determine star colors, calculate transformation coefficients to transform data to a standard system or obtain magnitudes of stars for which we don't have standards in the same field, we must reduce multiple observation frames together.

People fortunate enough to observe in photometric conditions can use a number of packages to reduce their data. For low altitude dwellers, the selection is not that large. For them, GCX implements multiple-frame reduction routines that are designed to work in less than perfect conditions.

Input data to the multi-frame reduction consists of observation reports as produced by the aperture photometry routine. For color coefficient fitting and transformation to a standard system, we need frames of the target objects taken in enough bands. For all-sky reductions, the observation reports need to have accurate time and airmass information (which implies that the original frames need to have enough information for the airmass determination).

8.1 Color Transformation Coefficients

To keep notation simple, let's assume that we reduce data taken in B and V. We'll use B and V for the standard magnitudes, and b and v for the instrumental magnitudes. The expressions for the standard magnitudes are:[1]

$$B_j = b_j + Z_k + k_B(B_j - V_j) \tag{8.1}$$

$$V_j = v_j + Z_k + k_V(B_j - V_j) \tag{8.2}$$

[1]These assume that a linear color transformation coefficient is enough to transform the data. While this is a largely used assumption, it is by no means guaranteed for any data set. One should carefully check the data to determine if a linear transformation is appropiate

The j subscripts go over all individual star observations in a given band, while the k subscripts iterate over the observation frames. For standard stars, the B_j and V_j are known while b_j and v_j are measured from the frames themselves. We have to fit the zeropoints Z_k and the transformation coefficients k. Because we have chosen to express the transformation coefficients in function of the standard magnitudes, we can proceed with one band at a time. We'll assume it is V. The steps are:

1. Set an starting value of the transformation coefficient. We can start with 0 without problems, as the coefficients are generally small numbers.

2. For each V frame, fit the zeropoint using the robust algorithm in Appendix B, with the difference that the current color transformation is applied when calculating the residuals, so Equation B.5 becomes:

$$\rho_j = y_j - \widetilde{Z_k} - k_V(B_j - V_j) \tag{8.3}$$

   Note that in this equation, the j subscripts iterate over the stars in frame k.

3. Now, for all the stars in all the V frames, estimate the "tilt" of the residuals, and adjust the transformation coefficient and zeropoints accordingly. We use the weights from the individual frames' fits when estimating the tilt:

$$\theta = \frac{\sum_j \rho_j (B_j - V_j) W'_j}{\sum_j (B_j - V_j)^2 W'_j} \tag{8.4}$$

$$k_V = k_V + \theta \tag{8.5}$$

$$Z_k = Z_k + \theta \frac{\sum_j (B_j - V_j) W'_j}{\sum_j W'_j} \tag{8.6}$$

4. Iterate the last two steps until the solution converges (the transformation coefficient doesn't change significantly).

We have now obtained the transformation coefficient for the V magnitudes, and also adjusted all the frame zeropoints so that their dependence on the color of the standard stars in each frame is eliminated.[2] The above is repeated for each band we want to reduce. Note that we have obtained the transformation coefficients without assuming any relation between the zero points of various frames—just differential photometry.

[2]When we fit frame zero points without taking the transformation coefficients, the zero points are "exact" for stars having the color index equal to the mean color index of the standard stars in each frame. The adjusted zeropoints are "exact" for stars with a color index of zero.

We can choose any color index for a given band. For instance, there is nothing stopping us from calculating a $(B-V)$ transformation coefficient for I or R magnitudes. In fact, if we have more standards data in B and V, it may prove better to do so. In general, Equation (8.1) can be written for any band M as:

$$M_j = m_j + Z_k + k_M(C_1^M - C_2^M) \tag{8.7}$$

where C_1^M and C_2^M are any bands for which we have standards data. We can fit the transformation coefficient k_M using only the M observations. However, when we want to transform the stars, we will need observations in M, C_1^M, C_2^M and all the bands that these depend on.

8.1.1 Transforming the Stars

To calculate the transformed standard magnitudes of our target stars, all we have to do is to write Equation (8.7) for each band, and solve the resulting system of linear equations for the standard magnitudes. The system is very well behaved (it's matrix is close to unity) so GCX uses the simple Gauss-Jordan elimination method to solve it.

8.2 All-Sky Reduction

When the field of our intended target doesn't contain any suitable standard stars, we have to determine their magnitudes by comparing to stars in a different field. To do this, we need to determine a relation between the zeropoints of different frames.

Under photometric conditions, we can consider that the *atmospheric extinction*[3] depends only on the thickness of the atmosphere along the light path. The ratio between the thickness of the atmosphere in the direction of field and it's thickness towards zenith is called the *airmass* of the field. The airmass depends on the zenital angle z of the field, and is close to $\sec(z)$ when far from the horison. The formula used by GCX is the following:[4]

$$s = \sec(z) - 1; \tag{8.8}$$

$$A = 1 + s[0.9981833 - s(0.002875 + 0.0008083s)] \tag{8.9}$$

The zenital angle of a frame can be determined given it's equatorial coordinates, the geographical coordinates of the observing site, and time.

If the extinction is unform in all directions, we can define an *extinction coefficient* E, so that for any frame:

$$Z(A) = Z_0 - EA \tag{8.10}$$

[3]The amount of light lost when passing through the atmosphere.

[4]R.H. Hardie, 1962, *Photoelectric Reductions, Chapter 8 of* Astronomical Techniques*, W.A. Hiltner (Ed), Stars and Stellar Systems, II (University of Chicago Press: Chicago), pp178-208.*

where Z_0 is the zeropoint outside the atmosphere, and A is the frame's airmass. If we have two frames with airmasses A_1 and A_2, their zeropoints can be related by:

$$Z_2 = Z_1 - E(A_2 - A_1) \tag{8.11}$$

Under photometric conditions, it is customary to determine the extinction coefficient by observing the same field at different airmasses and then fitting E from (8.11). This is only possible when E doesn't change (or changes in a smooth, linear fashion) over a period of the order of hours.

Because the GCX all-sky routine is targeted at less-than-perfect conditions, we will choose another strategy in determining the extinction coefficient. We use several standard fields located relatively near our target fields. Then we we try to "chop" the extinction coefficient as much as possible by alternating between the standard and target fields.[5]

We end up with a series of observations from different fields, all in the same general airmass range. By examining the standard fields' zeropoints variation with time and airmass, we can determine if there were any "windows" during which the extinction was stable.

Once a stable window was found, we can fit the extinction coefficient from the observations in that window. It is unlikely that the observations will span a wide range of airmasses, which will make the fitted value of the extinction coefficient somewhat imprecise. But this is offset by the fact that the airmass of the target frames is the same range, so the contribution of the extinction term is not very large. As long as the airmass of the standard fields brackets those of the target fields, we are *interpolating* rather than extrapolating the extinction.[6]

8.2.1 Extinction Coefficient Fitting

Before attempting to fit the extinction coefficient, the zeropoints and color transformation coefficient of all frames must be fitted. It is highly recommended to examine plots of the resulting zeropoints versus time and airmass to see if it's worth trying to do any all-sky reduction at all (more on this below).

With these precautions, the program will proceed to fit the extinction coefficients using a variant of the algorithm described in Appendix B,[7] with the initial weights assigned based on the calculated errors of the zeropoints.

[5] The standard fields can contain differential photometry targets.

[6] The program allows a small amount of extrapolation to take place.

[7] A robust regression, rather than averaging algorithm is used, which uses the same outlier down-weighting scheme.

The fitted model is:[8]

$$Z = \bar{Z} + E(A - \bar{A}) \tag{8.12}$$

where $\bar{A}$ is the mean airmass of the standard frames, while $\bar{Z}$ is the zeropoint of a mean airmass frame.

A different extinction coefficient is fitted for each band. Frames that are outliers of the fit (their standard error exceeds the threshold set in *Multi-Frame Photometry Options/Zeropoint outlier threshold*) are marked as such.

8.2.2 Calculating Zero Points

After fitting the extinction coefficient, we can apply Equation 8.12 and calculate the zeropoint of any target frame. The program tries to filter the frames for which such a determination would likely be in error. It will only calculate a zeropoint for frames which satisfy the following:

1. The frame has to be both preceded and succeded in time by non-outlier standard frames;

2. The frame's airmass has to be in the same range as the standard frames from which the extinction coefficient was fitted.

   If A_m is the minimum and A_M is the maximum standard airmass, and r is the value of *Multi-Frame Photometry Options/Airmass range* the zeropoint is only calculated for frames with airmasses between

   $$\frac{A_M + A_m}{2} - \frac{r}{2}(A_M - A_m) \tag{8.13}$$

   and

   $$\frac{A_M + A_m}{2} + \frac{r}{2}(A_M + A_m) \tag{8.14}$$

8.3 Running Multi-Frame Reduction

This section is a step-by-step tour of the multi-frame reduction tool. An realistically-sized example input file is provided in the distribution data directory (`cygs-aug19.out`). This file was generated by GCX aperture photometry from 143 frames taken in B, V, R and I in a single night, all in Cygnus. The standards data is from Henden sequence files, which were converted into GCX recipies with the import function. The file consists of individual aperture photometry reports appended together.

[8] It is common to include additional coefficients in the model, such as ambient or sensor temperature or time. But they are generally only effective in photometric conditions, where the extinction varies smoothly over long intervals of time.

8.3.1 Specifying Reduction Bands

Before we can reduce data, we have to define which color indices are used for each band. The *Multi-Frame Photometry Options/Bands setup* option specifies this. It contains a list of specifiers of the form: `<band>(<c1>-<c2>)` separated by spaces. Each specifier tells the program to use the color index "`<c1>-<c2>`" to reduce frames taken in "`band`". For example, the default setting:

```
b(b-v) v(b-v) r(v-r) i(v-i)
```

will set the following transformation model:

$$B = b + Z_B + k_B(B - V) \quad (8.15)$$
$$V = v + Z_V + k_V(B - V) \quad (8.16)$$
$$R = r + Z_R + k_R(V - R) \quad (8.17)$$
$$I = i + Z_I + k_I(V - I) \quad (8.18)$$

This model is appropiate if we reduce BV, BVI, BVR or BVRI data. However, it will have to be changed if for instance we want to reduce VI data, as using it will require B, V and I observations, because of the V dependence on B. An appropiate model for VI data would be:

$$V = v + Z_V + k_V(V - I) \quad (8.19)$$
$$I = i + Z_I + k_I(V - I) \quad (8.20)$$

which is specified by setting:

```
v(v-i) i(v-i)
```

in the *Bands setup* option.

The same option can be used to set initial transformation coefficients and their errors, by appending "`=<coeff>/err`" to each band specifier like for example:

```
b(b-v)=0.12/0.001 v(b-v)=-0.07/0.02
```

8.3.2 Loading Report Files

The data to be reduced can reside in one or more files. To load data, open the the multi-frame reduction dialog using *Processing/Multi-frame reduction* or **Ctrl-M**, and select *File/Add to Dataset*. Select the file name and press *Ok*. The data from the frames contained in the file will load, and the frames will appear in the "Frames" tab of the dialog.

More observations can be added by using *Add to Dataset* repeteadly.[9] We'll assume the example file (`cygs-aug19.out`) is loaded for the next steps.

[9] If we have a large numbers of files to add, it's probably easier if they are concatenated before loading (using `cat` for instance).

Frames The "Frames" tab contains a list with all the frames in the dataset, one per line. It will display increasing amounts of information as the fit progresses. The columns that are filled up right after loading the data files should be self-explanatory. The other columns show:

Zpoint	The fitted zero point of the frame;
Err	The calculated error of the zero point;
Fitted	The number of standard stars used in the fit;
Outliers	The number of standard stars that are considered outliers of the fit (have large standard errors);
MEU	The mean error of unit weight for the zeropoint fit of the frame.

Clicking on the column headers will make the program sort the list by the respective column. Clicking again will reverse the sort order. One or more frames can be selected in the list. All operations apply to the selected frames or, if none are selected, to the whole list.

Stars When a frame line is clicked, the stars from that frame are displayed in the "Stars" tab. Like the frames, the stars can be sorted on various columns by clicking on the column headers. The columns of the stars table show:

Name	The star's name. A star is identified across multiple frame by it's name;
Type	Star type (standard or target);
Band	The band the magnitudes are in;
Smag	The standard magnitude for the star. If the contents of this field are calculated by the program, as for target stars, the magnitude appears in square brackets;
Err	Error of the standard magnitude (either taken from the report file, or calculated by the program);
Imag	Instrumental magnitude in this observation;
Err	Error of the instrumental magnitude, taken from the report file;
Residual	The residual in the last fit of the frame. Only appears for standard stars;
Std Error	Standard error of the star (the residual divided by the estimated error). Only for the standard stars;
Outlier	"Y" or "N" depending on whether the star has a large standard error or not;
R.A, Dec	Star catalog position;
Flags	A list of flags that apply to the star. Some are taken from the report file, some are added by the fitting routines.

Bands The bands tab shows the currently configured bands, and the various transformation coefficients relating to these bands. They only show the fitted coefficients as they resulted from the last fit operation. If only some frames were selected in that operation, then these values may only apply to those frames.

8.3.3 Fitting Individual Zero Points

The simplest type of fit we can do is fit the zeropoints of each frame individually, without taking the other frames into consideration (like the last step in the aperture photometry routine). Even though the report files likely contained the individual fit information, it was discarded when the report was loaded. We need to perform at least this step before we can generate any plots for the data.

There are two variants of this command: one zeroes all the transformation coefficients before doing the fit (*Fit Zero Points with Null Coefficients*), while the other will apply the current transformation coefficients to the standard stars first. (*Fit Zero Points with Current Coefficients*).

Make sure the frames you want to fit are selected before applying the command (if no frames are selected, the command will apply to all frames).

After the fit, examine the MEU column, which will show the quality of the fit (the number should be around 1.0). Since we only fitted the zeropoint, and not the color coefficients the values are slightly larger than the best than can be obtained.

8.3.4 Plots

At this point, we can generate various plots, which are instrumental in judging the quality of the data, especially when we consider the more ellaborate fits. The program generates data file for the `gnuplot` utility, and will run `gnuplot` directly if the option *File and Device Options/Gnuplot command* is correctly set.[10] If the *Plot to File* option in the *Plot* menu is selected, the program will generate a data file instead of running gnuplot directly.[11]

Let's select the V frames (click on the band column header twice to bring the V band at the top, then click on the first V frame, and finaly shift-click on the last V frame). Now run *Plot/Residuals vs Magnitude*. A plot should appear that is similar to the one in Figure 8.1.

The plot generally has the familiar shape of photon-shot noise dominated observations, with random errors increasing as the stars become fainter. An additional feature of this dataset are the "branches" going up starting at around mag 12 and 11. These are caused by saturated standard stars (the

[10] The default setting will work if gnuplot is installed in the command path.

[11] The data frame is a simple ascii table with some gnuplot commands at the top; it can easily be imported in other plotting utilities or spreadsheet programs if desired.

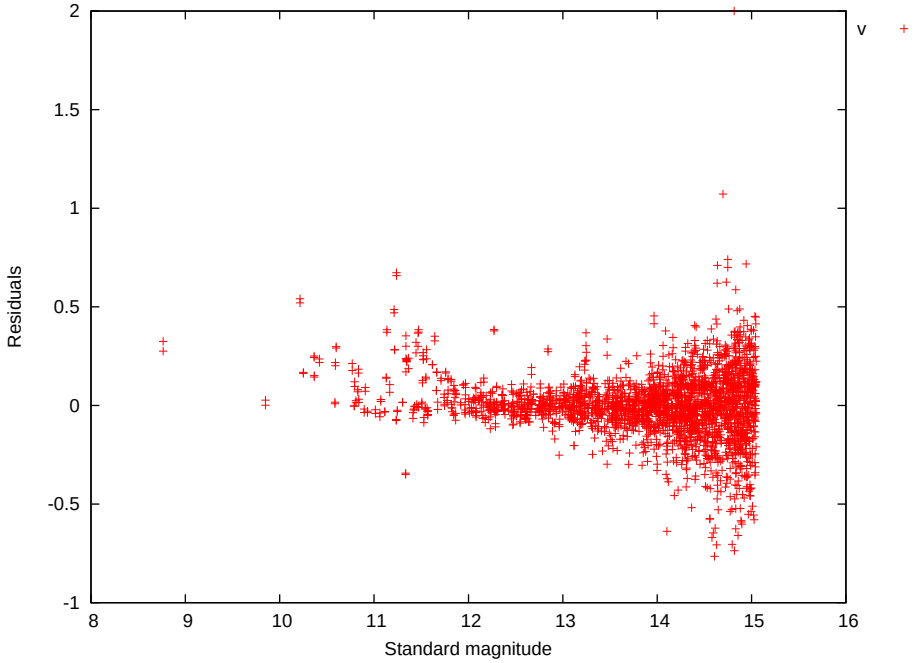

Figure 8.1: Residuals vesus standard magnitudes for all frames in the example data set, after zero point fitting without color transformation.

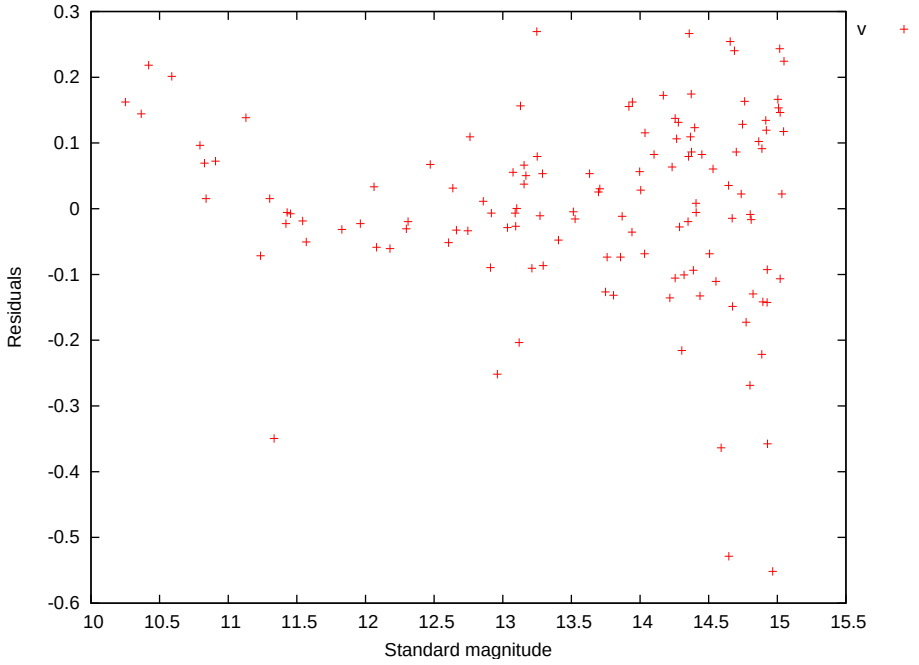

Figure 8.2: Residuals versus standard magnitudes of one AU CYG frame.

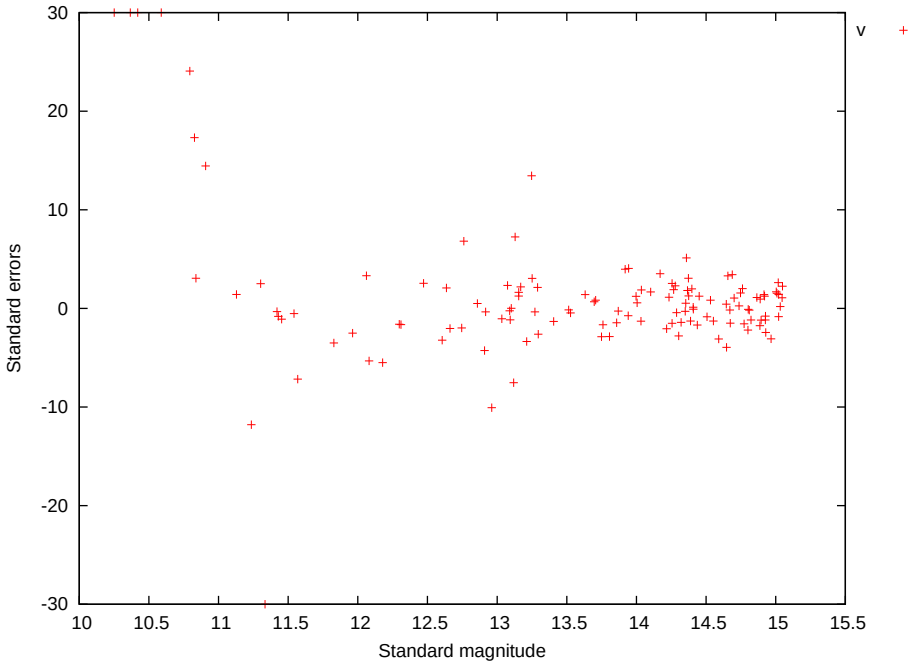

Figure 8.3: Standard errors versus magnitudes of one AU CYG frame.

standards we used are not reliable above mag 12.5 or so). If the stars would be saturated in our observations, the "branches" would go downward.[12]

To investigate the matter further, we select a frame with a large number of outliers, which is likely to contain such a "branch". For example, let's select aucyg. The *Residuals vs Magnitude* plot for this frame is shown in Figure 8.2. The bright stars branching up are obvious in this plot. However, the importance of the errors is difficult to judge, as the "normal" error changes with the stars' magnitudes (and fainter stars show similar residuals). The *Standard Errors vs Magnitude* plot comes handy in this situation. It is similar to the previous plot, only the residuals are divided by the expected error of the respective stars. We expect all stars to show similar standard errors, all within a 6 units wide band around zero. This plot is shown in Figure 8.3. We can clearly see that the relatively large residuals to the right of the plot are within normal limits (also indicated by the value of the MEU fit parameter). The "branch" is clearly deviant (with standard errors going up to 30 and more).[13] Fortunately, the robust fitting algorithm has downweighted the deviant points significantly, so the "good" values still

[12]If the data was reduced with GCX the saturated stars would normally be marked as such and excluded for the fit. If down-going branches appear, the *Saturation limit* option's value should be decreased.

[13]The plot routine clips standard errors at ±30, and residuals at ±2.

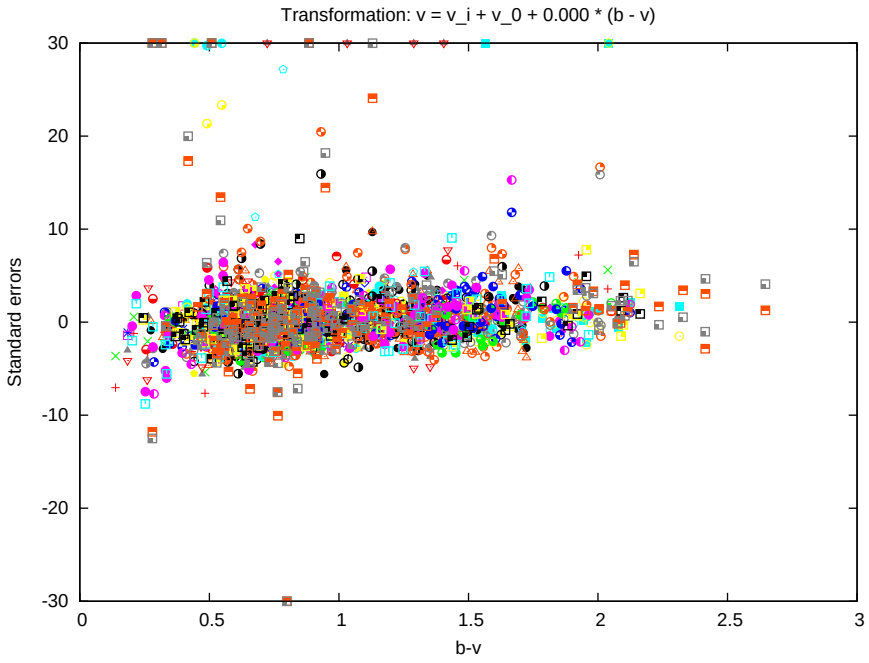

Figure 8.4: Standard errors vs color for the V frames, before transformation coefficient fitting.

spread symetrically around zero.[14]

8.3.5 Fitting Color Transformation Coefficients

With the V frames selected, let's plot the standard errors again, this time against the star's color index. For this, select *Plot/Standard Errors vs Color*. The output should look similar to Figure 8.4. Even given the scatter of the individual observations, the plot shows a clear sloping (making the residuals proportional to the color index).[15] To remove this slope and at the same time calculate the color transformation coefficient, we use *Reduce/Fit Zero Points and Transformation Coefficients*. After the fit is done, the slope is removed, as shown in Figure 8.5. The title of the figure shows the transformation used. In our case, the resulting tranformation coefficient is 0.062, a rather

[14]The most dangerous deviant points in this case are not the ones with large standard errors (which are easily detected), but the ones right near the turning point of the graph. Being bright stars, they will have small estimated errors, and can bias the solution significantly. In this particular case it didn't happen because of the large number of "normal" stars.

[15]It is important at this step to examine the data carefully and check if a simple linear transformation coefficient will remove any color trend in the data. Some data sets may show a curved dependence (for which a polynomial transformation would be better), while other can show turning points. In those cases, the linear transformation no longer holds.

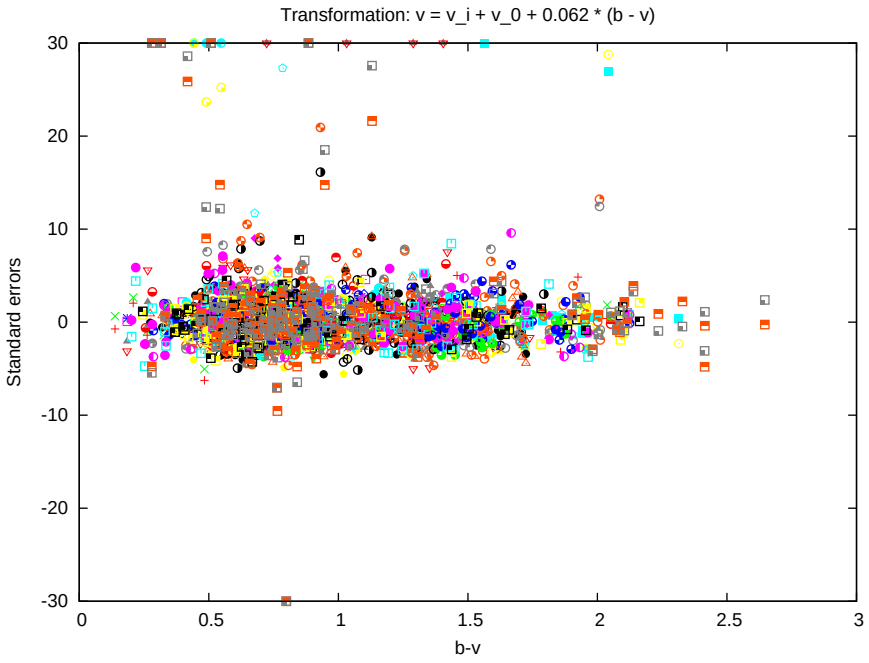

Figure 8.5: Standard errors vs color for the V frames, after transformation coefficient fitting.

small figure indicating a good fit between the filters used and the standard ones. If we check the MEU fields for each frame, we will see that they have decreased, showing that the data more closely matches the standard magnitudes after the color transformation.

After the fit, the list in the "Bands" tab is updated to show the fitted transformation coefficients and their expected errors. Note that the error is quite small in our case (0.002),[16] even though the data seemed to spread a lot. The large number of stars used in the fit helped reduce the error considerably.

A good sanity check for the transformation coefficient fit is to run the same routine on subsets of the initial data set and compare the resulting transformation coefficients. They should match within the reported error figures.

Before proceeding, let's do the transformation coefficient fit for the whole dataset: *Edit/Unselect All*, then *Reduce/Fit Zero Points and Transformation Coefficients*.

[16] This error propagates to the final magnitude in proportion with the target star's color index. An coefficient error of 0.002 will contribute a systematic error of 0.004 to a star with $B - V = 2.0$, and 0.001 to one with $B - V = 0.5$

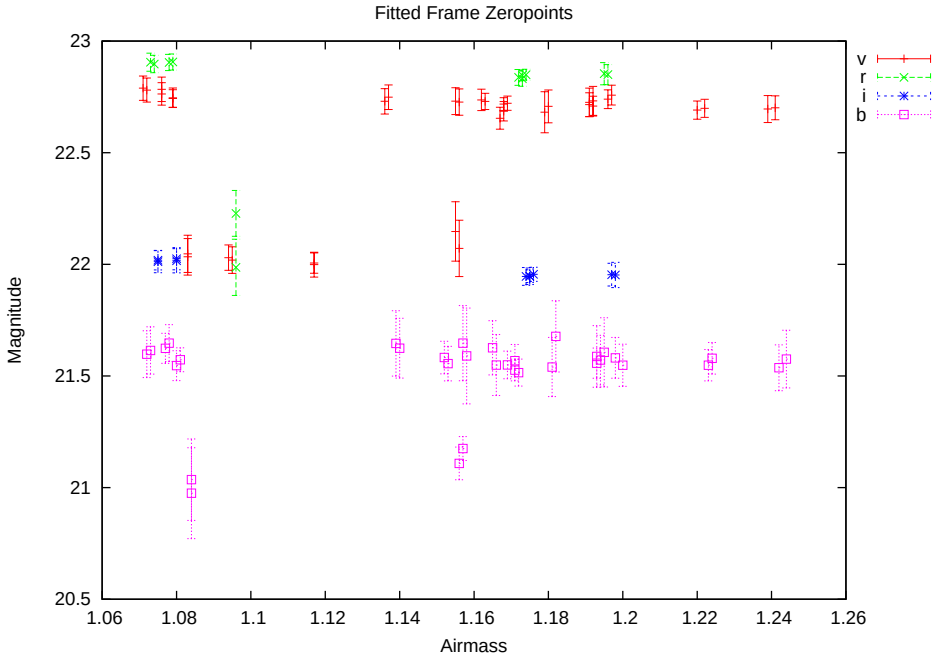

Figure 8.6: Zero points vs airmass for frames with standards data.

8.3.6 All-Sky Reduction

The example data set contains BVRI frames for all fields. However, only some of the fields have R and I standards data. The night was clear, but conditions were changing. Let's see what we can do about the R and I frames that need all-sky reduction.

We can examine the frame zero points versus the airmass, expecting them to fall on a down-sloping line.[17] Using *Plot/Zeropoints vs Airmass* will produce the plot in Figure 8.6, which shows all the bands' zeropoints on the same graph. We see that most of the frames do indeed lie on down-sloping lines with a scatter consistent with their expected errors as shows by the error bars, but there are some outliers. So the conditions weren't photometric. If we now plot the same zeropoints against time (*Plot/Zeropoints vs Time*), Figure 8.7 we can see what has happened: the transparency has improved starting at MJD 53236.95, to the point where we can use the all-sky method for frames taken after that point.

Let's run the all-sky reduction (*Reduce/Fit Extinction and All-Sky Zero Points*) and generate the plots again. As we can see in Figures 8.8 and 8.9, the program has selected the frames which are bracketed by other "good" frames, [18] and calculated their all-sky zeropoints. The all-sky frames are

[17] The zeropoints increase when the transparency improves.

[18] The status of the frames that have had a good extinction fit will end in "-AV".

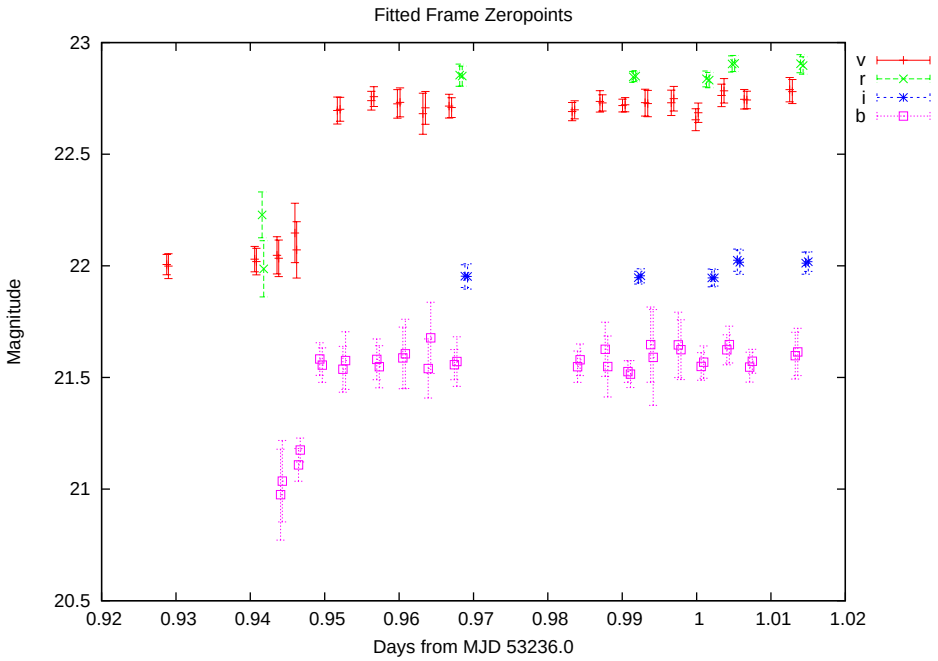

Figure 8.7: Zero points vs time for frames with standards data.

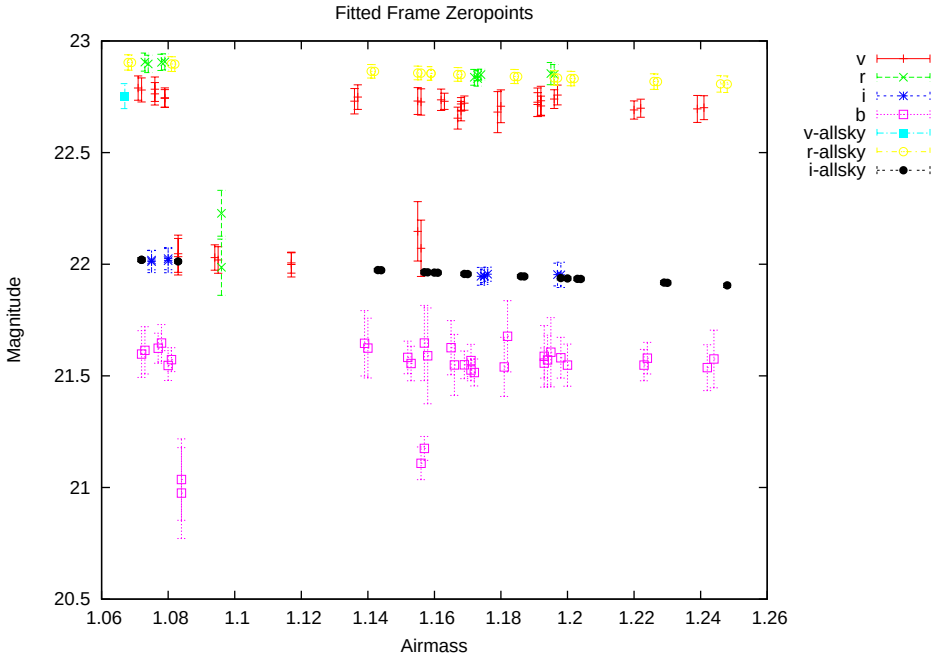

Figure 8.8: Zero points vs airmass for all frames, after the extinction fit.

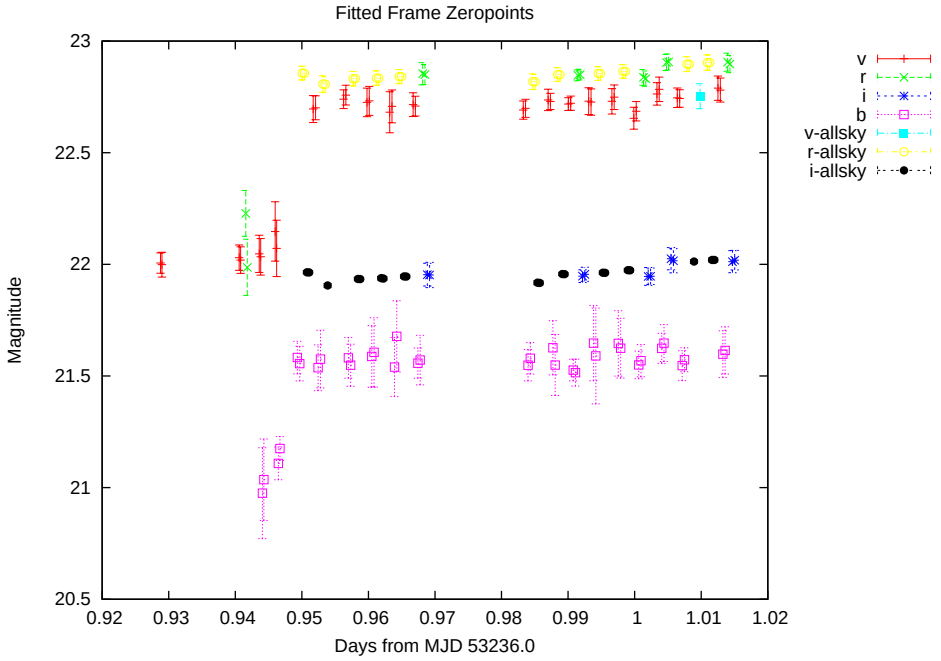

Figure 8.9: Zero points vs time for all frames, after the extinction fit.

shown with different colors. These plots should be carefully examined and any suspicious frames removed from the all-sky reduction. In our case however, it seems that the program has made a good choice of frames.

The calculated extinction coefficients and their errors are displayed in the "Bands" tab. The errors of the exctinction coefficients are relatively large. In this case, this is due to the fact that the frames are taken in a narrow range of airmasses. The same narrow range of airmasses will however reduce the impact of the errors on the calculated zeropoints. this can be seen on the graphs, where the error bars of the all-sky frames, which take the extinction coefficient errors into account, are of the same order as those of the "normal" frames.[19]

8.4 Reporting

After the fits are done, the complete dataset can be saved in the native format using *File/Save Dataset*. The native format preserves all the information in a future-proof fashion, but importing it into other applications can be a little involved.

[19]The points on the time graph can appear to wander more than the error bars would indicate. This is because the values there have not been corrected for airmass, and the airmass does not correlate with time.

The `--rep-to-table` or `-T` command-line option allows the native format to be converted into a table with fixed-width columns. The format and content of the columns are fully programmable by changing the *File and Device Options/Report converter output format* option. The following command will convert dataset.out from the native format to a table (dataset.txt) with the format as set in the option:

```
gcx -T dataset.out >dataset.txt
```

Alternatively, the table format can be specified on the command line. For example, to create a table with the stars' name, mjd of observation, and V magnitudes and errors use:

```
gcx -T dataset.out -S ".file.tab_format=name jdate \
smag 'v' serr 'v'" >dataset.txt
```

The complete format string specification can be found in Appendix E.

Finally, it is possible to list the target stars in the AAVSO format. If a validation file location is set in the *File and Device Options/Aavso validation file*, it will be searched for the designation of the stars. The observer code field will be filled in from the *general Observation Setup Data/Observer code* option.

Appendix A

Noise Modelling

The issue of noise modelling is essential in any photometric endeavour. Measurement values are next to meaningless if they aren't acompanied by a measure of ther uncertainty.

One can assume that the noise and error modelling only applies to deriving an error figure. This in true only in extremely simple cases. In general, the noise estimates will also affect the actual values. For instance, suppose that we use several standards to calibrate a field. From the noise estimate, we know that one of the standards has a large probable error. Then, we choose to exclude (or downweight) that value from the solution—this will change the calibration, and directly affect the result (not just it's noise estimate).

Precision and Accuracy. The precision of a measurement denotes the degree to which different measurements of the same value will yield the same result; it measures the repeatability of the measurement process. A precise measurement has a small *random error*.

The accuracy of a measurement denotes the degree to which a measurement result will represent the true value. The accuracy includes the *random error* of the measurement, as well as the *systematic error*.

Random errors are in a way the worst kind. We have to accept them and take into account, but they cannot be calculated out. We can try to use better equipment, or more telescope time and reduce them. On the other hand, since random errors are, well, random in nature (they don't correlate to anything), we can in principle reduce them to an aribitrarily low level by averaging a lerge number of measurements.

Systematic errors on the other hand can usually be eliminated (or at least reduced) by calibration. Systematic errors are not that different from random errors. They differ fundamentally in the fact the they depend on *something*. Of course, even random errors ultimately depend on something. But that something changes incontrollably, and in a time frame that is short

compared to the measurement time scale.

A systematic error can turn into a random error if we have no control over the thing that the error depends on, or we don't have something to calibrate against. We could treat this error as "random" and try to average many measurements to reduce it, but we have to make sure that the something that the error depends on has had a change to vary between the measurements we average, or we won't get very far.

Noise is the "randomest" source of random errors. We have no way to calibrate out noise, but it's features are well understood and relatively easy to model. One doesn't have a good excuse not to model noise reasonably well.

We will generally talk about "noise" when estimating random errors that derive from an electrical or optical noise source. Once these are combine with other error sources (like for instance expected errors of the standards), we will use the term "error". Of course, there are two ways of understanding an error value. If we know what the true value should be, we can talk about and *actual error*. If we just consider what error level we can expect, we talk about an estimated, or *expected error*.

A.1 CCD Noise Sources

There are several noise sources in a CCD sensor. We will see that in the end they can usually be modeled with just two parameters, but we list the main noise contributors for reference.

1. *Photon shot noise* is the noise associated with the random arrival of photons at any detector. Shot noise exists because of the discrete nature of light and electrical charge. The time between photon arrivals is goverened by Poisson statistics. For a phase-insensitive detector, such as a CCD,
$$\sigma_{\mathrm{ph}} = \sqrt{S_{\mathrm{ph}}} \tag{A.1}$$
where S_{ph} is the signal expressed in electrons. Shot noise is sometimes called "Poisson noise".

2. *Output amplifier noise* originates in the output amplifier of the sensor. It consists of two components: thermal (white) noise and flicker noise. Thermal noise is independent of frequency and has a mild temperature dependence (is proportional to the square root of the absolute temperature). It fundamentally originates in the thermal movement of atoms. Flicker noise (or $1/f$ noise) is strongly dependent on frequency. It originates in the existance of long-lived states in the silicon crystal (most notably "traps" at the silicon-oxide interface).

For a given readout configuration and speed, these noise sources contribute a constant level, that is also independant of the signal level, usually called the *readout noise*. The effect of read noise can be reduced by increasing the time in which the sensor is read out. There is a limit to that, as flicker noise will begin to kick in. For some cameras, one has the option of trading readout speed for a decrease in readout noise.

3. *Camera noise.* Thermal and flicker noise are also generated in the camera electronics. the noise level will be independent on the signal. While the camera designer needs to make a distiction between the various noise sources, for a given camera, noise originating in the camera and the ccd amplifier are indistinguishable.

4. *Dark current noise.* Even in the absence of light, electron-hole pairs are generated inside the sensor. The rate of generation depends exponentially on temperature (typically doubles every 6-7 degrees). The thermally generated electrons cannot be separated from photo-generated photons, and obey the same Poisson statistic, so

$$\sigma_{\text{dark}} = \sqrt{S_{\text{dark}}} \tag{A.2}$$

We can subtract the average dark signal, but the shot noise associated with it remains. The level of the dark current noise depends on temperature and integration time.

5. *Clock noise.* Fast changing clocks on the ccd can also generate spurious charge. This charge also has a shot noise component associated. However, one cannot read the sensor without clocking it, so clock noise cannot be discerned from readout noise. The clock noise is fairly constant for a given camera and readout speed, and independent of the signal level.

Examining the above list, we see that some noise sources are independent of the signal level. They are: the output amplifier noise, camera noise and clock noise. They can be combined in a single equivalent noise source. The level of this source is called *readout noise*, and is a characteristic of the camera. It can be expressed in electrons, or in the camera output units (ADU).

The rest of the noise sources are all shot noise sources. The resulting value will be:

$$\sigma_{\text{shot}} = \sqrt{\sigma_{\text{ph}}^2 + \sigma_{\text{dark}}^2} \tag{A.3}$$

$$\sigma_{\text{shot}} = \sqrt{S_{\text{ph}} + S_{\text{dark}}} = \sqrt{S} \tag{A.4}$$

S is the total signal from the sensor above bias, expressed in electrons. So to calculate the shot noise component, we just need to know how many

ADUs/electron the camera produces. This is a constant value, or one of a few constant values for cameras that have different gain settings. We will use A to denote this value.

A.2 Noise of a Pixel Value

We will now try to model the level of noise in a pixel value. The result of reading one pixel (excluding noise) is:

$$s = s_b + A(S_d + S_p) \tag{A.5}$$

where s_b is a fixed bias introduced by the camera electronics, S_d is the number of dark electrons, and S_p is the number of photo-generated electrons (which is the number of photons incident on the pixel multiplied by the sensor's quantum efficiency).

Now let's calculate the noise associated with this value.

$$\sigma^2 = \sigma_r^2 + A^2(S_d + S_p) = \sigma_r^2 + A(s - s_b) \tag{A.6}$$

Where σ_r is the readout noise expressed in ADU, and S is the total signal expressed in electrons. Note that we cannot calculate the noise if we don't know the bias value. The bias can be determined by reading frames with zero exposure time (bias frames). These will contribute some read noise though. By averaging several bias frames, the noise contribution can be reduced. Another approach is to take the average across a bias frame and use that value for the noise calculation of all pixels. Except for very non-uniform sensors this approach works well. GCX supports both ways.

Note that a bias frame will only contain readout noise. By calculating the standard deviation of pixels across the difference between two bias frames we obtain $\sqrt{2}$ times the readout noise.

A.3 Dark Frame Subtraction

A common situation is when one subtracts a dark frame, but doesn't use bias frames. The noise associated with the dark frame is:

$$\sigma_d^2 = \sigma_r^2 + A^2 S_d \tag{A.7}$$

The resulting pixel noise after dark frame subtraction will be:

$$\sigma_{\mathrm{ds}}^2 = 2\sigma_r^2 + A^2(2S_d + S_p) \tag{A.8}$$

while the signal will be

$$s_{\mathrm{ds}} = AS_p \tag{A.9}$$

Using just the camera noise parameters, we cannot determine the noise anymore. We have to keep track of the dark subtraction and it's noise effects. We however rewrite the dark-subtracted noise equation as follows:

$$\sigma_{\mathrm{ds}}^2 = \left(\sqrt{2\sigma_r^2 + 2A^2 S_d}\right)^2 + A^2 S_p \tag{A.10}$$

If we use the notation $\sigma_r' = \sqrt{2\sigma_r^2 + 2A^2 S_d}$, we get:

$$\sigma_{\mathrm{ds}}^2 = \sigma_r'^2 + A^2 S_p \tag{A.11}$$

This is identical in form to the simple pixel noise equation, except that the true camera readout noise is replaced by the equivalent read noise σ_r'. What's more, the bias is no longer an issue, as it doesn't appeear in the signal equation anymore. We can derive the pixel noise from the signal directly, as:

$$\sigma_{\mathrm{ds}}^2 = \sigma_r'^2 + A s_{\mathrm{ds}} \tag{A.12}$$

The same parameters, σ_r' and A are sufficient to describe the noise in the dark-subtracted frame.

A.4 Flat Fielding

To flat-field a frame, we divide the dark-subtracted pixel value s_{ds} by the flat field value f. The noise of the flat field is σ_f. The resulting signal value is

$$s_{\mathrm{ff}} = \frac{1}{f} A S_p \tag{A.13}$$

If we neglect second-order noise terms, the noise of the flat-fielded, dark subtracted pixel is:

$$\sigma_{\mathrm{ff}}^2 = f\sigma_r'^2 + A^2 S_p + \left(\frac{\sigma_f}{f} A S_p\right)^2 \tag{A.14}$$

$$\sigma_{\mathrm{ff}}^2 = f^2\sigma_r'^2 + A f s_{\mathrm{ff}} + (\sigma_f s_{\mathrm{ff}})^2 \tag{A.15}$$

The problem with this result is that f is not constant across the frame. So in general, the noise of a flat-fielded frame cannot be described by a small number of parameters. In many cases though, f doesn't vary too much across the frame. We can then use it's average value, $\widetilde{f}$ for the noise calculation. This is the approach taken by the program.

We can identify the previous noise parameters, $\sigma_r'' = \widetilde{f}\sigma_r'$ and $A' = A\widetilde{f}$. For specifing the effect of the flat-fielding, we introduce a new parameter, σ_f.

Without reducing generality, we can arrange for $\widetilde{f} = 1$. This means that the average values on the frames don't change with the flatfielding operation, and is a common choice.

In this case, σ_r and A aren't affected by the flatfielding operation, while the third noise parameter becomes $\sigma_f/\widetilde{f}$, which is the reciprocal of the SNR of the flat field.

GCX models the noise of each pixel in the frame by four parameters: σ_r, A, $\sigma_f/\widetilde{f}$ and $\widetilde{s_b}$. The noise function $n(s)$ of each pixel is:

$$n^2(s) = \sigma^2 = \sigma_r^2 + A|(s - \widetilde{s_b})| + \left(\frac{\sigma_f}{\widetilde{f}}\right)^2 (s - \widetilde{s_b})^2 \tag{A.16}$$

σ_r comes from the `RDNOISE` field in the frame header. A is the reciprocal of the value of the `ELADU` field. $\sigma_f/\widetilde{f}$ comes from `FLNOISE`, while $\widetilde{s_b}$ comes from `DCBIAS`.

Every time frames are processed (dark and bias subtracted, flatfielded, scaled etc), the noise parameters are updated.

A.5 Instrumental Magnitude Error of a Star

Once we know the noise of each pixel, deriving the expected error of an instrumental magnitude is straightforward. Let N_b be the number of pixels in the sky annulus, and s_i the level of each pixel. The noise of the sky estimate is:[1]

$$\sigma_b^2 = \frac{1}{N_b} \sum_{i=1}^{N_b} n^2(s_i) \tag{A.17}$$

Now let N_s be the number of pixels in the central aperture. The noise from these pixels is:

$$\sigma_s^2 = \sum_{i=1}^{N_s} n^2(s_i) \tag{A.18}$$

The total noise after sky subtraction will be:

$$\sigma_n^2 = \sigma_s^2 + N_s\sigma_b^2. \tag{A.19}$$

The program keeps track and reports separately the photon shot noise, the sky noise, the read noise contribution and the scintillation noise.

Scintillation is an atmospheric effect, which results in a random variation of the received flux from a star. We use the following formula for scintillation noise:

$$\sigma_{\text{sc}} = 0.09F\frac{A^{1.75}}{D^{\frac{2}{3}}\sqrt{2t}} \tag{A.20}$$

[1] This assumes that the method used for sky estimation has a statistical efficiency close to the mean, which isn't generally the case. Perhaps this should be taken into account, at least for methods whose efficiency is well known, like the median.

Where F is the total flux received from the star, A is the airmass of the observation, D is the telescope aperture in cm, and t is the integration time. Scintillation varies widely over time, so the above is just an estimate.

Finally, we can calculate the expected error of the instrumental magnitude as

$$\epsilon_i = 2.51188 \log\left(1 + \frac{\sqrt{\sigma_n^2 + \sigma_{\text{sc}}^2}}{F}\right). \tag{A.21}$$

Appendix B

Robust Averaging

A robust averaging algorithm is implemented by GCX and used in several places, most notably for zeropoint fitting by the aperture photometry and multiframe reduction routines. The algorithm calculates the robust average of a number of values (for the zeropoint routines, these are the differences between the standard and instrumental magnitudes of standard stars).

The data used consists of the values we want to calculate, and the estimated error of each value. For fitting frame zeropoints they are:

$$y_k = S_k - I_k \tag{B.1}$$
$$\epsilon_k^2 = \epsilon_{ik}^2 + \epsilon_{sk}^2 \tag{B.2}$$

where S is the standard magnitude, I is the instrumental magnitude, ϵ_i is the estimated error of the instrumental magnitude, ϵ_s is the error of the standard magnitude of each star. Each star is assigned a *natural weight*, calculated as

$$W_k = \frac{1}{\epsilon_k^2} \tag{B.3}$$

We start with a very robust estimate of the average:

$$\widetilde{Z} = \text{median}(y_k) \tag{B.4}$$

and calculate the *residuals* of each value:

$$\rho_k = y_k - \widetilde{Z} \tag{B.5}$$

and the *standard errors*:

$$\rho'_k = (y_k - \widetilde{Z})\sqrt{W_k} \tag{B.6}$$

The expected value of each standard error is 1. We can identify possible outliers by their large standard errors. A simple way to treat outliers is to just exclude from the fit any value that has a standard error larger than

a certain threshold. This has the disadvantage that small changes in the values can cause large jumps in the solution if an outlier just crosses the threshold. Instead, we adjust the weights of the data points to reduce the outliers' contribution to the solution:

$$W'_k = \frac{W_k}{1 + \left(\frac{\rho'_k}{\alpha}\right)^\beta} \tag{B.7}$$

The weighting function reduces the weight of values that have residuals α times larger than expected to one half. Of course values with even larger residuals are downweighted even more. The parameter β tunes the "sharpness" of the weighting function.[1] A new estimate of the average is produced by:

$$\widetilde{Z} = \sum_k (y_k - \widetilde{Z}) W'_k \tag{B.8}$$

The residual calculation, weighting and average estimating are iterated until the estimate doesn't change.

Finally, the error for the estimated parameters is calculated. the error of the zero point is:

$$\epsilon^2_{\mathrm{zp}} = \frac{\sum \rho_k^2 W'_k}{\sum W'_k} \tag{B.9}$$

and the *mean error of unit weight* is:

$$\mathrm{me1}^2 = \frac{\sum \rho_k^2 W'_k}{N - 1} \tag{B.10}$$

where N is the number of standard stars. The mean error of unit weight is 1 in the ideal case (when all the errors are estimated correctly). A significantly larger value should raise doubts about the error estimates.

[1] See Peter B. Stetson, *The Techiques of Least Squares and Stellar Photometry with CCDs* at http://nedwww.ipac.caltech.edu/level5/Stetson/Stetson_contents.html.

Appendix C

Native Star Files

The file format used by GCX for star catalogs, recipies and observation reports consists of lists of token–value pairs. Each list is enclosed in brackets. The value part of each pair can be a number, a string (enclosed in double quotes) or another list. The order of the pairs within the list is not important.

A file can contain several top-level lists. Each such list is called a "frame". A typical recipe frame looks like:

```
( recipe ( object "aucyg" ra "20:18:32.76" dec "34:23:21.3"
    equinox 2000 comments "generated from tycho2" )
  sequence "tycho2"
  stars (

    (name "2680-1551" type std mag 7.48
    ra "20:17:54.500" dec "34:05:25.22" perr 0.0071
    comments "p=T "
    smags "v=7.344/0.020 b=8.654/0.020 vt=7.483/0.011 bt=9.024/0.017"
    flags ( astrom ) )

    (name "2680-1588" type std mag 7.79
    ra "20:16:56.755" dec "34:08:22.86" perr 0.0057
    comments "p=T "
    smags "v=7.795/0.020 b=7.745/0.020 vt=7.790/0.011 bt=7.731/0.015"
    flags ( astrom ) )

    (name "aucyg" type target mag 9.5
    ra "20:18:32.76" dec "34:23:21.3"
    comments "p=V t=M s=M6e-M7e m(p)=9.50/15.30"
    flags ( var ) ) )
  )
```

An observation report frame can look like:

```
( observation ( filter "b" object "aucyg" ra "20:18:39.31"
    dec "34:23:05.6" equinox 2000 mjd 53236.9839856
    telescope "SCT" aperture 30 exptime 20 sns_temp 238.3
    latitude "44:25:50.0" longitude "-26:06:50.0"
```

```
    altitude 75 airmass 1.223 observer "R. Corlan" )
  noise ( read 7 eladu 2 flat 0 )
  ap_par ( r1 5 r2 9 r3 13 sky_method "synthetic_mode" )
    (name "AU_CYG_39" type std mag 12.1
    ra "20:17:47.15" dec "34:32:01.4"
    smags "v=12.062/-0.001 b-v=1.982/-0.002 b=14.044/0.002"
    imags "b=-7.256/0.167"
    residual -0.209 stderr 1.25
    noise (photon 0.13 sky 0.074 read 0.075 scint 0.0021)
    centroid (x 876.89 y 385.24 xerr 0.0085 yerr 0.0065 dx -0.18 dy -0.15)
    flags ( astrom centered ) )

    (name "AU_CYG_10" type std mag 13
    ra "20:17:44.52" dec "34:21:00.0"
    smags "v=12.961/-0.001 b-v=0.765/-0.001 b=13.726/0.001"
    imags "b=-7.991/0.089"
    residual 0.208 stderr 2.33
    noise (photon 0.066 sky 0.038 read 0.038 scint 0.0021)
    centroid (x 569.61 y 331.56 xerr 0.002 yerr 0.0017 dx 0.19 dy -0.17)
    flags ( astrom centered ) )

    (name "aucyg" type target mag 9.5
    ra "20:18:32.76" dec "34:23:21.3"
    comments "p=V t=M s=M6e-M7e m(p)=9.50/15.30"
    smags "b=12.642/0.085"
    imags "b=-8.867/0.042"
    centroid (x 782.90 y 615.40 xerr 0.0079 yerr 0.0064 dx 0.17 dy 0.07)
    noise (photon 0.031 sky 0.017 read 0.017 scint 0.0021)
    flags ( var centered ) ) )
  transform ( band "b" zp 21.509 zperr 0.074 zpme1 1.29 ) )
```

A typical catalog frame would be:

```
( catalog (comments "Internal catalog output")
    stars (
      ( name "piuma" type catalog ra "08:39:11.70" dec "65:01:15.0"
        flags ( var ) )
      ( name "lpup" type catalog ra "07:13:31.90" dec "-44:38:39.0"
        flags ( var ) )
      ( name "piori" type catalog ra "04:54:15.10" dec "02:26:26.4"
        flags ( var ) ) ) )
```

All three forms of the star file have in common the top-level token `stars` which is followed by the list of all stars within the frame. Each star is also represented by a list, having some of the following tokens:

name	string	The name of the star;
type	symbol	The type of the star: `std` for standards, `target` for targets, `catalog` far catalog objects and `field` for field stars;
ra	string	Right ascension in HMS format;
dec	string	Declination in DMS format;
perr	number	Position error in arc seconds;

equinox	number	Equinox of coordinates;
mag	number	Generic magnitude of the star, used for display purposes only;
smags	string	A string describing the standard magnitudes of the star and their errors. See Section 7.7 for the format;
imags	string	A string describing the instrumental magnitudes of the star and their errors;
centroid	list	A list containing positional information. `x` and `y` are frame coordinates of the star's centroid. `xerr` amd `yerr` are the estimated position errors calculated by the centroiding code. `dx` and `dy` are the amounts by which the star's centroid is shifted from the catalog position;
noise	list	A list of components of the star's instrumental noise model;
residual	number	The residual of the star after the zeropoint fit;
stderr	number	The standard error (scaled residual) of the star after the zeropoint fit;
flags	list	A list of reduction flags.

Some top-level tokens of the star files are:

stars	list	List of stars in the frame;
recipe	list	A list of recipe parameters. Most important are the target object and field center coordinates;
observation	list	Parameters of the ccd frame from which the data was obtained. See Section 7.5 for more details;
catalog	list	Identifies the frame as a catalog frame. The most important token in this list is `comments`;
sequence	string	The source of the standard magnitudes in the star list;
noise	list	A list of parameters of the CCD noise model used to calculate instrumental errors;
ap_par	list	Parameters used by the aperture photometry routine (aperture sizes and sky estimation method);
transform	list	transformation coefficients, zeropoint and extinction coefficients used to calculate the standard magnitudes of target stars.

The native star format can be converted to a tabular format using the report converter function as described in Appendix E.

Appendix D

Command Line Options

Most of the data reduction functions of GCX are available from the command line. The program's command-line options are listed below. The same text can be obtained using `gcx --help`.

```
gcx version 0.9.8

usage: gcx [options] [<fits file> ...]

                       General options:
-h, --help                          Print command line options
    --help-all                      Print all the on-line help on stdout
    --version                       Print program version
-D, --debug <level>                 Set debug level to <level>; 0=quiet, 4=noisy
-o, --output <file_name>            Specify output file name for import, convert
                                      and frame operations
-r, --rcfile <params_file>          Load parameters file
-S, --set <option>=<value>          Set an option overriding the parameters file
-i, --interactive                   Display frames as they are being processed

       Observation scripting, photometry and file conversion options:
-p, --recipe <recipe_file>          Load recipe file (searches rcp_path)
-P, --phot-run <recipe_file>        Load recipe file and run photometry
                                      in batch mode. Report in native
                                      format.
-V, --phot-run-aavso <recipe_file> Load recipe file and run photometry
                                      in batch mode. Report in AAVSO format.

  If <recipe_file> is set to one of the following three special tokens,
  the recipe will change depeding on information in the frame header:
        _TYHCO_ will create a recipe on-the-fly;
        _OBJECT_ will search the recipe path for a file with the same
                   name as the object in the frame (ending in .rcp);
        _AUTO_ will search for a recipe by object name, and if that
                   is not found, create a tycho one.

    --import <catalog name>         Convert a tabular catalog file to the gcx
                                      Lisp-like format. Reads stdin.
```

```
                                  Current table formats are:
                                  gcvs, gcvs-pos, landolt, henden, sumner
    --merge <recipe_file>         Merge a new recipe file over the one loaded
                                  with the --recipe option. Checks are made
                                  for either positional or name duplicates.
                                  Only stars brighter than mag_limit are
                                  merged.
    --set-target                  Specify a target object to be merged into a
                                  recipe file. It will also set the recipe
                                  object, ra and dec fields.
    --make-tycho-rcp <radius>     Create a recipe file for the object specified
                                  with --object using tycho2 stars in a box
                                  radius arcminutes around the object
    --rcp-to-aavso <recipe_file>  Conevrt a recipe file to the aavso db
                                  (tab-delimited) format
                                  If the file argument is '-', stdin is read
                                  The recipe comment and star comment fields
                                  are interpreted to get some db fields.
                                  see the --help-all for more info
-T, --rep-to-table <report_file>  Convert a report file to tabular format
                                  If an output file name is not specified
                                  (with the '-o' argument), stdout is used
                                  If the file argument is '-', stdin is read
-O, --obsfile <obs_file>          Load/run obs file (searches obs_path)
-n, --to-pnm                      Convert a fits file to 8-bit pnm
                                  If an output file name is not specified
                                  (with the '-o' argument), stdout is used
-j, --object                      Specify a target object (useful for setting
                                  an initial wcs
    --mag-limit                   Set a magnitude limit for the output of
                                  import and merge commands.

                    CCD Reduction Options
-d, --dark <dark_frame>           Set the dark frame / do dark subtraction
-b, --bias <bias_frame>           Set the bias frame / do bias subtraction
-f, --flat <flat_frame>           Set the flat field frame / flatfield
-G, --gaussian-blur <fwhm >       Set blur FWHM / apply Gaussian blur
-a, --align <align_ref_frame>     Set the alignment reference frame
                                     / align frames
-A, --add-bias <bias>             Set a constant bias to add to all frames
                                     / add a bias to frames
-M, --multiply <multiplier>       Set a constant to multiply all frames with
                                     / multiply frames by a scalar
                                  Multiplication is performed before addition.
-u, --update-file                 Save reduction results back to
                                  the original files
-s, --stack                       Stack the frames using the method set in
                                  the parameters file; for some methods
                                  additive background alignment is performed
-F, --superflat                   Stack the frames using an multiplicative
                                  background alignment procedure; the frames
                                  should be dark-substracted
-N, --no-reduce                   Do not run the reduction operations, just
                                  load the frame list / reduction options
```

```
When any of the CCD reduction options is set and the -i flag
is not specified, the reduction operations are run in batch mode
on all the supplied fits files. When no output file is specified
or -i is set, the files are loaded into the batch processing file
list, the reduction options set in the dialog, and the program
starts up in gui mode
```

Appendix E

Report Converter

The output format of GCX can be converted to a more common tabular form using the `-T` option. This option uses the value of the `.file.tab_format` option (*File and Device Options/Report converter output format*) to specify it's output format. The text below, taken from the on-line help, describes this option.

```
             Report Converter Format String

The report converter option converts the native gcx output to a
fixed-width tabular format that is easy to import in other programs
for further processing. The table's format is defined by an option
string. The option string consists of tokens separated by spaces.
There are two types of tokens: option tokens, and column tokens.

Options tokens set global table options when present. They can appear
anywhere in the format string.

tablehead        Generate a table header line containing
                   the column titles
collist          Generate a list of columns with position information
                   at the start of the output


res_stats        Generate a line with descriptive statistics on
                   the stars' residuals at the end of each frame.Column tokens specify what information get
The first column token corresponds to the first output column, and so
on in order. Each column token can optionally be followed by a specifier
of the form: width.precision. The width excludes a single character
spacer between the columns. Supported colum tokens are:

name [w]         Output the star's designator
ra [w]           Output the right ascension in h:m:s format
dra [w.p]        Output the right ascension in decimal degrees format
dec [w]          Output the declination in d:m:s format
ddec [w.p]       Output the declination in decimal degrees format
smag [w.p] "<band>" Output the standard magnitude with the given name
serr [w.p] "<band>" Output the error of the standard magnitude
                    with the given name
```

```
imag [w.p] "<band>" Output the instrumental magnitude with the given name
ierr [w.p] "<band>" Output the error of the instrumental magnitude
                     with the given name
flags [w]        Output reduction flags and the star type
observation [w]  Output the name of the observation (a synthetic name that
                   can be used to group stars reduced from the same frame
airmass [w.p]    Output the airmass of the observation
jdate [w.p]      Output the Julian date of the observation
mjd [w.p]        Output the modified Julian date of the observation
filter [w]       Output the filter name used for the observation
xc/yc [w.p]      Output the frame coordintes of the star's centroid
xerr/yerr [w.p]  Output the estimated centroiding errors
dx/dy [w.p]      Output the amount the star was moved from it's catalog
                   position when the measuring aperture was centered
residual [w.p]   Output the star's residual in the ensemble solution
stderr [w.p]     Output the star's standard error (residual divided by
                   the estimated error

Fields for which data is not available are left blank


When converting recipies to AAVSO database format, some fields are obtained
from the recipe and star comments. Tokens of the form <token>=<value> are
searched for. The value is taken for the database field, with underscores and
minus signs converted to spaces. The following tokens are used:

                        Chart comment tokens:

d                AAVSO (Harward) designation of the main var on the chart
s                Original chart scale
t                Latest chart date (MM/YY, YYYY or YYMMDD)
n                Name of the main variable on the chart
w                Who converted the chart
p                Sequence source

                        Star comment tokens:

p                Position source
c                Comments
n                Name of variable star
i                Id of star on chart (if different from v(aavso))
```

Appendix F

Global Options

GCX has a large number of configuration options. They can be accesses through the options dialog, which can be brought up by pressing **O** or *File/Edit Options*. Clicking on "save" in the options dialog will update the default configuration file (`.gcxrc`), located in the home directory.

When the program stars, it looks for the configuration file. If it cannot be read, it will initialise all parameters with defaults.

There are two ways of modifying some of the options for a specific run: either use the `-S` command-line option for each option that needs to change, or make a configuration file that contains the option changes for the run and use the `-r` command-line option. The name used in the configuration file is listed with each option.

F.1 File and Device Options

Serial line for gps receiver control

Config file	`.file.gps_serial`
Type	`string`
Default	`/dev/ttyS0`

Command for uncompressing files

The command used to uncompress files; it should take the filename as an argument, and output the uncompressed stream on stdout. The command is called when trying to open files with the name ending in .gz, .zip or .z

Config file	`.file.uncompress`
Type	`string`
Default	`zcat`

Command for compressing files

The command used to uncompress files; it should take the filename as an argument, and replace the file with the compressed version.

Config file	`.file.compress`
Type	`string`
Default	`gzip -f`

Gnuplot command

Command used to invoke gnuplot for ploting graphs in a window.

Config file	`.file.gnuplot`
Type	`string`
Default	`gnuplot -persist`

Plot to file

Create a gnuplot file instead of calling gnuplot directly.

Config file	`.file.plot_to_file`
Type	`boolean`
Default	`no`

Monospaced Font

The XFLD for the monospaced font used to display FITS headers and help pages.

Config file	`.file.mono_font`
Type	`string`
Default	`-misc-fixed-medium-r-*-*-*-120-*-*-*-*-*-*`

Phot report file

Output file for obscript phot commands.

Config file	`.file.phot_out_fn`
Type	`string`
Default	`phot.out`

Obs files search path

Path (colon-delimited list of directories) searched when opening observation script files.

Config file	`.file.obs_path`
Type	`string`
Default	`.:../obs`

Rcp files search path

Path (colon-delimited list of directories) searched when opening recipe files.

Config file	`.file.rcp_path`
Type	`string`
Default	`.:../rcp`

GSC location

Location of gsc catalog files. Full pathname of the toplevel gsc directory.Make sure all subdirs and files names use lower case.

Config file	`.file.gsc_path`
Type	`string`
Default	`/usr/share/gcx/catalogs/gsc`

Tycho2 location

Full pathname of the tycho2 catalog file.

Config file	`.file.tycho2_path`
Type	`string`
Default	`/usr/share/gcx/catalogs/tycho2`

Catalog files

Colon-delimited list of catalog files to be searched or loaded into the in-memory catalog. The elements of the list are glob- and tilde-expanded so for instance ~/*.gcx will load all .gcx files from the home directory. The files are expected to be in gcx recipe format.

Config file	`.file.catalog_path`
Type	`string`
Default	`/usr/share/gcx/catalogs/*.gcx`

Preload catalogs

Enable preloading the native format catalog files into memory for faster searching. Preloading takes significantly more time that searching the on-disk files by name; it's recommended only for merging catalogs or similar operations.

Config file	`.file.catalog_preload`
Type	`boolean`
Default	`no`

Edb directory

Directory containing edb object files.

Config file	`.file.edb_dir`
Type	`string`
Default	`/usr/share/gcx/catalogs/edb`

Aavso validation file

Location of aavso validation file. The validation file is used to determine the designation given a star name. Any of the validation file formats can be used, except the 'short form by name'. The designation should start in column 0 and the name in column 10.

Config file	`.file.validation`
Type	`string`
Default	`valnam.txt`

Force unsigned FITS

Force all values read from a fits file to be interpreted as unsigned numbers (supporting broken files generated that way).

Config file	`.file.unsigned_fits`
Type	`boolean`
Default	`no`

Western longitudes

Interpret fits header longitudes as western, rather than eastern. Report file longitudes are always western.

Config file	`.file.west_long`
Type	`boolean`
Default	`yes`

Report converter output format

The output format string used by the report converter. See the on-line help for a description.

Config file `.file.tab_format`
Type `string`
Default `name ra dec mjd smag "v" flags`

Auto-unload frames

Unload unmodified frames from the frame list whenever possible to reduce memory usage. The frames will have to be reloaded if needed again.

Config file `.file.mem_save`
Type `boolean`
Default `yes`

New frame width

The width of a newly-created blank frame in pixels.

Config file `.file.new_width`
Type `integer`
Default `1024`

New frame height

The height of a newly-created blank frame in pixels.

Config file `.file.new_height`
Type `integer`
Default `768`

New frame scale

The default wcs scale of a newly-created blank frame in arc seconds per pixel.

Config file `.file.new_scale`
Type `real`
Default `4.00`

Merge positional tolerance

The distance in arc seconds below which we consider two stars to be the same object when merging recipe files.

Config file `.file.merge_tolerance`
Type `real`
Default `3.000`

F.1.1 Fits Field Names

Fits field for X coordinate of reference pixel

Config file	`.file.fits.crpix1`
Type	`string`
Default	`CRPIX1`

Fits field for Y coordinate of reference pixel

Config file	`.file.fits.crpix2`
Type	`string`
Default	`CRPIX2`

Fits field for WCS R.A. of reference pixel

Config file	`.file.fits.crval1`
Type	`string`
Default	`CRVAL1`

Fits field for WCS Dec of reference pixel

Config file	`.file.fits.crval2`
Type	`string`
Default	`CRVAL2`

Fits field for degrees/pixel in R.A.

Config file	`.file.fits.cdelt1`
Type	`string`
Default	`CDELT1`

Fits field for degrees/pixel in dec

Config file	`.file.fits.cdelt2`
Type	`string`
Default	`CDELT2`

Fits field for field rotation

Config file	`.file.fits.crota1`

Type `string`
Default `CROTA1`

Fits field for equinox of WCS

Config file `.file.fits.equinox`
Type `string`
Default `EQUINOX`

Fits field for equinox of WCS

Config file `.file.fits.epoch`
Type `string`
Default `EPOCH`

Fits field for object name

Config file `.file.fits.object`
Type `string`
Default `OBJECT`

Fits field for object R.A.

Config file `.file.fits.objctra`
Type `string`
Default `OBJCTRA`

Fits field for object dec

Config file `.file.fits.objctdec`
Type `string`
Default `OBJCTDEC`

Fits field for image scale

Config file `.file.fits.secpix`
Type `string`
Default `SECPIX`

Fits field for field center

Config file	`.file.fits.ra`
Type	`string`
Default	`RA`

Fits field for field center

Config file	`.file.fits.dec`
Type	`string`
Default	`DEC`

Fits field for filter name

Config file	`.file.fits.filter`
Type	`string`
Default	`FILTER`

Fits field for exposure time

Config file	`.file.fits.exptime`
Type	`string`
Default	`EXPTIME`

Fits field for julian date of observation

Config file	`.file.fits.jdate`
Type	`string`
Default	`JDATE`

Fits field for modified julian date of observation

Config file	`.file.fits.mjd`
Type	`string`
Default	`MJD`

Fits field for date/time of observation

Config file	`.file.fits.dateobs`
Type	`string`

Default `DATE-OBS`

Fits field for time of observation

Config file `.file.fits.timeobs`
Type `string`
Default `TIME-OBS`

Fits field for telescope name

Config file `.file.fits.telescop`
Type `string`
Default `TELESCOP`

Fits field for focus designation

Config file `.file.fits.focus`
Type `string`
Default `FOCUS`

Fits field for telescope aperture

Config file `.file.fits.aperture`
Type `string`
Default `APERT`

Fits field for focal length

Config file `.file.fits.flen`
Type `string`
Default `FLEN`

Fits field for instrument name

Config file `.file.fits.instrument`
Type `string`
Default `INSTRUME`

Fits field for observer name

Config file `.file.fits.observer`
Type `string`
Default `OBSERVER`

Fits field for latitude of observing site

Config file `.file.fits.latitude`
Type `string`
Default `LAT-OBS`

Fits field for longitude of observing site

Config file `.file.fits.longitude`
Type `string`
Default `LONG-OBS`

Fits field for altitude of observing site

Config file `.file.fits.altitude`
Type `string`
Default `ALT-OBS`

Fits field for airmas

Config file `.file.fits.airmass`
Type `string`
Default `AIRMASS`

Fits field for sensor temperature

Config file `.file.fits.snstemp`
Type `string`
Default `SNS_TEMP`

Fits field for horisontal binning

Config file `.file.fits.binx`
Type `string`
Default `CCDBIN1`

Fits field for vertical binning

Config file	`.file.fits.biny`
Type	`string`
Default	`CCDBIN2`

Fits field for horisontal window origin

Config file	`.file.fits.skipx`
Type	`string`
Default	`CCDSKIP1`

Fits field for vertical window origin

Config file	`.file.fits.skipy`
Type	`string`
Default	`CCDSKIP2`

Fits field for electrons per ADU

Config file	`.file.fits.eladu`
Type	`string`
Default	`ELADU`

Fits field for average bias of frame

Config file	`.file.fits.dcbias`
Type	`string`
Default	`DCBIAS`

Fits field for readout noise

Config file	`.file.fits.rdnoise`
Type	`string`
Default	`RDNOISE`

Fits field for multiplicative noise coefficient

Config file	`.file.fits.flnoise`
Type	`string`

Default `FLNOISE`

F.2 General Observation Setup Data

Latitude of observing site

Used to annotate frames' fits header, and for calculation of objects' hour angle and airmass. Set in decimal degress.

Config file `.obs.latitude`
Type `real`
Default `44.431`

Longitude of observing site

Used to annotate frames' fits header, and for calculation of objects' hour angle and airmass. Set in decimal degress, E longitudes negative.

Config file `.obs.longitude`
Type `real`
Default `-26.114`

Altitude of observing site

Used to annotate frames' fits header, and for scintillation calculations; In meters above sea level.

Config file `.obs.altitude`
Type `real`
Default `75.0`

Observer name

Used to annotate frames' fits header.

Config file `.obs.observer`
Type `string`
Default `R. Corlan`

Observer code

Used to annotate AAVSO format reports.

Config file `.obs.obscode`
Type `string`
Default `CXR`

Telescope name

Used to annotate frames' fits header.

Config file	`.obs.telescope`
Type	`string`
Default	`SCT`

Telescope focal length

Used to annotate frames' fits header, and for calculating the initial WCS image scale (together with the camera pixel size). In centimeters.

Config file	`.obs.flen`
Type	`real`
Default	`200.0`

Flipped field

Set the inital wcs as flipped (CDELT1 and CDELT2 having opposite signs). The star matching algorithm does not handle flips, so this has to be set right. A normal (non-flipped) field has N up and W to the right.

Config file	`.obs.flipped`
Type	`boolean`
Default	`no`

Telescope aperture

Used to annotate frames' fits header, and for estimating scintillation noise. In centimeters.

Config file	`.obs.aperture`
Type	`real`
Default	`30.0`

Focus designation

Used to annotate frames' fits header.

Config file	`.obs.focus`
Type	`string`
Default	`Cassergrain`

List of filters

Space-delimited list of filters used to set the choices in the obscript dialog. They are also used as default names for the filters in filter wheels that don't supply their own names

Config file	`.obs.filters`
Type	`string`
Default	`v r b i RI GI BI clear`

F.3 Telescope control options

Serial line for telescope control

Config file	`.tel.scope_serial`
Type	`string`
Default	`/dev/ttyS0`

Serial line for filter wheel control

Config file	`.tel.fwheel_serial`
Type	`string`
Default	`/dev/ttyUSB0`

Eastern limit for goto ops (hour angle)

Config file	`.tel.elimit`
Type	`real`
Default	`-1.0`

Enable eastern limit

Config file	`.tel.elimit_en`
Type	`boolean`
Default	`no`

Western limit for goto ops (hour angle)

Config file	`.tel.wlimit`
Type	`real`
Default	`1.0`

Enable western limit

Config file `.tel.wlimit_en`
Type `boolean`
Default `no`

Southern limit for goto ops (degrees)

Config file `.tel.slimit`
Type `real`
Default `-45.0`

Enable southern limit

Config file `.tel.slimit_en`
Type `boolean`
Default `yes`

Northern limit for goto ops (degrees)

Config file `.tel.nlimit`
Type `real`
Default `80.0`

Enable northern limit

Config file `.tel.nlimit_en`
Type `boolean`
Default `yes`

Abort obscript gotos that cause a meridian flip

Config file `.tel.abort_flip`
Type `boolean`
Default `yes`

Mount speed when centering (times sidereal rate)

Config file `.tel.centering_speed`
Type `real`

Default `32.0`

Mount speed when guiding (times sidereal rate)

Config file `.tel.centering_speed`
Type `real`
Default `0.5`

Use timed centering moves for small slews

Config file `.tel.use_centering`
Type `boolean`
Default `yes`

Precess coordinated sent to the scope to the epoch of the day

Config file `.tel.precess_eod`
Type `boolean`
Default `yes`

Amount of gear play we take out at the end of slews (degrees)

Config file `.tel.gear_play`
Type `real`
Default `0.10`

Mount stabilisation delay (ms)

Config file `.tel.stabilisation_delay`
Type `integer`
Default `1000`

Amount of pointing dither between frames (arcmin)

Config file `.tel.dither`
Type `real`
Default `0.5`

Enable pointing dithering

Config file	`.tel.dither_en`
Type	`boolean`
Default	`no`

Max error we accept in telescope pointing (degrees)

Config file	`.tel.max_point_err`
Type	`real`
Default	`0.05`

F.4 Guiding options

Guiding algorithm

Algorithm used to determine the guide star position error. The reticle algorithm determines the error as the ratio between the total flux in box-shaped areas located horisontally and vertically around the target position; The ratioed reticle tries to maintain the initial flux ratio, rather than a unit ratio.

Config file	`.guide.algorithm`
Type	`multiple choice`
Choices	`centroid` `reticle` `ratioed_reticle`
Default	`centroid`

Reticle size

Size of the reticle in pixels. Must be an even value.

Config file	`.guide.reticle_size`
Type	`integer`
Default	`2`

Centroid area radius

Radius of area inside which the target centroid is calculated.

Config file	`.guide.centroid_box`
Type	`integer`
Default	`2`

Guide box size

Size of box around the target guide star position used by the guiding algorithm.

Config file	`.guide.box_size`
Type	`integer`
Default	`16`

Guide box zoom

Zoom level of the small image of the guide box.

Config file	`.guide.zoom`
Type	`integer`
Default	`4`

F.5 CCD Reduction options

Minimum backgound sigmas

Minimum number of sigmas the background must have for multiplicative background alignment to be performed when stacking frames.

Config file	`.ccdred.min_bg_sigmas`
Type	`real`
Default	`3.0`

Method used for stacking frames

Config file	`.ccdred.stack_method`
Type	`multiple choice`
Choices	`average median kappa_sigma mean_median`
Default	`kappa_sigma`

Iteration limit for stacking operations

Config file	`.ccdred.iterations`
Type	`integer`
Default	`4`

Clipping sigmas for mmedian and k-s

Config file	`.ccdred.sigmas`

Type `real`
Default `2.0`

Run CCD reductions automatically on frame display

Config file `.ccdred.auto`
Type `boolean`
Default `yes`

F.6 Star Detection and Search Options

Star Detection SNR

A star must be above this treshold above the backround to be considered for detection. Expressed in sigmas of the image. Useful range between 3 and 24.

Config file `.stars.det_snr`
Type `real`
Default `9.0`

Maximum Detected Stars

The maximum number of starts the program will extract from an image. If more are found, only the brightest are kept.

Config file `.stars.det_maxstars`
Type `integer`
Default `200`

Maximum Catalog Stars

The maximum number of starts the program will extract from a field star catalog. If more are found, only the brightest are kept.

Config file `.stars.cat_maxstars`
Type `integer`
Default `300`

Faintest Magnitude for Catalog Stars

Config file `.stars.cat_maxmag`
Type `real`
Default `16.0`

Maximum Radius for Catalog Stars

The maximum radius from the frame center in which we look for catalog stars. The actual radius depends on the frame size; this is a global limit. In minutes of arc.

Config file	`.stars.cat_maxradius`
Type	`real`
Default	`120.0`

F.7 Wcs Fitting Options

Scale tolerance

The maximum amount by which the image scale is changed during the fit. A value of 0.1 implies that scales between 0.9 and 1.1 are accepted. Using a lower value for this parameter will decrease the running time and reduce the chances for an incorrect fit.

Config file	`.wcs.fit_scale`
Type	`real`
Default	`0.10`

Star position tolerance

The maxiumum position error, in pixels, between a star and a catalog position at which we still consider that the two match. It is unadvisable to increase this parameter much without a coresponding increase in the minimum number of pairs, or the chances for incorrect fits are greatly augumented.

Config file	`.wcs.fit_tol`
Type	`real`
Default	`1.5`

Rotation tolerance

Maximum accepted field rotation, in degress A value of 180 will match fields with any rotation. Using a lower value for this parameter will decrease the running time and reduce the chances for an incorrect fit.

Config file	`.wcs.fit_rot`
Type	`real`
Default	`180.0`

Minimum number of pairs

When the algorithm finds a solution with at least this many pairs, it considers it has found a good match and stops. Using less than 5 pairs will likely give many incorrect fits in crowded fields

Config file	`.wcs.fit_min_pairs`
Type	`integer`
Default	`5`

Max skip

The maximum number of stars that don't seem to have a match in the catalog and are skipped during the search

Config file	`.wcs.max_skip`
Type	`integer`
Default	`20`

Maximum pairs

Maximum number of pairs fitted. After finding this many, the algorithm stops.

Config file	`.wcs.max_pairs`
Type	`integer`
Default	`50`

Min A-B distance

The minimum distance between two stars to be used as an initial wcs hypothesis.

Config file	`.wcs.min_ab_dist`
Type	`real`
Default	`100.000`

Max error for WCS validation

Maximum RMS position error of all pairs at which the WCS is validated. Expressed in pixels.

Config file	`.wcs.validate_err`
Type	`real`
Default	`1.50`

Min pairs for WCS validation

The minimum number of pairs needed to validate a WCS solution.

Config file	`.wcs.validate_pairs`
Type	`integer`
Default	`3`

Scale change per iteration

Maximum scale change accepted when iterating the WCS solution. This is an internal parameter few people (if any) will want to change.

Config file	`.wcs.max_s_chg`
Type	`real`
Default	`0.0200`

Default image scale

Default image scale used for setting the initial WCS when this information is not available from the frame header. In arc seconds per pixel

Config file	`.wcs.default_scale`
Type	`real`
Default	`1.50`

Plate Model

Model used for plate to projection plane transformations. Simple consists of translation, scale and rotation; Linear is a general affine transformation, which includes shear; Auto selects a plate model based on the number of star pairs.

Config file	`.wcs.plate_model`
Type	`multiple choice`
Choices	`auto simple linear`
Default	`auto`

Calculate Refraction

Enable the calculation of refracted positions of stars. If the time and/or geographical coordinates aren't precise, this option is better turned off.

Config file	`.wcs.refraction`
Type	`boolean`
Default	`no`

F.8 Aperture Photometry Options

Radius of central aperture

Radius, in pixels, of the central aperture from which the star flux is measured.

Config file	`.aphot.r1`
Type	`real`
Default	`4.0`

Inner radius of sky annulus

Inner radius, in pixels, of the annulus surrounding the star from which the sky flux is estimated.

Config file	`.aphot.r2`
Type	`real`
Default	`9.0`

Outer radius of sky annulus

Outer radius, in pixels, of the annulus surrounding the star from which the sky flux is estimated.

Config file	`.aphot.r3`
Type	`real`
Default	`18.0`

Center apertures

Try to center the measuring aperture on the star before performing the measurement.

Config file	`.aphot.auto_center`
Type	`boolean`
Default	`yes`

Max centering error

The maximum amount a measuring aperture is moved when auto-centering (in pixels).

Config file	`.aphot.center_err`
Type	`real`
Default	`1.00`

Discard unlocated standards

When this option is set, standard stars than cannot be located within the specified error radius when centering apertures are not used in the solution.

Config file	`.aphot.discard_unlocated`
Type	`boolean`
Default	`yes`

Move targets

Update the world coordinates of stars that don't have the Astrometric flag set when their apertures are centered. Useful for moving targets like asteroids or comets or instances where the coordinates of the target are only approximately known.

Config file	`.aphot.move_targets`
Type	`boolean`
Default	`no`

Max center distance

The maximum distance from the frame center at which a standard star must lie in order to be used in the photometry solution. The distance is expressed relative to half the frame's diagonal. A value of 1 will include all stars. This option is useful when we expect large flat-fielding errors or distorted star images at the frame's corners.

Config file	`.aphot.std_max_distance`
Type	`real`
Default	`1.00`

Sky method

Method used for sky estimation: Average uses the average of the sky annulus pixels; it's not recommended, as it will not reject nearby stars. Median uses the median of the sky pixels; it's fast and robust, but will not average out quantisation noise. Kappa-Sigma computes the sky value by clipping off pixels too far away from the median, and averaging the rest; the standard deviation of the remaining pixels is calculated, and the clipping/averaging is iterated until values converge; Synthetic mode is calaulated as 3*median - 2*mean, where median and mean are the statistics of the clipped sky annulus.

Config file	`.aphot.sky_method`
Type	`multiple choice`

Choices	`average median kappa_sigma synthetic_mode`
Default	`kappa_sigma`

Region growing

Amount, in pixels, by which regions containing peaks are grown in order to exclude the tails of spurious stars in the sky aperture. A value larger than 3 is silently clamped at 3. Region growing only applies to the kappa-sigma and synthetic mode sky methods.

Config file	`.aphot.grow`
Type	`integer`
Default	`2`

Sigmas

Rejection limit of the mean-median, kappa-sigma and synthetic mode sky estimation methods, in sigmas.

Config file	`.aphot.sigmas`
Type	`real`
Default	`3.0`

Saturation limit

The value over which we mark the stars as being "bright", i.e. possibly saturated.

Config file	`.aphot.sat_limit`
Type	`real`
Default	`48000.000`

Instrumental band

Name of band the stars are measured into. Used when the frame being reduced does not specify a filter or the Force iband option is set.

Config file	`.aphot.iband`
Type	`string`
Default	`v`

Force iband

Override the filter specification from the frame header with the one set in the Instrumental band parameter.

Config file	`.aphot.force_iband`
Type	`boolean`
Default	`no`

Default catalog error

Default standard magnitude error used when a value is not available.

Config file	`.aphot.def_std_err`
Type	`real`
Default	`0.050`

Alpha

Parameter of the robust fitting alogorithm. It sets the fwhm of the rejection function (in standard deviations).

Config file	`.aphot.alpha`
Type	`real`
Default	`2.000`

Beta

Parameter of the robust fitting alogorithm. It sets the sharpness of the rejection function, with higher values resulting in a sharper falloff.

Config file	`.aphot.beta`
Type	`real`
Default	`2.000`

F.9 Multi-Frame Photometry Options

Bands setup

A string which specifies the color indices against which we reduce various bands. It consists of a space-separated list of specifiers of the form: band(b1-b2)[=k/kerr]. In this case, "band" will be reduced against the b1-b2 color index. The optional k and kerr set the default transformation coefficient and it's error for the specified band, like for example v(b-v)=0.06/0.002.

Config file	`.mframe.bands`
Type	`string`
Default	`b(b-v) v(b-v) r(v-r) i(v-i)`

Outlier threshold

Threshold in standard error over which we flag stars as "outliers" for easy identification. Outliers are not excluded from the fit; they are down-weighted by a factor depending on the values of the alpha and beta robust fit parameters.

Config file	`.mframe.outlier`
Type	`real`
Default	`3.000`

Zeropoint outlier threshold

Threshold in standard error over which an image frame is marked as an outlier of the extinction fit. The existance of an outlier frame usually shows that transparency has had a significant change around that time. As a result, calculation of all-sky zeropoints is inhibited in the vicinity of an outlier frame, until a well-fitting frame is encountered.

Config file	`.mframe.zp_outlier`
Type	`real`
Default	`2.000`

Minimum airmass variance

The minimum variance of a data set's airmass for which we try to fit an extinction coefficient. For datasets with lower variance, the extinction coefficient is not considered - which doesn't affect the result, as all frames are taken at nearly the same airmass.

Config file	`.mframe.min_am_variance`
Type	`real`
Default	`0.0010`

Minimum color variance

The minimum variance of a data set's color index for which we try to fit a transformation coefficient. For datasets with lower variance, the transformation coefficient is not fitted, as the possibility of obtaining a meaningless value is quite real.

Config file	`.mframe.min_color_variance`
Type	`real`
Default	`0.1000`

Airmass range

Relative size of the range of airmasses over which we calculate frame zeropoints when doing all-sky reduction. It is expressed relative to the airmass range of the valid standard frames. For example, if the standard frames are taken at airmasses between 1.5 and 2.0, an airmass range of 2.0 will enable all-sky zeropoint calculations for frames with airmasses between 1.25 and 2.25.

Config file	`.mframe.airmass_range`
Type	`real`
Default	`2.000`

Airmass maximum range

The absolute maximum amount we add to or subtract from the standard frames' airmass range when calculating the range in which frames which are reduced all-sky must lie.

Config file	`.mframe.airmass_max_range`
Type	`real`
Default	`0.500`

Limiting magnitude from zeropoint

The amount we subtract from the zeropoint of a frame in order to obtain the limiting magnitude used for reporting purposes.

Config file	`.mframe.lmag_from_zp`
Type	`real`
Default	`7.000`

F.10 Star Display Options

Label Target Stars

Show name labels for target stars

Config file	`.display.target_labels`
Type	`boolean`
Default	`yes`

Label Catalog Objects

Show name labels for catalog objects

Config file `.display.cat_labels`
Type `boolean`
Default `no`

Label Standard Stars

Show magitude labels (with the decimal point omitted) for standard stars

Config file `.display.std_labels`
Type `boolean`
Default `yes`

Standard Star Label Band

The band from which the magnitude used for standard star labeling is taken

Config file `.display.std_label_band`
Type `string`
Default `v`

Minimum star size

The minimum radius of a star symbol on screen, in pixels

Config file `.display.min_sz`
Type `integer`
Default `1`

Maximum display star size

The maximum radius of a star symbol on screen, in pixels.

Config file `.display.max_sz`
Type `integer`
Default `18`

Default display star size

The default radius of a star symbol on screen, used when no magnitude information is known; in pixels.

Config file `.display.default_sz`
Type `integer`
Default `6`

Display pixels per magnitude

The amount the star symbols increase in radius for an increase in brightness of 1 magnitude; in pixels

Config file	`.display.pix_per_mag`
Type	`real`
Default	`1.00`

Brightest symbol magnitude

The magnitude at which we stop increasing star symbol sizes

Config file	`.display.maxmag`
Type	`real`
Default	`5.0`

Display limiting magnitude

Magnitude of the faintest displayed objects

Config file	`.display.dlimmag`
Type	`real`
Default	`17.0`

Show fainter stars

Show stars fainter than the display limiting magnitude (at minimum size)

Config file	`.display.dlimfainter`
Type	`boolean`
Default	`no`

Zoom star shapes

Scale star shapes when the image is zoomed. Regardless of the value of this parameter, aphot shapes are always zoomed, while blob symbols never are.

Config file	`.display.zoom`
Type	`boolean`
Default	`yes`

Star shapes zoom limit

The maximum scale factor applied to star shapes. This limit does not apply to aphot shapes.

Config file	`.display.zoom_limit`
Type	`integer`
Default	`4`

Plot position errors

Draw the amount the apertures were moved from their catalog position when stars are centered.

Config file	`.display.plot_err`
Type	`boolean`
Default	`yes`

Plot error scale

The amount by which the position errors are multiplied when the position error plot is generated.

Config file	`.display.plot_err_scale`
Type	`real`
Default	`100.0`

F.10.1 Star Display Colors

Autodected Stars Color

Config file	`.display.color.simple`
Type	`multiple choice`
Choices	`red orange yellow green cyan blue light_blue gray white`
Default	`green`

Field Stars Color

Config file	`.display.color.field`
Type	`multiple choice`
Choices	`red orange yellow green cyan blue light_blue gray white`
Default	`red`

Standard Stars Color

Config file	`.display.color.apstd`
Type	`multiple choice`

Choices `red orange yellow green cyan blue light_blue gray white`
Default `red`

Target Stars Color

Config file `.display.color.target`
Type `multiple choice`
Choices `red orange yellow green cyan blue light_blue gray white`
Default `red`

Catalog Objects Color

Config file `.display.color.catalog`
Type `multiple choice`
Choices `red orange yellow green cyan blue light_blue gray white`
Default `yellow`

User-Marked Stars Color

Config file `.display.color.usel`
Type `multiple choice`
Choices `red orange yellow green cyan blue light_blue gray white`
Default `green`

Align Stars Color

Config file `.display.color.align`
Type `multiple choice`
Choices `red orange yellow green cyan blue light_blue gray white`
Default `orange`

Selected Stars Color

Config file `.display.color.selected`
Type `multiple choice`

Choices	`red orange yellow green cyan blue light_blue gray white`
Default	`white`

F.10.2 Star Display Shapes

Autodected Stars Shape

Config file	`.display.shape.simple`
Type	`multiple choice`
Choices	`circle square blob aphot diamond cross`
Default	`square`

Field Stars Shape

Config file	`.display.shape.field`
Type	`multiple choice`
Choices	`circle square blob aphot diamond cross`
Default	`diamond`

Standard Stars Shape

Config file	`.display.shape.apstd`
Type	`multiple choice`
Choices	`circle square blob aphot diamond cross`
Default	`aphot`

Target Stars Shape

Config file	`.display.shape.target`
Type	`multiple choice`
Choices	`circle square blob aphot diamond cross`
Default	`cross`

Catalog Objects Shape

Config file	`.display.shape.catalog`
Type	`multiple choice`
Choices	`circle square blob aphot diamond cross`
Default	`diamond`

User-Marked Stars Shape

Config file	`.display.shape.usel`
Type	`multiple choice`
Choices	`circle square blob aphot diamond cross`
Default	`circle`

Align Stars Shape

Config file	`.display.shape.align`
Type	`multiple choice`
Choices	`circle square blob aphot diamond cross`
Default	`circle`

F.11 Synthetic Star Generation Options

Type of star profile

Profile function used to generate synthetic stars.

Config file	`.synth.profile`
Type	`multiple choice`
Choices	`gaussian moffat`
Default	`gaussian`

FWHM

The FWHM of synthetically-generated stars in pixels.

Config file	`.synth.fwhm`
Type	`real`
Default	`2.50`

Moffat beta

The beta parameter of the Moffat function.

Config file	`.synth.beta`
Type	`real`
Default	`4.00`

Zero Point

Zero point (magnitude of a star with a total flux of 1) used to scale synthetic stars.

Config file	`.synth.zp`
Type	`real`
Default	`23.00`

Oversampling

Oversampling factor when generating the reference psf.

Config file	`.synth.oversample`
Type	`integer`
Default	`11`

F.12 On-line Catalogs

Vizquery command

Config file	`.query.vizquery`
Type	`string`
Default	`vizquery`

Maximum Catalog Stars

The maximum number of starts the program will download from a field star catalog.

Config file	`.query.maxstars`
Type	`integer`
Default	`5000`

Maximum Radius for Catalog Stars

The maximum radius from the frame center in which we look for catalog stars. The actual radius depends on the frame size; this is a global limit. In minutes of arc.

Config file	`.query.maxradius`
Type	`real`
Default	`20.0`