A User's Guide to CCD Reductions with IRAF

Philip Massey

February 5, 1989

Abstract

This document is intended to guide you through the basic stages of reducing your CCD data with IRAF, be it spectroscopic or direct imaging. It will take you through the removal of the "instrumental signature", after which you should be ready to extract your spectra or to do your photometry.

Contents

1 Introduction

This document is intended to guide you through the basic stages of reducing your CCD data with IRAF, be it spectroscopic or direct imaging. It will take you through the removal of the "instrumental signature", after which you should be ready to extract your spectra or to do your photometry. It assumes that you are using IRAF Version 2.7 or later.

If you are a brand-new IRAF user I strongly recommend first reading the document "A User's Introduction to the IRAF Command Language" by Shames and Tody, which can be found in Volume 1A of the 4 blue binders that compose the IRAF documentation.

2 How Many and What Calibration Frames Do You Need?

The answer to this depends to some extent on what it is that you are doing. The goal is to not let the quality of the calibration data degrade your signal-to-noise in any way. If you are in the regime where the read-noise of the chip is the dominant source of noise on your program objects, then subtracting a single "bias frame" from your data would increase the noise by $\sqrt{2}$. However, if you instead use the average of 25 bias frames, the noise will be increased by only 10%. Of course, if you are into high signal-to-noise spectroscopy, so you have lots and lots of signal compared to read-noise, or if you have high sky background on direct images, so that read-noise is again immaterial, then the signal-to-noise will be little affected by whether you have only a few bias frames. However, in this regime the quality of your flat fielding is all important if you want to get the most out of your data.

The following list contains the type of calibration images you may need, and provides some guide to the consideration of how many you may want to have.

bias frames. These are zero second integration exposures obtained with the same pre-flash (if any) you are using on your program objects. If read-noise will sometimes dominate your source of error on the program objects, take 25 *bias frames* per night. Take them over dinner and you'll never notice it. You will want to have a new sequence of these each day.

dark frames. These are long exposures taken with the shutter closed. If your longest exposure time is over 15 minutes you may want to take an equal length *dark frame*, subtract a bias frame from it, and decide if you are worried about how much dark current is left. Few of the Kitt Peak or Tololo chips suffer from significant dark current, but it won't hurt you to check once or twice during your run. Mostly I take them but never use them. Applications where dark current *will* matter are long-slit spectroscopy and surface brightness studies — cases where the background is not removed locally. If you do find that you need to take care of dark current, then you should take at least 3 and preferably 5 to 10 dark frames during your run, each with an integration time equal to your longest exposure. You had better make sure that your system is sufficiently light-tight to permit these to be done during the day—if not, hope for a few cloudy nights!

flat field exposures. At a minimum, flat field exposures are used to remove pixel-to-pixel variations across the chip. Usually *dome flats* (exposures of an illuminated white spot) or *projector flats* (exposures of a quartz lamp illuminating the spectrograph slit) will suffice to remove the pixel-to-pixel stuff. You will want to expose the dome or projector flats so that you get sufficient counts to not degrade the signal-to-noise of the final images. If you are after 1% photometry per pixel then you will need to have several times more than 10,000 electrons accumulated in your flats, but you need to be careful not to exceed the good linearity limit in any single flat exposure. Generally if you have 5 or more flats each with 10,000 electrons per pixel you are probably fine. You will need a set like this for every filter or every grating tilt, and you probably will want to do a new sequence every day.

twilight flats. If you are interested in good photometry of objects across your field, or in long-slit spectroscopic work, you need to know if the sky looks different to your CCD than the projector lamp or dome flat. It is not unusual to find 5-10% gradients in the illumination response between a dome flat and a sky exposure, and this difference will translate directly into a 5-10% error in your photometry. For most applications, exposures of bright twilight sky (either for direct imaging or spectroscopy) will cure this problem. With direct imaging this requires you to be very quick on your feet to obtain a good level of sky exposure in each of your filters while the sky is getting darker and darker. (Only the truly desperate would take twilight flats in the morning!) For direct imaging I would recommend 3 to 5 exposures in each filter, stepping the telescope slightly in between the exposures so that any faint stars can be effectively cleaned out. For long-slit spectroscopy take a few exposures of the the twilight sky, stepping the telescope perpendicular to the slit orientation. In both cases you should make sure that tracking is on and that the dome is enabled! You will find that you need to keep increasing the exposure time to maintain an illumination level of $\approx$ 10,000 electrons.

blank dark sky exposures. Some observers doing sky-limited direct imaging may wish to try exposures of blank sky fields rather than twilight sky, as the color of twilight and the color the dark sky do differ considerably. Obtain at least three, and preferably four, long exposures through each filter of some region relatively free from stars ("blank sky" coordinates can be found at Kitt Peak and Tololo), stepping the telescope 10-15 arcseconds between each exposure. The trick here, of course, is to get enough counts in the sky exposures to make this worth your while. Unless you are willing to devote a great deal of telescope time to this, you will have to smooth these *blank dark sky* exposures to reduce noise, but the assumption in such a smoothing process is that the color response of the chip does not vary over the area you are smoothing. You might try dividing a U dome flat by a V dome flat and seeing how reasonable an assumption this might be. Also, if the cosmetics are very bad the smoothing process will wreak havoc with your data if you are not successful in cleaning out bad columns and pixels.

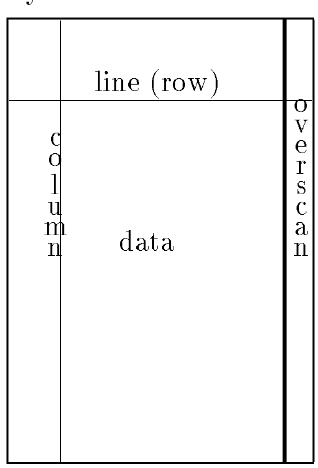
fringe frames. Some CCD's produce an interference fringe pattern when they are illuminated by monochromatic light. For spectroscopy or narrow-band imaging this won't matter, as the fringe pattern will come out nicely with the dome flats, but if you are doing deep exposures in V, R, or I with an RCA chip then the night sky lines may cause a fringe pattern. The only CCD data I've had to defringe was that taken with the Tololo prime focus RCA chip, now honorably retired. Even here the fringes seldom had amplitudes greater than a few percent of the night sky. If you are after equally faint objects of small spatial scale, then you may find yourself having to defringe. For this you will need very, very long blank sky frames obtained as above, but you will not be able to smooth them without destroying the fringe information. Prevention is the best cure for fringes: avoid using an RCA chip, avoid making long VRI exposures within an hour of twilight (as the night sky lines are strongest then), and avoid letting your objects fall on the part of the chip where the fringing is the most servere.

3 Why Your Data Needs Work And What We Intend to Do About It

This section will briefly outline what we will do with the calibration images.

Most of the calibration data is intended to remove "additive" effects: the electronic pedestal level (measured from the overscan region on each of your frames), the pre-flash level (measured from your bias frames), and, if necessary, the dark current. The flat-field data (dome or projector flats and twilight sky exposures) will remove the multiplicative gain and illumination variations across the chip. Fringes are an additive effect that must be removed last.

When you obtained your frames at the telescope, the output signal was "biased" by adding a pedestal level of several hundred ADU's. We need to determine this bias level for each frame individually, as it is not stabilized, and will vary slightly ($\approx$ 5-30 ADU's) with telescope position, temperature, and who knows what else. Furthermore, the bias level is usually a slight function of position on the chip, varying primarily along columns. We can remove this bias level to first-order by using the data in the *overscan region*, the (typically) 32 columns at the right edge of your frames.



We will average the data over all the columns in the overscan region, and fit these values as a function of line-number (i.e., average in the "x" direction within the overscan region, and fit these as a function of "y"). This fit will be subtracted from each column in your frame; this "fit" may be a simple constant. At this point we will chop off the overscan region, and keep only the part of the image containing useful data. This latter step usually trims off not only the overscan region but the first and last few rows and columns of your data.

If you pre-flashed the chip with light before each exposure, there will still be a non-zero amount of counts that have been superimposed on each image. This extra signal is also an additive amount, and needs to be subtracted from your data. In addition, there may be column-to-column variation in the structure of the bias level, and this would not have been removed by the above procedure. To remove both the pre-flash (if any) and the residual variation in the bias level (if any) we will make use of frames that you have obtained with a zero integration time. These are referred to in IRAF as "zero frames" but are called "bias frames" in KPNO and CTIO lingo. We need to average many of these (taken with pre-flash if you were using pre-flash on your object frames), process the average as described above, and subtract this frame from all the other frames.

"Dark current" is also additive. On some CCD's there is a non-negligible amount of background added during long exposures. If necessary, you can remove the dark current to first-order by taking " dark" exposures (long integrations with the shutter closed), processing these frames as above, and then scaling to the exposure time of your program frames. However, it's been my experience that the dark current seldom scales linearly, so you need to be careful. Furthermore, you will need at least 3 dark frames in order to remove radiation events ("cosmic rays"), and unless you have a vast number of dark exposures to average you may decrease your signal-to-noise; see the discussion above in Section 2. The bottom line of all this is that unless you really need to remove the dark current, don't bother.

The next step in removing the instrumental signature is to flat-field your data. This will remove the pixel-to-pixel gain variations, and (in the case of long-slit spectroscopy and direct imaging) the larger-scale spatial variations. If you are doing direct imaging, or plan to flux calibrate your spectroscopic data, then you are probably happy by normalizing the flat-field exposures to some average value, but if you are interested in preserving counts for the purposes of statistics in spectroscopic data we may want to first fit a function to remove the color of the lamp. The final step in the flat-fielding process is to see if your twilight sky exposures have been well flattened by this procedure—if not, we may have to correct for this.

Finally, if you have broadband direct images and absolutely must remove fringes, then this will be your last step, but it will be a long and lonely one—IRAF currently doesn't provide much help. The fringe pattern is an additive effect, and must be subtracted from each program frame that is affected. This means that you will first have to construct master "fringe frames" for each filter you are planning to defringe, and then you are going to have to determine a scaling factor to multiply the appropriate fringe frame by in order to properly reproduce the amplitude of the fringes for each and every affected program frame.

4 How to Do It

Throughout this section I will assume that you know how to examine the "hidden parameters" in an IRAF task, how to change these parameters, and how to execute the task. If you don't know this already, read the Shames and Tody guide mentioned above, *and* sit down with someone who knows all this stuff and have him or her give you a crash course. Everyone seems to have their own style of doing these things: I always like to start out by invoking the task parameter editor **epar** *taskname*, setting all the parameters, and exiting with a **:go** in order to make darn sure that I have everything set the way I thought I did.

At the end of this guide you will find a complete example of a common reduction procedure: reducing direct imaging. Throughout the following discussion I will draw on this example as well as illustrating the steps with the case of reducing spectroscopic data.

4.1 Reading in your data.

The first step is to transfer your CCD frames into "IRAF images". These will consist of IRAF "header files" with the extension ".imh", and IRAF "pixel files" with the extension ".pix". The later will be found in your image directory (**show imdir** will show you where this is), while the headers will reside in whatever directory you happen to be in when you create the pictures. The header files contain the information about where the pixel data resides on disk, and when you refer to an image by its name without the extension, the ".imh" is assumed. Most of this is supposed to be "invisible to the user", but it won't hurt you to know what is going on.

4.1.1 Transferring Data from the CCD Computer

If you are on the mountain and your data resides on one of the CCD systems you can transfer the data directly over to the SUN. However, to avoid confusion, I strongly recommend that you keep each night's data in its own subdirectory. Therefore I would begin by making a new subdirectory **mkdir nite1** and then moving over to that directory by doing a **cd nite1**.

A sequence of picture numbers **n1** through **n2** can be transfered by typing **getpix n1-n2**. Note that this has to be done in the IRAF window or the pictures will wind up somewhere else! (**getpix** is not an IRAF command but an alias for a Unix routine.) If the pictures you want to transfer are not all consecutive numbers then you can instead type **getpix 1-15 203 301-305 210-215 898** or somesuch. Note that the sets are not separated by commas.

When the data comes across you can examine it in the usual way; i.e., **dir *.imh** or **imhead *.imh**. You will find that all the "bias" frames are labeled "biasXXX.imh", where XXX is the original CCD picture number. Similarly there may be names like "pflatXXX.imh", "dflatXXX.imh", and, hopefully, "objXXX.imh".

4.1.2 Transferring Data From Tape

The first step in reading in your tape is to find an available tape drive, and figure out what its IRAF name is. Usually the names are something like **mta**, **mtb**, etc., but if you are on a

Figure 1: Parameters for **rfits**

network you may also need to refer to the machine that the tape drive is attached to, such as **orion!mta**, **tucana!mta**, and so on. To allocate the tape drive type **all tucana!mta** or somesuch. Next, put the tape on (you might want to remove the write-ring first!) and fiddle with the tape drive until the tape is on-line and at the load-point. If your tape is in FITS format we will use **rfits** in the **dataio** package, so type **dataio** and then **lpar rfits** to see what the parameters are. Modify the parameters using **epar rfits** until they resemble those shown in Figure 1. This will read in all the files on your tape and give them names **nite1001, nite1002, nite1003, nite1004....**. If you have data from more than one night on tape (e.g., your calibration frames may be different) then you might consider reading the tape in pieces and giving the second night's data names like **nite2....** A few seconds of thought at this stage may save you some hassle later on. If data from one night takes up more than one tape, then you can **rewind tucana!mta** when you are done, exchange the second tape for the first (note that this way you don't run the risk of losing your tape drive!), and then execute **rfits** again this time inserting the current picture number for the "offset" parameter. When you are done, do a **deallocate tucana!mta** to get rid of the tape drive, and a **bye** to get rid of the "dataio" package. If your tape is in "CAMERA" format rather than "FITS" format, then you will need to load **noao** and **mtlocal** and use **rcamera**; the syntax is similar to that of **rfits**.

4.2 Setting things up: "setinstrument" and "ccdlist"

When we actually reduce our CCD data we will use the "ccdproc" task within the "ccdred" package, and this task will depend upon the image headers to get everything right. So our first operation has to be to set up the translation table between the header information and things that "ccdred" will want to know, such as what a "zero frame" is called, what distinguishes a U filter flat-field from a R filter flat-field, and so on. The command for setting up this translation file is **setinstrument** in the **ccdred** package. So load **noao**, **imred**, and **ccdred**, and then type **setinstrument**. If you type a **?** you will get the list shown in Figure 2. In this case we

Figure 2: Possible answers to **setinstrument**

are trying to reduce spectroscopic data taken with one of the GEC chips on the CTIO 1.5-m spectrograph, and for lack of anything better to answer we will answer **setinstrument** with **specphot**. This will at least set the picture types correctly, although we will find that we will have to change some of the defaults that are automatically set in **ccdproc** as a result of this operation.

As soon as you do this you will find yourself in the parameter editor staring at the page for the entire package **ccdred**. The only thing to check at this point is that "pixelty=real"; doubtless it will be. Getting out of this with a **CNTL-Z** will put us in the the parameter editor for **ccdproc** itself. Rather than set the parameters at this point (before we've even looked at anything!) let's exit with another **CNTL-Z** and come back to this stuff later.

Did it work? We can run **ccdlist** on our images to see if IRAF is going to successfully recognize the type of image (object, comp, flat, zero) and the filter number (for direct imaging). So we will want to say **ccdlist *.imh** at this point to get a list like that shown in Figure 3. Note that in the case of the spectroscopy example shown here the filter numbers are all "[0]", but in the case of direct imaging the filter numbers correspond to the filter bolt positions. The image type ("zero", "flat", "object", "comp") are also shown. These must be right if "ccdred" is to succeed. In newer versions of **ccdlist** it is possible to pull out the exposure information, which has to be discernible to **ccdproc** if you are going to be subtracting darks.

The things you should check at this point are whether the filter numbers are correct (that number in square brackets), and whether the object types make sense. If they don't, you are going to have to choose a different answer from the **setinstrument** list. To me, this always seems the most obscure part of CCDRED; if you are going to have trouble, it will be here. If you are stuck, ask someone, and if that gets you nowhere, try looking at a long version of your headers **imhead *.imh l+ | page** and then nose around by doing a **dir ccddb$kpno/*** until you find a file that will provide an appropriate translation.

If you are reducing non-KPNO/CTIO data and have different key-words for flats and

Figure 3: Sample outputs from **ccdlist**.

bias's and so on, you will need to set up your own translation file; just follow the example of one of the files you find in "ccddb$kpno". If you don't have the necessary information (really just the image type and filter numbers) you can add this information using **ccdhedit**; the best way would be to put the names of all the biases in a file *biasfile*, say, and then do a **ccdhedit @biasfile imagetype zero**; other examples can be found by doing a **help ccdhedit**. If you are going to be subtracting darks you are also going to need to translate some exposure time into the word "exptime" so that ccdproc can scale the darks.

4.3 Combining bias frames and flats

For our next act we want to combine the bias frames and flat field exposures obtained with a given filter. To combine the bias frames we will use **zerocombine** with the parameters shown in Figure 4. This will result in all the images with type "zero" being averaged together, but with the highest value being ignored when forming the average for any given pixel ("combine=maxreject"). In other words, if you have 30 bias frames, 29 will be averaged in producing the value for each pixel in the image "Zero", with the ignored one being the one that would have given the highest value for that pixel. This seems to do a good job of keeping radiation events out of your average bias frame. Note that even though the images were "short integer" (16-bit) to start out with, the averaging will be done as "reals" (32-bit) and that your final image, *Zero*, will be a 32-bit real image. The output will look like that shown in Figure 4, in which we see that **zerocombine** has successfully picked up all the right

Figure 4: Parameters and Output of **zerocombine**.

Figure 5: Parameters and Output for **flatcombine**.

images and averaged them together using the "maxreject" option.

We now want to do the same thing with our dome or projector flats. Here we do an **epar flatcombine** and make it look like that of Figure 5, with the "combine" option now set to "avsigclip". This will average together all the frames, compute the standard deviation σ of the average at each pixel, and recompute the average excluding all those that deviated by 3σ or more. This does a very credible job of getting rid of radiation events. We use the "avsigclip" option here rather than the "maxreject" used above, as the chances are greater that you will have a radiation event in a pixel on more than one frame, given the finite length of your flat field exposures. As the GEC chips are notorious for cosmic ray events, and since these are 30 minute exposures, this option for combining is probably the absolute best. If we had had many more frames (>10, say) we might have chosen to set "combine=median".

In this case **flatcombine** produced an image with name *Flat0.imh*. If we had a number of flats taken through different filters, then **flatcombine** would have produced a separate output image for each filter: *Flat0.imh, Flat1.imh, Flat2.imh, Flat3.imh* ....

How well did we do? We might want to examine the results before we go any further. If

Figure 6: Cosmic Ray removal in flats using "avsigclip" option in **flatcombine**.

you are using a SUN you can load the **tv** package and then say **display Flat0.imh 1** to get a picture of your final flat. To see the entire frame you may have to do a **set stdimage=imt800** first and then "click the mouse" on "FitFrame"; if you haven't used the SUN before I would grab someone who will show you how to do this kind of stuff. If you are not using a SUN, you will have to ask someone where the nearest image display is. Compare the final flat field with one of the individual flats to see if you are happy with your combining operation. In the example shown in Figure 6 we see that we just couldn't have done much better: "avsigclip" did its job and the multitude of cosmic rays have all vanished.

This would be a reasonable time for you to get rid of your individual bias and flat-field exposures that you have carefully combined. One relatively painless way is to put the names of all the individual calibration frames into a file by doing something like **ccdlist nite1*.imh names+ ccdtype="zero" > tempfile** followed by a **ccdlist nite1*.imh names+ ccdtype="flat" >> tempfile**. Do an **imhead @tempfile** to make sure that you don't have something in there that you want to save! (If we had executed those two **ccdlist** commands using **.imh* as the argument we would have gotten our combined frames *Zero.imh* and *Flat*.imh* as well; we would then have to delete those entries using the editor.) If you are sure, then do an **imdelete @tempfile**.

Figure 7: A sample of a long header from some Tololo data. Note the values for TRIMSEC and BIASSEC.

4.4 Taking a Look at Your Data: Picking a TRIMSEC and BIASSEC.

The next step is to decide what part of the chip contains useful data, and exactly where the overscan region is and what part of it you want to use for determining the fit to the bias-level. "CCDRED" knows these two regions as "trimsec", the section of the raw image that will be saved, and "biassec", the section of the raw image that will be used to fit the bias-level. As a reminder, IRAF uses the notation *imagename[x1:x2,y1:y2]* to describe that part of the image that goes from column "x1" to column "x2", and from row "y1" to row "y2".

If you are using one of the Kitt Peak or Tololo chips, and have relatively recent data, you will find image sections listed for TRIMSEC and BIASSEC in your headers. Pick an image and do an **imhead** *imagename* **l+** | **lprint** to get a printed listing of everything in the header. An example is shown in Figure 7. If you are doing direct imaging and you have these things in your header, you probably can ignore the rest of this section. Still, it wouldn't hurt you to make a few plots of your flat-field data to make sure that no one was being wildly conservative when they assigned a TRIMSEC to your chip.

If we do an **implot Flat0** we will get a plot across the middle line. An ":a 10" command will tell it to average 10 rows or columns (whichever it is about to plot); a ":c 250" would then plot 10 columns centered on column 250. Similarly a ":l 300" would plot 10 lines centered on line number 300. You can expand by placing the cursor on the lower-left corner of the region you want to expand and hitting "e" (that's a lower-case "e"), and then placing the cursor in the upper-right corner of the region you wish to expand and hitting any key. A "C" will tell you the cursor position—if you are willing to type it twice for every time you want to know this! You can also expand with an "E" (upper-case "E") but you will then have to follow this with an "A" to get a scale.

After looking at our implots (Figure 8) we decide to keep the data that runs from column 122 to 270, and from line 4 all the way through 576, the top of the image. Thus we choose

Figure 8: Row and Column plots through a flat-field exposure.

to have TRIMSEC=[122:270,4:576]. The bias region looks clean from column 388 through 417, and so we decide to have BIASSEC=[388:417,4:576]. Note that this differs very substantially from the "default" values listed in the headers (Figure 7). This is not uncommon in spectroscopic applications, but usually the defaults work just fine for direct imaging.

4.5 Processing Your Data: Pass 1.

Believe it or not, we are now ready to make our first pass through **ccdproc**, the task which will actually do the processing. At this stage we will get rid of only the "additive" effects discussed above in Section 2; how we do the flat fielding depends upon what sort of data we have.

We first must edit the parameter file for **ccdproc** so it resembles that of Figure 9. If you are happy with the default BIASSEC and TRIMSEC, then the entry for these could simply say "image"; this will tell "CCDPROC" to get the values from the header. If you are going to be subtracting an average dark exposure, then go ahead and turn that switch on ("darkcor=yes"), and put in the name of the averaged dark exposure ("dark=Dark"); in this case you had better be sure that **ccdlist** reports the exposure times correctly (a feature only available in version 2.8 and later). In the example shown at the end of the manual the observer has in fact not saved the individual bias frames but has averaged the biases at the telescope and the average bias is not called *Zero.imh* or anything special. In this case leave **zero=""** in the parameter file, and **ccdproc** will just pick out the first input image that happens to have type "zero".

OK, now let's run **ccdproc**. (See Figure 10.) The first thing that will happen is that it will ask if we want to fit the overscan region of the first image interactively or not; go ahead and say yes and take a look at the plot. We have started out with the defaults fitting a simple constant to the overscan region after rejecting any points 3 σ above or below the average, but

Figure 9: Parameters for the first pass through **ccdproc**

.

Figure 10: Doing the initial processing with **ccdproc**.

if you want to go to a higher order, be my guest: doing a ":o 50" followed by a "f" will fit a 50th order equation to the data. Once you are happy with the fit and exit with a "q" the function and order that you last used will be used with all subsequent images. If you answer "NO" the next time it asks if you wish to fit the overscan interactively, it will take this as a firm decision and just go ahead and process all your data.

If you did do a bunch of dark exposures and plan to use them (or at least see if you want to use them) this is the time to combine them, now that they have gone through the initial processing. Use **darkcombine** with "combine=avsigclip", "ccdtype=dark", and "exposure=yes" (but see note above under Section 3).

4.6 Flat-fielding

How we proceed from this point depends upon what kind of data you have, and what you plan to do with it.

In many applications (direct imaging, flux-calibrated spectroscopy) you can simply normalize your flat-fields to some average value and divide these into all your frames. In this case, skip to the section entitled "CCDPROC: The Next Pass". In the case that you are doing spectroscopy and want to preserve counts a little bit better than that, you will want to normalize the flat-fields by a low-order fit along the dispersion axis.

In some spectroscopic applications your flat fields will cover only a few pixels, as in the case of Echelle data or data obtained with fibers. Another example would be if you observe both stars and flat-fields through a skinny decker for some reason. In these cases you will need to extract all your spectra and then worry about flat-fielding. At least you are now done with this manual! Similarly, if you are dealing with slitlet data or some such, there are other flat fielding complications, and you should go from here to the "Multi-object Guide".

4.7 Fitting Your Flats

If you have elected to fit your flat fields along the dispersion axis, then we must first tell the system whether your spectra run along columns or rows. Load **twodspec** and then do a **setdisp *.imh disp=2** if your spectra run along columns, or a **setdisp *.imh disp=1** if your spectra run along lines.

Next load **longslit** and do an **epar response** until it resembles that of Figure 11. In this example the input image is *Flat0.imh* and the output image will be *nFlat0.imh.*

When you run **response** it will sum up all the columns in your flat-field (assuming the dispersion axis runs along columns) and do a fit on this sum. The "fit" shown in Figure 11 looks awful, but in fact we do not want to go to any higher order. After all, the whole point in dividing by a flat-field is to take out the gain variations, and if you were to fit low-frequency gain variations, then the flat-fields just won't be doing their job. So, do yourself a favor: stick to a straight line (or at least a very low order).

If you are using one of the older versions of "response" then you are going to have to modify the header at this point. To see if you have a problem do an **hselect nFlat0.imh**

Figure 11: Parameters for **response** and sample fit.

Figure 12: Parameters for combining skies.

“CCDMEAN” yes and see if it finds CCDMEAN set equal to 1.0. If not, edit this information into the normalized flat(s) by doing an **hedit nFlat0.imh CCDMEAN 1.0 add+ up+ ver+**.

4.8 CCDPROC: the Next Pass

At this point we want to divide by our flat-fields, and this is so simple you shouldn't even have to **epar ccdproc**. Instead, just type **ccdproc *.imh flatcor+ flat=”***imagename***”**, where *imagename* is “Flat*.imh” in the case that you haven't fit your flats, and *imagename* is “nFlat0.imh” in the case that you have fit your flat.

The astute among you will have caught on that if you don't intend to run your flat(s) through **response** then you could have done the flat field correction in the first pass through **ccdproc** by simply setting “flatcor=yes” and “flat=Flat*.imh” along with everything else in Figure 10. Feel free to do it that way the next time!

Remember, too, that if you did not have to **flatcombine** your flats because you did all this at the telescope, and these average flats are just inconspicuously sitting around with your other images, you needn't specify the names of these flats (e.g., **ccproc *.imh flatcor+**). In this case **ccdproc** will just seize the first thing it comes upon with type “flat” and the right filter number. This is what happens in the example given at the end of the manual.

4.9 So How Good Were Those Flats?

The moment of truth has arrived: how flat is the division of sky by your flat? If you have several sky exposures through each filter you will want to combine them before making this comparison. First put the names of all your sky exposures into a single text file by using **files nite1005.imh,nite1006.imh,nite1007.imh,nite1008.imh > allskies**; then **epar combine** until it looks like that of Figure 12. Note that this will do the combining filter-by-filter, and that it will scale by the mode before averaging (with clipping) so the fact that the sky levels are not all the same in each exposure will not cause a problem.

Figure 13: Sky and dome flat agree to 5% from one side of the slit to the other.

For each filter that you used you will have an image *Sky0.imh, Sky1.imh, Sky2.imh ...* which shows what the sky looks like when divided by the appropriate dome flat. Examine these using **display** and **implot** and see how well your domes did.

In Figure 13 we see a typical spectroscopic example. I've averaged 100 lines in order to better see the match. At the left edge of the slit we have ≈ 1370 counts; at the right, ≈ 1450. Thus the from one side of the slit to the other our dome flats match the sky to 80/1410, or about 5%. If you are doing long-slit work you may want to do better than this, but if you are going to be extracting the spectra of stars that are sitting in the middle of the slit, and you are merely concerned that sky on either side is taken out, then you need not do more.

What if you did want to correct your data for this disagreement? In the case of spectroscopy we couldn't very well just divide by the sky exposures— not unless you want the spectra of your program stars to look like the inverse of a G2V star. Instead, we will use **illum** (also in the **twodspec longslit** package) to collapse the sky spectrum along the dispersion axis and fit a smooth function in the spatial direction. Edit the parameters of **illum** until they resemble that of Figure 14. It is possible to set "nbins" to a larger value than 1, which would result in more than one fit—i.e., the correction function would depend upon wavelength. Since this is physically unrealistic, you might want to be certain that "nbins=1".

The simplest way to apply this correction to your data would be to write the names of your images into a file (**files *.imh > everything**) and then do an **imarith @everything / rSky0 cor//@everything verb+**. This will create a new image for each input image, with the image name having "cor" appended to the beginning of the old name.

What about the direct imaging case? If you have access to an image display, use it to

Figure 14: Parameters and output of **illum**.

Figure 15: Parameters for **mkskycor**.

display each of the sky images. Use this to guide you when you **implot** them. If you don't have an image display, you will just have to hunt around plotting (averaged) columns and rows to see what sort of variations you have.

In the example given in the back of the manual we find that although the dome flats did a good job of flattening the skies, the U sky is decidedly non-flat, with an amplitude of some 4%. In this case we will not use our sky flats to correct our program frames except for U.

Now the question is do we want to smooth the sky exposures before we use them, or use them straight? The CCDRED tasks are set up with the assumption that you will always want to smooth, but there are ways around this and reasons we might want to take this slightly tortuous path.

If the cosmetics of your chip are reasonably good (a few bad columns and a few low pixels, say) *and* if the structure in your sky exposures is of pretty low frequency, then you can smooth. (If you have taken exposures of the dark sky, and have poor counting statistics in your frames, then you are going to have to smooth in any event.) There are advantages to smoothing: for one thing, exposures of bright twilight will often show fringe patterns (particularly at U) that disappear as the sky grows darker, and smoothing will get rid of these.

If you want to try smoothing, then **eparm mkskycor** so it looks like that of Figure 15. (If you want to do this on more than one sky frame, you can do a **files Sky1,Sky2,Sky3 > skyfix** and then set "input=@skyfix" and "output=r//@skyfix".) After you run the task, you should divide the new images into your old and examine the resultant images (**imarith @skyfix / r//@skyfix test//@skyfix** will produce images with names *testSky1, testSky2*, and so on). If you are happy with these after you've examined them with **display** and **implot** then skip the next paragraph.

If you are not going to smooth your sky exposures then you are going to have to fudge your headers a bit to fool **ccdproc**. First let's clone them so that if we goof up we still have the originals around: if you just have one you want to use do a **imcopy Sky1 rSky1**; if you have more than one than **files Sky1,Sky3,Sky4,Sky6 > doem** followed by **imcopy @doem r//@doem** will do the specific ones you want. Next we need to edit the headers: **hedit r//@doem "MKILLUM" "fudge" add+ ver-**. Next we are going to have to normalize these sky exposures. Load the package **generic** and run the task **normalize r//@doem**. To be safe let's also do an **hedit r//@doem "CCDMEAN" 1 add+ ver-**. This last step is not needed in the current version of "CCDPROC", but let's not take more chances than we have to. OK, your "rSky" frames will now look the same to IRAF as if you had run them through **mkskycor**.

The final step in the processing will seem rather anti-climatic: simply do an **ccdproc *.imh illumcor+ illum="rSky*.imh"** to correct whichever images need correcting. If you've created only an rSky1.imh, then only those with filter 1 will get processed. (If you are using an older version of **ccdproc** you may bomb out at this point when it discovers you don't have an illumination correction for each filter—in that case, just make up a file containing the names of all the "U" exposures, for example, and feed this to **ccdproc** instead of "*.imh").

4.10 Dealing with Fringes

As discussed earlier, you really don't want to do this unless you absolutely have to, but if you insist I will give a crude outline to follow. First, you must construct a "master flat" from your "blank sky" exposures: for each filter create files containing the names of your blank sky exposures, and then use **combine** with the "avsigclip" option to get rid of the faint stars and cosmic rays. Determine the average sky value on each of these master by doing **implot**s on them, and then using **imarith** subtract this value so that your master fringe frames have an average value of zero and the fringe pattern goes positive and negative. Now the fun begins: for every program frame you must determine the scaling factor to use when subtracting the appropriate master fringe frame. Try multiplying the fringe frame by .5, 1.0, 1.5, and 2.0 to start out with, and subtract each of these scaled images from your program frame. Display the results on your image display, and decide which scale factor did the best job. Next, scale the fringe frame by several values centered around the one that did the best job using step sizes of .1. If you are really fussy you can probably do this a third time using intervals of 0.05. When you find the best one, keep it, and start on the next one.

4.11 Cleaning Up

If you are going to be using your images you will no doubt want to keep them around on disk for a while, but there are several rather gigantic files which "CCDRED" produces just to show you it can do it. You might consider doing a **delete logfile,plotfile**. Of course, if you really want to look at each and every fit to your overscan region you can do an **gkimosaic plotfile**, and if you want to save a record of everything you did you can do an **lprint logfile** (don't try reversing these two!).

If you want to put your data on tape, then find and **allocate** a tape drive (remember this?) and then **eparm wfits** until it looks like Figure 16. Be sure you have a write-ring in

Figure 16: Parameters for **wfits**.

place! If you wanted to specify the density of the tape then you need to say "mta.6250" for the "FITS filename". On a VMS VAX you must also have done an **! initialize/dens=6250 mua0: mydata**, where "mua0:" is the "real" name of the tape drive. If you are desparate to save tape and are going to be reading this with IRAF somewhere else you can also up the "FITS tape blocking factor" to "10". (I **do not** recommend setting "bitpix" to some other value; this will indeed save tape but believe me, you will regret it!)

4.12 A Long Last Word

Because examples are sometimes the easiest thing to refer to I am going to close this manual with an example of reducing a night of direct imaging data from the soon-to-be-bulldozed No. 1 0.9-m. Everything that I had to do to reduce these data is here.