

# A User's Guide to CCD Reductions with IRAF

Philip Massey (Feb 1997)

## Contents

- Introduction
- Why Your Data Needs Work And What to Do About It
- Doing It
- Outline of Reduction Steps
- Examining Your Frames to Determine the Trim and Bias Sections
- Setting things up: setinstrument, parameters of ccdproc, and ccdlist
- Combining Bias Frames with zerocombine
- The First Pass Through ccdproc
- Constructing a bad pixel mask
- Dealing with The Darks
- Combining the Flatfield Exposures
- Normalizing Spectroscopic Flats using response
- Flatfield Divisionm ccdproc Pass
- Getting the FlatFielding Really Right
- Combing the twilight and blank-sky flats
- Creating the Illumination Correction
- Finishing the Flatfielding
- Fixing Bad Pixels
  
- A How Many and What Calibration Frames Do You Need
- B The Ins and Outs of Combining Frames
- C Summary of Reduction Steps
  - C.1 Spectroscopic Example
  - C.2 Direct Imaging Example

## 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.

Additional resources you may wish to review are:

A Beginner's Guide to Using IRAF by Jeannette Barnes

Once you are done with this manual you may wish to go on to do stellar photometry on direct frames or to reduce slit spectrograph data. You can find details and sage advise in the following sources:

A User's Guide to Stellar Photometry with IRAF by Phil Massey and Lindsey Davis

A User's Guide to Reducing Slit Spectra with IRAF by Phil Massey, Frank Valdes, and Jeannette Barnes

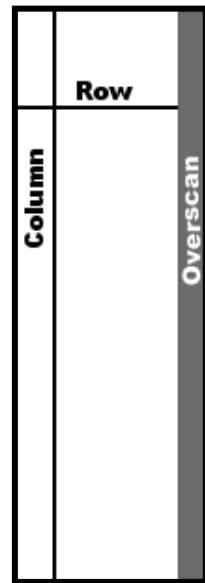
Copies of all these documents are available over the net; contact the IRAF group for details. In Sec.2 we will discuss whys and wherefores of CCD reductions in general terms. In Sec. 3 we will go through the actual IRAF steps involved in implementing these reductions, using spectroscopic data as the primary example. In the Appendices we provide a guideline to what calibration data you may want to collect while you're observing (Sec. A), and discuss the nitty-gritty of the algorithms available to you in IRAF for combining images (Sec. B). Finally we close with a summary of the reduction steps for spectroscopic data (Sec. C.1), and the reduction steps for direct imaging (Sec. C.2).

## 2. Why Your Data Needs Work And What to Do About It

This section will briefly outline how and why your CCD images need work. For a less heuristic treatment, see Sec. A, which discusses how many of what kind of calibration frames you need.

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 and/or underlying bias structure, and, if necessary, the dark current. The flatfield 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 (a few 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 number 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 in Sec. A. 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 flatfield your data. The variations in sensitivity are multiplicative, and we need to divide the data by the flatfield to 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 to just normalize the flatfield 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 flatfielding process is to see if your twilight sky exposures have been well flattened by this procedure; if not, we may have to correct for any remaining illumination gradients.

Some CCDs have a few pixels and/or partial columns where the response to light is not linear. For direct imaging one is usually happy to leave these bad regions alone: they’re perfectly apparent on the reduced frames. But for spectroscopic reductions you, may want to interpolate over these bad pixels. This step is known as “fixing bad pixels,” and there are some new, powerful tools for constructing bad pixel maps and then applying the interpolation.

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. If the fringe pattern on your frames simply scaled with exposure time, you would now be able to process your data; but in fact, fringes are caused by night-sky lines that may

change quite drastically in intensity through out the night. Thus for each and every affected program frames, you will have to “manually” determine what additional scaling factor is needed to adequately remove the fringes.

### 3. Doing It

Throughout this section we 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 A Beginner’s Guide to IRAF 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 his or her 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.

We will often call on the IRAF *ccdred* routines for combining frames in various ways. There are a number of very sophisticated algorithms for doing this. Details can be found by doing a help combine from within IRAF; for convenience, there is a short summary of the nitty-gritty of these various options given in Sec. B.

#### 3.1 Outline of Reduction Steps

The steps we will go through in order to process CCD frames are the following:

- Examine a flatfield exposure using *implot* and determine the area of the chip that contains good data and the area of the chip that contains good “overscan” information.
- Set up the translation table for the image headers by running *setinstrument*; this will also set various defaults within the *ccdred* package. Enter the proper *biassec* and *trimsec* into *ccdproc* at this time. Substantiate that the header translations are valid by running *cdlist*.
- Combine the individual bias frames using *zerocombine* to produce an averaged, combined bias image (Zeron 3, for example).
- Process all the frames to remove the *overscan* and average bias, and to trim the images (first pass through *ccdproc*). Be sure that you have the appropriate switch settings (*overscan+*, *trim+*, *zerocor+*, *darkcor-*, *flatcor-*, *illum-*, *fring-*, and that the name of the combined bias frame has been entered for the zero calibration image (*zero=Zeron3*, say).
- For spectroscopic applications, you may want to construct a bad pixel map at this point. We recommend taking five long-exposure (~3000 e/pixel) flats and 20-30 short-exposure (~50-100 e/pixel) flats (either dome flats or projector flats); combining each group using *flatcombine*; dividing one by the other; and running *ccdmask* (currently in the NMISC package) with its default setting. We will use this bad pixel map (*badmap*, say) at the end. Be sure to stick these images in where the subsequent reduction steps won’t affect them, such as a subdirectory. Note that this does not use the *fixpix* option in *ccdproc* itself, but relies on the newer (and not yet integrated) routines in the NMISC package. These are available as add-ons, if they are not already in your installation.
- If you are concerned about dark current:
  - Combine your dark frames using *darkcombine* with *scale=exposure*; call the combined image **Darkn3**, for example.
  - Examine the combined dark exposure and decide if you want to use it or not.
  - If you do want to use it, run everything through *ccdproc* again, this time specifying **darkcor+** and **dark=Darkn3**, for example.
- Combine your flatfield exposures using **flatcombine scale=mode reject=crreject gain=gain rdnoise=rdnoise**. This will reject cosmic rays and scale by the mode.
- (optional) For spectroscopic data, normalize the combined flatfield exposure along the dispersion axis by dividing it by a low-order fit using response in the TWODSPEC.LONGSLIT package.
- Process all the program frames and sky flats using the combined flatfield exposures: **ccdproc .imh ccdtype= “” flatcor+ flat=Flat\*.imh** (If you used response for your spectroscopic data, be sure to specify the normalized image as

the flatfield.) This will flatten your data to the first approximation (second pass through *ccdproc*).

- Do the final flattening correction on your data as follows:
  - Examine your longest exposures (or at least the ones with the most sky) in each filter using *display* and *imexamine* to determine if your flatfield exposures did an adequate job flattening your data. Are there significant gradients (> 1%) present in your sky values in direct imaging data? Is the spatial cut in spectroscopic data flat?
  - If you need to correct your data for any illumination problems revealed by the previous step create an illumination correction:
    - \*\*Combine any blank-sky or twilight frames with *combine*, scaling and weighting by the mode.
    - \*\*For spectroscopic data, use *illum* in the LONGSLIT.TWODSPEC package to create a slit function illumination correction from the combined sky flat.
    - \*\*For direct imaging, use *mkskycor* on your combined twilight or blank-sky flats to create a smooth illumination correction.
  - Finish flattening your data by turning on the illumination correction switch and specifying these illumination function in *ccdproc* (third and final pass).
- For spectroscopic applications, you may wish to fix bad pixels as a final step: run *fixpix* in the NMISC package, using *badmap* as the mask, and leaving things at their default setting (linterp=1, cinterp=2).

### 3.2 Examining Your Frames to Determine the Trim and Bias Sections

The first step in reducing your data 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. The CCDRED package 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 1. 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 flatfield data to make sure that no one was being wildly conservative when they assigned a TRIMSEC to your chip.

If we do an *implot n30010*, 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—but you may have to type it twice. A space bar [amazing!] will also do this, kind of. 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.

The plots (Fig. 2) reveal that the first few rows of our chip were not illuminated, and that the last few columns (3065-3072) are not good. We decide to keep the section [1:3064,10:101] for our TRIMSEC. Note that if had the slit extended over the entire chip format, we would in fact have been happy with the defaults in the header, [1:3064, 1:101]. In addition, the BIASSEC (= [3074:3104, 1:101]) appears to be fine. For direct imaging, the header values for BIASSEC and TRIMSEC are usually correct.

### 3.3 Setting things up: *setinstrument*, parameters of *ccdproc* and *cdlist*

The reduction of our CCD data mostly takes place within the *ccdred* package. The tasks in this package rely heavily on 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 flatfield from a R filter flatfield, and so on. The command for setting up this translation file is *setinstrument* in the CCDRED package. So load NOAO.IMRED.CCDRED, and then type *setinstrument*. If you type a ? you will get the list shown in Figure 3. In this case we are trying to reduce spectroscopic data taken with the FORD 3K x 1K chip

(formatted to 3072 x 101) in the Kitt Peak 0.9-m spectrometer WhiteCam (=GoldCam, but on the White Spectrograph). We select *specphot* as good generic spectroscopic defaults.

```

m920004[289_506][real]: M-92 V
No bad pixels, min=0, max=0. (ok)
Line storage mode, physdim [289,506], length of user area 2673 su.
Created Thu 14:12:27 15-Apr-2004, Last modified Thu 14:12:31 15-Apr-2004
Pixel file 'm920004.fits'[ok]
EXTEND = F / File may contain extensions
ORIGIN = 'NOAO-IRAF FITS Image Kernel December 2001' / FITSfile originator
IRAF-TLM= '14:12:31 (15/04/2004)'
OBJECT = 'M-92 V' /
DATE = '2004-04-13 T23:49:58'
IRAF-MAX= 1.064900E4 / DATA MAX
IRAF-MIN= 4.810000E2 / DATA MIN
CCDPICNO= 7 / ORIGINAL CCD PICTURE NUMBER
EXPTIME= 60 / ACTUAL INTEGRATION TIME (SECONDS)
DARKTIME= 60 / TOTAL ELAPSED TIME (SECONDS)
OTIME = 60 / SHUTTER OPEN TIME (SECS)
IMAGETYP= 'OBJECT' / OBJECT DARK, BIAS, ETC.
DATE-OBS= '01/09/87' / DATE DD/MM/YY
RA = '17:17:24.00' / RIGHT ASCENSION (TELESCOPE)
DEC = '43:03:42.00' / DECLINATION (TELESCOPE)
EPOCH = 1987.70 / EPOCH OF RA AND DEC
ZD = '20:17:00.00' / ZENITH DISTANCE
UT = '13:30:24.00' / UNIVERSAL TIME
ST = '18:43:08.00' / SIDEREAL TIME
DETECTOR= 'RCA1' / DETECTOR (CCD TYPE, PHOTON COUNTER, ETC)
CAMTEMP= -104.46 / CAMERA TEMPERATURE, DEG C
DEWTEMP= -187.88 / DEWAR TEMPERATURE, DEG C
FILTERS= '6 0' / FILTER BOLT POSITIONS
TVFILT = 0 / TV FILTER
COMPLAMP= 0 / COMPARISON LAMP
CCDSEC = '30:318,5:510' / ORIENTATION TO FULL FORMAT FRAME
ORIGSEC = '11320,1:512' / ORIGINAL SIZE OF FULL FORMAT FRAME
CCDSUM = '1 1' / ON CHIP SUMMATION (XY)
HISTORY 'KPNO-IRAF' /
HISTORY '13-01-92' /
HISTORY 'New copy of Au2/jscoby/m92010.imh'
HISTORY 'New copy of urs/Au2/jbarne/iraf/m92/m92010.imh'
AIRMASS = 1.066077
UTMIDDLE= '3:30:54.0'
WCSDIM = 2
LTM1_1 = 1.
LTM2_2 = 1.
WAT0_001= 'system=physical'
WAT1_001= 'wtype=linear'
WAT2_001= 'wtype=linear'
TRIM = 'Apr 13 15:49 Trim data section is [30:318,5:510]'
OVERSCAN= 'Apr 13 15:49 Overscan section is [318:350,5:510] with mean=509.653'
ZERO COR= 'Apr 13 15:49 Zero level correction image is m920001.fits'
FLATCOR = 'Apr 13 15:49 Flat field image is m920003.fits with scale=1337.823'
BIASSEC = '289:289,1:506'
LTV1 = -29.
LTV2 = -4.
CCDPROC = 'Apr 13 15:49 CCD processing done'

```

Figure 1: A sample of a long header from an Astronomy 480 reduction of images of the globular cluster M92. Note the values for TRIM = and OVERSCAN = (TRIMSEC and BIASSEC) The values in this image header will NOT correspond to the discussion here, but you should be able to see how the parameters relate.

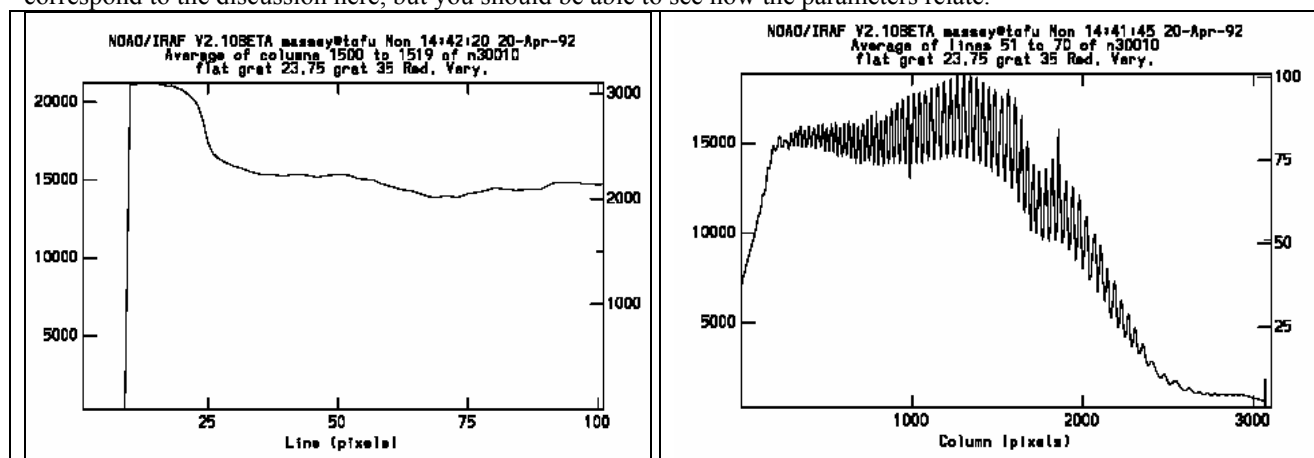


Figure 2: Plots through a flatfield exposure. Note the rather severe fringing evident in the flatfield exposure along the dispersion axis. These will hopefully come out in the flatfielding.

```

cc> setinstrument
Instrument ID (type ? for a list) (?): ?

direct      Current headers for Sun plus CCDPROC setup for direct CCD
specphot    Current headers for Sun plus CCDPROC setup for spectrophoto-
            tometry, ie GoldCam, barefoot CCD
foe         Current headers for Sun plus CCDPROC setup for FOE
fibers      Current headers for Sun plus CCDPROC setup for fiber array
coude       Current headers for Sun plus CCDPROC setup for Coude
cyrocam     Current headers for Sun plus CCDPROC setup for Cryo Cam
echelle     Current headers for Sun plus CCDPROC setup for Echelle
kpnoheaders Current headers with no changes to CCDPROC parameters
fits        Mountain FITS header prior to Aug. 87 (?)
camera      Mountain CAMERA header for IRAF Version 2.6 and earlier

Instrument ID (type q to quit): specphot

```

Figure 3: Possible answers to *setinstrument*. The user has chosen the “specphot” default.

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 the pixel type of the output images, and the pixel type that will be used in our calculations, are both real. If you want to keep your output images in 16-bit “short” integer format, you should nevertheless retain the calculation type to be “real” or serious problems may occur when you do flatfield division. Getting out of this with a CNTL-Z will put us in the parameter editor for the *ccdproc* task (Fig. 6). This task is what we will use for doing the “image crunching” – removing bias, trimming, and flatfield division.

At this point we should enter the values for the *biassec* and *trimsec* parameters. Note that the default (for “specphot”) is that *biassec*=image, which is correct—we are perfectly happy with the answer we found in the image header above for the bias-section ([3074:3104, 1:101]). We need to explicitly enter the value for the trim-section, however, as [1:3064,10:101]. Had we selected, say, “*direct*” rather than “*specphot*” the values both for *biassec* and *trimsec* would have defaulted to “image,” would probably have been correct.

Rather than worry about the other parameters now, let us simply exit with a CNTL-Z.

Figure 4 – Sample outputs from *ccdlist*. [Figure not shown here.]

Did *setinstrument* do its job in setting up the translation table? To check this, 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 4. [Not shown here.]

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 the tasks in CCDRED are to succeed.

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, there is some inconsistency between the keywords in your headers and the translation table set up by your having run *setinstrument*. If you are stuck, try looking at a long version of your headers (**imhead .imh** | **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 biases and so on, you will need to set up your own translation file; just follow the example in “*ccddb\$kpno/direct.dat*,” say. 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 “darktime” so that *ccdproc* can scale the darks. If you want to use a “superior” combining algorithm, it would also be convenient to have the read-noise (in electrons) and gain (in electrons/ADU) in your headers, although you could simply enter these values in the various combining tasks.

### 3.4 Combining Bias Frames with *zerocombine*

For our next act we want to combine the bias frames into an average frame. To combine the bias frames we will use *zerocombine* with the parameters shown in Figure 5.

```
cc> lpar zerocombine
input =          List of zero level images to combine
(output = "Zero")  Output zero level name
(combine = "average")  Type of combine operation
(reject = "minmax")  Type of rejection
(ccdtype = "zero")  CCD image type to combine
(process = no)      Process images before combining?
(delete = no)       Delete input images after combining?
(clobber = no)      Clobber existing output image?
(scale = "none")    Image scaling
(statsec = "")      Image section for computing statistics
(nlow = 0)          minmax: Number of low pixels to reject
(nhigh = 1)         minmax: Number of high pixels to reject
(nkeep = 1)         Minimum to keep (pos) or maximum to reject (neg)
(mclip = yes)       Use median in sigma clipping algorithms?
(lsigma = 3.)       Lower sigma clipping factor
(hsigma = 3.)       Upper sigma clipping factor
(rdnoise = "0.")    ccddclip: CCD readout noise (electrons)
(gain = "1.")       ccddclip: CCD gain (electrons/DN)
(snoise = "0.")     ccddclip: Sensitivity noise (fraction)
(pclip = -0.5)      pclip: Percentile clipping parameter
(blank = 0.)        V value if there are no pixels
(mode = "ql")
```

Figure 5 -- Parameters and Output of *zerocombine* (image names not included)

The default parameters 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. In other words, if you have 10 bias frames, 9 will be averaged in producing the value for each pixel in the image “Zeron3” ignoring the highest value at each pixel. (The various options available for combining images are discussed in Sec. B). This option will 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 “Zeron3” will be a 32-bit real image. The output will look like that shown in Figure 5 in which we see that *zerocombine* has successfully picked up all the right images and averaged them together using the “minmax” option.

### 3.5 The First Pass Through *ccdproc*

We are now ready to process our data through *ccdproc* in order to do the “simple” stuff: we will subtract the pedestal determined from the overscan region, remove any remaining bias structure, and trim the image. Edit the parameters for *ccdproc* as shown in Fig. 6, making sure that the switches overscan, trim, and zerocor are on, but that darkcor, flatcor, and illumcor are off. Check that the correct values have been entered for biassec and trimsec. Finally, edit in the name of the combined bias frame as the zero image.

When we run *ccdproc*, we will be asked if we want to fit the overscan vector interactively. The first couple of times, through, answer “yes,” and you will see a plot something like that shown in Fig. 7. The fit shown in Fig. 7 is somewhat unusually flat. It is not unusual for bigger chips to exhibit a gradient of an ADU or two from one end to the other. However, even in these cases one might want to retain a single scalar (:order=1) as the fit, as we will be subtracting off the combined bias frame. Thus any residuals not removed by fitting the overscan region will probably be removed in subtracting the combined bias frame: remember, the overscan region exists in order to monitor things that change from exposure to exposure, such as small differences in the pedestal level. If you were to find that you had different gradients from frame to frame, then you would want to use a higher order fit, but still maybe nothing higher than a straight line. You can change the order while looking at the plot by doing a (:order=2, say, followed by an f for a new fit. The new order will be retained for subsequent frames.

After you have looked at a few of the overscan plots, answer NO (note the capitals) to the question “Fit overscan vector for bleh-bleh interactively?” Fig. 8 shows (but not shown here) the output from this first pass through *ccdproc*.

```

images = ""      List of CCD images to correct
(output = "")   List of output CCD images
(ccdtype = "object")  CCD image type to correct
(max_cache = 0)  Maximum image caching memory (in Mbytes)
(noproc = no)   List processing steps only?
(fixpix = yes)  Fix bad CCD lines and columns?
(overcan = yes) Apply overscan strip correction?
(trim = yes)    Trim the image?
(zeroor = yes)  Apply zero level correction?
(darkor = yes)  Apply dark count correction?
(flator = yes)  Apply flat field correction?
(illumor = no)  Apply illumination correction?
(fringeor = no) Apply fringe correction?
(reador = no)   Convert zero level image to readout correction?
(scanor = no)   Convert flat field image to scan correction?
(readaxis = "line")  Read out axis (column|line)
(fixfile = "")  File describing the bad lines and columns
(biassec = "")  Overscan strip image section
(trimsec = "")  Trim data section
(zero = "")     Zero level calibration image
(dark = "")     Dark count calibration image
(flat = "")     Flat field images
(illum = "")    Illumination correction images
(fringe = "")   Fringe correction images
(minreplace = 1) Minimum flat field value
(scantype = "shortscan") Scan type (shortscan|longscan)
(nscan = 1)     Number of short scan lines
(interactive = no) Fit overscan interactively?
(function = "legendre") Fitting function
(order = 1)     Number of polynomial terms or spline pieces
(sample = "")   Sample points to fit
(naverage = 1)  Number of sample points to combine
(niterate = 1)  Number of rejection iterations
(low_reject = 3) Low signa rejection factor
(high_reject = 3) High signa rejection factor
(grow = 0)     Rejection growing radius
(mode = "q")

```

Figure 6 -- The parameters for *ccdproc*. We have left the parameter *biassec* equal to "image" (ignore) to use the region listed in the header, but explicitly stated the image section to use for trimming (*trimsec*). We have inserted the name of the combined bias image as the zero image.

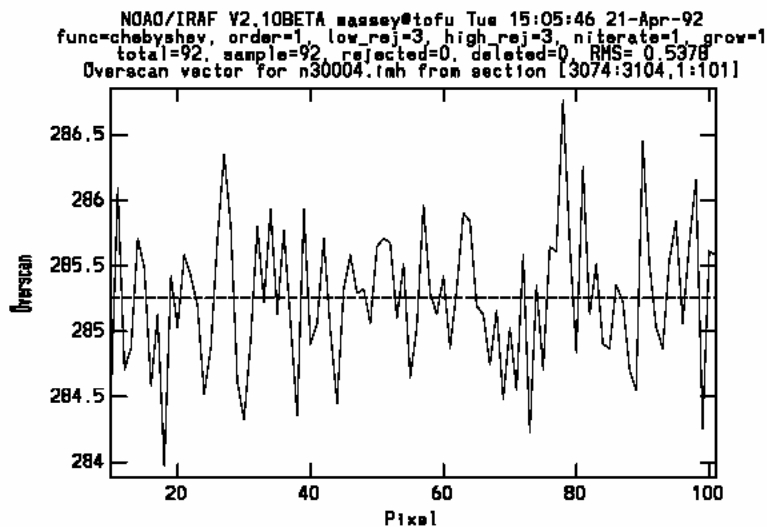


Figure 7 -- The overscan region for this chip is exceptionally well behaved, and a constant value is an excellent fit. However, one should not exceed a straight-line fit (:**order 2** followed by an **f**) if one can help it.

Note that *ccdproc* is smart enough to know what steps have and haven't been done; you are never in any danger of redoing a particular step. If you were to rerun *ccdproc* at the end of this step, without changing any of the switches, nothing would happen. However, if you were to turn other switches on (such as flatfield division!) those steps would then take place.

### 3.6 Constructing a bad pixel mask

(This section is not included in these instructions as we do not do this during Astronomy 480. Please see the original document for details.)

### 3.7 Dealing with The Darks

We argued earlier that the dark current is unlikely to be significant, but it wouldn't kill us to check that. We have already removed the overscan and bias level from our dark exposures, so any counts we see on the dark frames are probably real dark current (or a light leak!). We can either examine an individual dark exposure, or combine our dark exposures using *darkcombine*. The parameters for *darkcombine* are shown in Fig. 9. The reject algorithm chosen is again the simplest (and safest); we throw away the highest value at each pixel in constructing the average by specifying *minmax* with *nlow=0* and *nhigh=1*; see Sec. B for more details.

Note that *darkcombine* is smart enough to select only dark frames; and furthermore, will combine them scaling by the exposure times if they are different.

Examine your dark exposure by using *display* and *imexamine*. Are there significant counts there? If the answer is “yes,” we could choose to do the dark scaling and subtraction now, simply run:

- `ccdproc n3*.imh darkcor+ dark=Darkn3` (for the example in these instructions)

```
cc> lpar darkcombine
input =          List of dark images to combine
(output = "Dark")  Output dark image root name
(combine = "average")  Type of combine operation
(reject = "minmax")  Type of rejection
(ccdtype = "dark")  CCD image type to combine
(process = yes)     Process images before combining?
(delete = no)      Delete input images after combining?
(clobber = no)     Clobber existing output image?
(scale = "exposure")  Image scaling
(statsec = "")     Image section for computing statistics
(nlow = 0)         minmax: Number of low pixels to reject
(nhigh = 1)        minmax: Number of high pixels to reject
(nkeep = 1)        Minimum to keep (pos) or maximum to reject (neg)
(mclip = yes)      Use median in sigma clipping algorithms?
(lsigma = 3.)      Lower sigma clipping factor
(hsigma = 3.)      Upper sigma clipping factor
(rdnoise = "0.")   ccclip: CCD readout noise (electrons)
(gain = "1.")      ccclip: CCD gain (electrons/DN)
(snoise = "0.")   ccclip: Sensitivity noise (fraction)
(pclip = -0.5)    pcclip: Percentile clipping parameter
(blank = 0)        Value if there are no pixels
(mode = "ql")
```

Figure 9 -- Parameters for *darkcombine*.

### 3.8 Combining the Flatfield Exposures

We next want to combine the flatfield exposures. For direct imaging a separate flatfield image is needed for each filter. Fortunately, the CCDRED tasks pay attention to the “subset parameter,” which, for direct imaging, is defined in terms of the filter position. (For coudé data, the subset parameter is the grating position, “gratpos,” which makes sense.)

The parameters for the *flatcombine* task are shown in Fig. 10. The *reject* option is now set to *crreject* rather than the *minmax* option used for the bias frames; this will do a fine job of removing any radiation events from your flatfield exposures. (You could have equally well used *avsigclip*, in which case the “typical” sigma would have been determined from the data itself rather than an *a priori* knowledge of the noise characteristics of your CCD. See Sec. B for more details.) (However, none of these options would work right if the data had not first been processed.)

Note that we have decided not to use as input simply `n3*.imh` for running *flatcombine*; if you refer back to Fig. 4, you will see that there are actually projector flats taken at each new telescope position, in addition to the dome-flats exposures (n30007-12). Until we have reduced these data we do not know exactly the best way to remove the horrendous fringing pattern evident in Fig. 2, but for now we simply want to use the dome-flats. When we run *flatcombine*, we will see output like that in Fig. 10.

```

cc> lpar flatcombine
input =          List of flat field images to combine
(output = "Flat") Output flat field root name
(combine = "average") Type of combine operation
(reject = "avsigclip") Type of rejection
(ccdtype = "flat") CCD image type to combine
(process = yes) Process images before combining?
(subsets = yes) Combine images by subset parameter?
(delete = no) Delete input images after combining?
(clobber = no) Clobber existing output image?
(scale = "mode") Image scaling
(statsec = "") Image section for computing statistics
(nlow = 1) minmax: Number of low pixels to reject
(nhigh = 1) minmax: Number of high pixels to reject
(nkeep = 1) Minimum to keep (pos) or maximum to reject (neg)
(mclip = yes) Use median in sigma clipping algorithms?
(lsigma = 3.) Lower sigma clipping factor
(hsigma = 3.) Upper sigma clipping factor
(rdnoise = "0.") ccdclip: CCD readout noise (electrons)
(gain = "1.") ccdclip: CCD gain (electrons/DN)
(snoise = "0.") ccdclip: Sensitivity noise (fraction)
(pclip = -0.5) pclip: Percentile clipping parameter
(blank = 1.) Value if there are no pixels
(mode = "ql")

```

Figure 10 -- Parameters and Output for flatcombine.

The default parameter of flatcombine also called for scaling each of the individual frames by the [inverse] of the mode. This then allows for the possibility that lamps providing the flatfield illumination have varied during the series of exposures—usually a true assumption. By scaling these to some common value one obtains a less biased average given that we are rejecting some pixels.

Had we been doing direct imaging through a variety of different filters, we would have obtained one averaged flatfield exposure for each filter; *flatcombine* would have (by default) used the filter number shown by *ccdlist* to decide what to combine with what.

### 3.9 Normalizing Spectroscopic Flats using *response*

(This section is not included here since we will not be doing any spectroscopy. See the original document for instructions.)

### 3.10 Flat-field Division: *ccdproc* Pass 2

We now will make our second pass through *ccdproc*, this time letting it do the flatfield division. Although we will be overwriting the images (it's hard to avoid this in *ccdproc*), this step is still perfectly reversible, and if we find that we need additional corrections to the flatfields, that is easily made by a third pass through *ccdproc*. So we first edit the parameters of *ccdproc* to turn on *flatcor* and to explicitly give it the correct name of the flatfields, which in this case is the normalize output from *response*: nFlat0. The parameters are shown in Fig. 13. (Not shown here.)

In the case of direct imaging we would likely have set the flatfield names to **Flat\*.imh**. See Sec. C.2 for a full example.

### 3.11 Getting the Flat-Fielding Really Right

How well did we do on our flatfielding? We need to now really look at our data to evaluate this. Start with a long exposure (something containing lots of sky) and display it and use *imexamine* to make cuts at various places on the images. For spectroscopic data, we want to mainly see that the spatial cut is uniform.

For the data reduced here, our longest exposure was none too long. The left plot in Fig. 14 shows the sky averaged over many columns for the longest exposure. This plot was made by using *implot imagename* and then an **:a 100** and a **:c 1500** to plot the average of 100 columns centered on column 1500. We can see that we don't have many counts in the sky. However, it does look like the right side is a bit lower than the left side.

We next examine one of our twilight sky exposures. This is shown in the right panel of Fig. 14. In this spatial, cut there is a clear gradient present, although it is only 2% from one side to the other. This is consistent with what we see in the exposure of the object, however, and we decide to remove it.

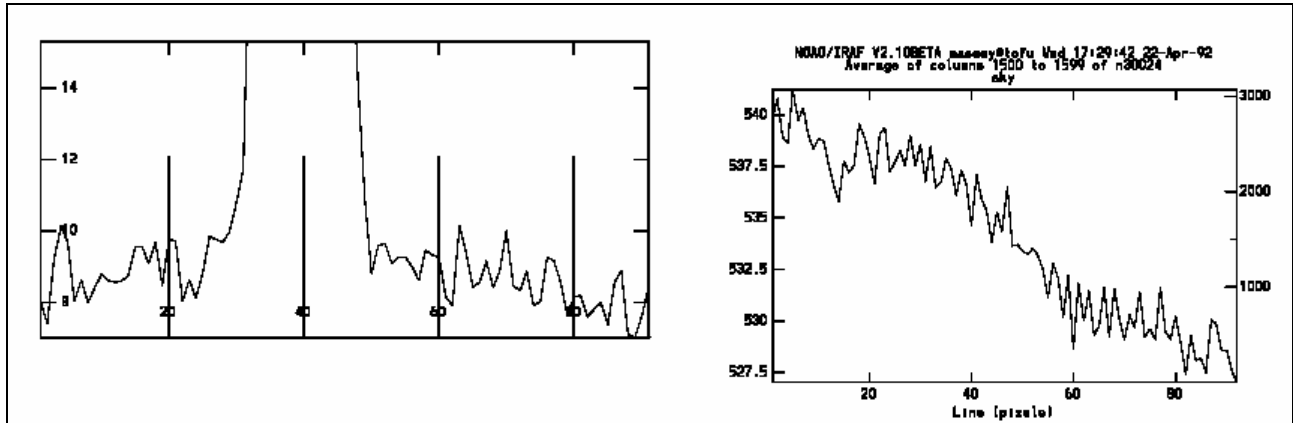


Figure 14 -- The cut along the spatial axis of a long program exposure appears to be consistent with a cut along the spatial axis of a bright twilight sky exposure: there is a 2% gradient from one side of the chip to the other.

### 3.11.1 Combining the twilight-blanksky flats

In order to determine the slit illumination function more accurately than that likely to be achieved with the dome flatfield, two exposures of the sky were obtained. (One would like to have obtained at least three such exposures, moving the telescope slightly between each one, [see Sec. A] but you can't always get what you want.) We will combine these two frames using the generic routine *combine*:

- **combine n30023, n30024 Sky reject=avsigclip scale=mode weight=mode subsets- blank=1**

This produces output shown in Fig. 15. If we had been doing this using direct imaging, we would have specified subsets, to make sure this was done in a filter-by-filter manner; in that case, the output images would have been automatically named Sky1, Sky2, Sky3, and so on, with the extension determined by the filter number. This is shown explicitly in our example of direct imaging reduction in Sec. C.2.

```
cc> combine n30023,n30024 Sky subsets- reject=avsigclip scale=mode weight=mode
Apr 22 13:56: IMCOMBINE
combine = average
reject = avsigclip, mclip = yes, lsigma = 3., hsigma = 3.
blank = 1.
      Images  Mode  Scale  Weight
      n30023 177.16 0.659 0.639
      n30024 56.393 2.071 0.361

Output image = Sky, ncombine = 2
cc>
```

Figure 15 -- Combining two sky exposures, scaling and weighting by the mode.

### 3.11.2 Creating the Illumination Correction

Having decided to correct our data (which have already been divided by the flatfields, remember), we need to create an illumination correction. For spectroscopic data, this will be created from the combined sky exposure (which has also been flatfielded), and will be done by fitting a function along the spatial axis, collapsing the image in the dispersion direction. (In practice, we let the slit function be a slight function of wavelength, and so do this procedure three or four times through-out the length of the spectrum. For direct imaging, we will use our blank-sky (or low-level twilight) exposures (which have been divided by the flatfield) by smoothing them extensively and using the smoothed image to correct any large-scale gradients.

**Spectroscopic:** (excluded)

**Direct imaging:** *mkskycor* If we have combined blank-sky flats in order to remove large-scale gradients, we can then use *mkskycor* to smooth the combined blank-sky frame. The parameters of *mkskycor* will resemble those of Fig. 18 (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=n//@skyfix”.) After you run the task, you should divide the new images into your old and examine the resultant images:

**imarith @n3skyfix / n//@skyfix test//@n3skyfix**

will produce images with names testSky1, testSky2, and so on.

Figure 17 – [Not shown here.] The wavelength bins for our *illum* run are shown in the top panel under the spectrum of the sky. The middle plot shows the fit along the spatial cut of one of the five wavelength regions. The bottom plot shows the spatial plot along the output image.

```
cc> lpar mkskycor
  input =          Input CCD images
  output =        Output images (same as input if none given)
(ccdtype = "")    CCD image type to select
(xboxmin = 5)    Minimum smoothing box size in x at edges
(xboxmax = 0.25) Maximum smoothing box size in x
(yboxmin = 5)    Minimum smoothing box size in y at edges
(yboxmax = 0.25) Maximum smoothing box size in y
  (clip = yes)    Clip input pixels?
(lowsigma = 2.5) Low clipping sigma
(highsigma = 2.5) High clipping sigma
(ccdproc = "")   CCD processing parameters
  (mode = "ql")
```

Figure 18 -- Parameters for *mkskycor*.

If you have well-exposed twilight flats, rather than blank-sky flats, you may instead find that you wish you had used the twilight sky exposures as your flatfield instead of the dome flats. Do you have to reread all the data from disk and begin again. No. We can cheat, and fool IRAF into thinking that it is simply using the “flatfielded” twilight flats as an “illumination correction” for the “flatfielded” program frames. If we don’t smooth the twilight-sky exposures, then this is algebraically equivalent (other than integer-truncation) to never having used the dome-flats at all. However, we must first fix the headers so that *ccdproc* will be willing to swallow the combined twilight sky exposures as an “illumination correction”:

- `hedit Sky*.imh MKILLUM “fake” add+ ver- show+`

You are now ready to proceed to the final steps.

### 3.12 Finishing the Flatfielding

To correct for this illumination function, simply do the following to whichever images need correcting:

- `ccdproc .imh illumcor+ illum=”nSky.imh”`

If you’ve created only an *nSky1.imh*, then only those with filter 1 will get processed. If you are using unaltered twilight flats, then of course you will have to substitute *illum=”Sky\*.imh”* in the above.

### 3.13 Fixing Bad Pixels

The final step in our reductions will be to interpolate over non-linear pixels using our bad pixel map. Do this using the task *fixpix* in the NMISC package.

- `fixpix *.imh mask=calib/badmap linterp=1 cinterp=2`

Congratulations! You're done, and now ready to go on to do photometry on your frames or extract some spectra.

### **A 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, and what chip you are doing it with. 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  $\sqrt{2}$ . If instead you use the average of 25 bias frames, the noise would be increased by only 10%. However, with modern CCDs with read-noise of a few electrons, hardly anyone finds his/herself in this regime any more. Particularly 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 flatfielding 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. These days, most CCD's have read-noises on the order of a few electrons, with the gain usually set so that you are at best barely sampling the read-noise (e.g., < 3 ADU). In this case there isn't much reason for you to overdo it on the biases; 10 of them will bring the effective noise below the digitization noise of a single exposure. You may want to make a new sequence of biases 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. I usually take a few of these 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!

**bad pixel data:** If you are doing spectroscopy and would like to correct for non-linear bad pixels/partial columns, then you should take a series of long and short exposures of some type of flatfield (dome flat or projector lamp). I typically aim for five exposures of several thousand e/pixel followed by 20 or so exposures that are about 100 e/pixel.

**flatfield exposures:** At a minimum, flatfield 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 take 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 twilight sky, stepping the telescope perpendicular to the slit orientation. In both cases, you should make sure that tracking is on and that the telescope is clear of the dome. You will find that you

need to keep increasing the exposure time to maintain an illumination level of ~10,000 electrons.

**blank 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 of 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 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 usually come out nicely with the dome flats; but if you are doing deep exposures in V, R, or I with a chip that fringes a lot, 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 fringe; avoid using chips that fringe a lot, 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 severe.

## B The Ins and Outs of Combining Frames

There are a number of powerful and sophisticated algorithms available in IRAF V2.10 for combining images. In particular there are a number of clever “rejection” criterion you can use for “hopefully” eliminating cosmic rays or stars in twilight exposures without “hopefully” eliminating the data you want to keep. A complete description can be found by *help combine*. In this section we provide a quick “astronomer’s review” of the rejection choices.

The parameters for the combine task are shown in Fig. 19. All of the various CCDRED combining routines (*flatcombine*, *darkcombine*, *zerocombine*) simply call combine with the appropriate switch settings.

Figure 19 – The parameters for the combine task.

In combining  $n$  images, the routine must decide at each pixel which of the  $n$  data values, if any, to reject in forming the average. Deciding what point(s) might be discrepant can be broken into three categories:

**The trivial:** The simplest, But for many applications the best, is the **reject=minmax** option. In this, the same number of values are excluded in determining the “best” average at each pixel. This assumes nothing about the data, or the expected spread in data value at each pixel, but is efficient at removing bad apples at the cost of always computing the average from fewer data values than what is available. Still, this works fine if you have data that you don’t mind losing; e.g., in combining biases, say, or darks.

input =	List of images to combine
output =	List of output images
(pfile = "")	List of output pixel list files (optional)
(sigma = "")	List of sigma images (optional)\n
(ccdtype = "")	CCD image type to combine (optional)
(subsets = no)	Combine images by subset parameter?
(delete = no)	Delete input images after combining?
(clobber = no)	Clobber existing output image?\n
(combine = "average")	Type of combine operation
(reject = "none")	Type of rejection
(project = no)	Project highest dimension of input images?
(outtype = "real")	Output image pixel datatype
(offsets = "none")	Input image offsets
(masktype = "none")	Mask type
(maskvalue = 0.)	Mask value
(blank = 0.)	Value if there are no pixels\n
(scale = "none")	Image scaling
(zero = "none")	Image zero point offset
(weight = "none")	Image weights
(statsec = "")	Image section for computing statistics\n
(lthreshold = INDEF)	Lower threshold
(hthreshold = INDEF)	Upper threshold
(nlow = 1)	minmax: Number of low pixels to reject
(nhigh = 1)	minmax: Number of high pixels to reject
(nkeep = 1)	Minimum to keep (pos) or maximum to reject (neg)
(mclip = yes)	Use median in sigma clipping algorithms?
(lsigma = 3.)	Lower sigma clipping factor
(hsigma = 3.)	Upper sigma clipping factor
(rdnoise = "0.")	ccdclip: CCD readout noise (electrons)
(gain = "1.")	ccdclip: CCD gain (electrons/DN)
(snoise = "0.")	ccdclip: Sensitivity noise (fraction)
(sigscale = 0.1)	Tolerance for sigma clipping scaling correction
(pclip = -0.5)	pdclip: Percentile clipping parameter
(grow = 0)	Radius (pixels) for 1D neighbor rejection
(mode = "cl")	

- **minmax** With this rejection algorithm, at each pixel there will be *nlow* low pixels and *nhigh* high pixels rejected. Thus to reject only the highest value in combining images we set **reject=minmax nhigh=1 nlow=0**. This is the default setting for *zerocombine*, and will do an excellent and fast job.

**Sigma determined from CCD noise characteristics:** In the schemes **reject=ccdclip** and **reject=crreject**, we assume we know what a reasonable spread is of our data values at each pixel, given the average value and the known gain  $g$  (in  $e/ADU$ ) and read-noise  $r$  (in  $e$ ). If the average data value at a given pixel is  $I$ , then the number of electrons at that pixel is  $g \times I$ , and we expect from Poisson statistics that spread in the data should have a value of sigma ( $\sigma$ ) in ADU's, of

$$\sigma_{ADU} = \frac{\sqrt{g \times I + r^2}}{g}$$

This is the most “legitimate” (mathematically justified) rejection criteria, and is suitable when you haven’t mucked around (subtracted sky, or averaged or summed frames) with the data.

- **ccdclip** With this rejection algorithm, we first guess  $I$  at a given pixel by taking the median (**mclip=yes**), computing the expected spread in ADU's based upon this value and the specified values of the gain (*gain*) and read-noise (*rdnoise*), and then rejecting any points that are more than *lsigma* below that median or *hsigma* above that median. The process is iterated until no more values are rejected at a given pixel.
- **crreject** This is identical to *ccdclip*, except that low pixels are ignored and only high pixels are rejected.

**Sigma determined empirically:** Rather than determine the expected spread in our data values from the known characteristics of the CCD, we may want to make some attempt to determine the “expected” sigma based upon the data itself. Times that this might be useful would be if you have altered the data in some way, particularly by subtracting sky, say. The two primary schemes here are *sigclip* and *avsigclip*, whose effectiveness is determined by how many images you are attempting to combine.

- **sigclip** With this rejection algorithm, the median is first computed at each pixel value by first ignoring the low and high value. The sigma about this median is then determined using all the data values at this pixel. Next, the median is recomputed rejecting any data points that are *lsigma* or *hsigma* low or high. The sigma about this new median is computed as well as the new sigma, ignoring data values that were excluded in determining the median. The last two steps are repeated until no more data values are rejected. As you may guess, this works well if there are many ( $> 10$ ) images.
- **avsigclip** This is a variant on the *sigclip* algorithm, and works well in the case that there are only a few images. It is also probably the hardest algorithm to understand or describe. Rather than compute the sigma from the spread in the data values at a given point, however, the algorithm assumes that the “expected” sigma is equal to the square-root of the median multiplied by some constant. (This would lead to determining the gain for a CCD in the ideal case, if there were no read-noise and the data had been unaltered.) As an extra twist, it determines this constant independently for each row of the data.

## C Summary of Reduction Steps

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 spectroscopic data (C.1 – not included in these notes) and an example of reducing a night of direct imaging data (C.2). For either example we assume that the calibration exposures you have are:

- **biases (zerosecond exposures):** Used to remove pre-flash illumination or any residual structure in the DC offset not removed by the over-scan region.
- **flatfield exposures:** These are used to remove the pixel-to-pixel gain variations and possibly some of the lower-order, wavelength-dependent sensitivity variations. Depending upon the instrument, these flatfield exposures may or may not do an adequate job of matching the illumination function along the slit (i.e., in the spatial direction).

- **twilight exposures:** These are used to correct any mismatch between the flatfield exposure and the slit illumination function.

**C.1 Spectroscopic Example (Not included. See original document.)**

**C.2 Direct Imaging Example (Not included. See original document.)**