

Astronomy 337

Basic CCD Data Reduction Guide

Spring 2009

I. Introduction

A single "raw" (unprocessed, fresh-from-the-telescope) CCD digital image of the sky contains pixel values measured in counts, or analog-digital units, ADUs, with the number of electrons per ADU set by the gain in the CCD electronics, typically 2 or 3 e⁻ ADU⁻¹. These counts come from several possible sources :

- Real photoelectrons from astronomical sources and sky background, or "signal"
- Bias, a low-level electrical voltage added to each pixel during image readout
- Dark current, thermal electrons collected in each pixel during the exposure
- Noise, or fluctuations above and below the three sources listed above.

The raw images will need to be "reduced", or processed, (1) to distinguish among those sources of pixel counts and separate out the desired signal, and (2) to calibrate each pixel's measured value by its known sensitivity using a "flat field" image, or image of a smooth, uniform target such as twilight sky or the inside of the telescope dome.

II. Overview

The overall process for reducing and calibrating raw CCD data can be summarized as follows:

$$\text{final reduced image} = \frac{\text{raw image} - \text{master bias} - (\text{master dark} \times t_{\text{exp}} / t_{\text{dark}})}{\text{master flat}}$$

where

- master bias = median or mean of all bias frames
- master dark = median or mean of all (dark - master bias), ideally obtained for each t_{dark}
- master flat = median or mean of all (flat - master bias - [master dark × t_{flat} / t_{dark}]) for each filter

We'll use IDL to execute all these tasks. Note the questions to address as you go through the process!

III. CCD Data Reduction Procedure

1. Make a Master Bias Frame

A low-noise mean or median of a series of individual bias frames

```
files = dialog_pickfile(/multiple)
```

; Select all bias frame FITS files with the file selector window (Hint: use command-click on Macs!).

```
nfiles = n_elements(files)
```

```
fits_read, files[0], im, hdr
```

```
result = float(im) / nfiles
```

```
for i=1, nfiles-1 do begin & fits_read, files[i], im, hdr & result = result + float(im) / nfiles & endfor
```

```
sxaddpar, hdr, 'BZERO', 0.0
```

```
fits_write, 'my_bias.fits', result, hdr
```

Questions: What is the mean bias level in the bias frames? What is the range in that bias level? What is the rms fluctuation per pixel in a single bias frame? What is the true rms read noise? How did you measure that?

2. Make a Master Dark Frame

A low-noise mean or median of bias-subtracted dark frames divided by their exposure time.

```
files = dialog_pickfile(/multiple)
```

; Select all dark frame FITS files with the file selector window.

```
nfiles = n_elements(files)
```

```
fits_read, 'my_bias.fits', bias
```

```
result = bias * 0
```

```
for i=0, nfiles-1 do begin & $  
  fits_read, files[i], im, hdr & $  
  exptime = sxpar(hdr, 'EXPTIME') & $  
  result = result + (float(im) - bias) / exptime / nfiles & $  
endfor
```

```
sxaddpar, hdr, 'BZERO', 0.0
```

```
fits_write, 'my_dark_per_second.fits', result, hdr
```

Question: For each temp, what is the dark current rate in $e^- \text{pixel}^{-1} \text{sec}^{-1}$?

3. Make a Master Flat Field

A low-noise mean or median of bias- and dark-subtracted flat field frames, normalized by the result's mean or median.

```
files = dialog_pickfile(/multiple)
```

; Select all flat field FITS files **observed in a given filter** with the file selector window.

```
nfiles = n_elements(files)
```

```
fits_read, 'my_bias.fits', bias, biashdr
```

```

fits_read, 'my_dark_per_second.fits', dark, darkhdr

result = bias * 0

for i=0, nfiles-1 do begin & $
  fits_read, files[i], im, hdr & $
  exptime = sxpar(hdr, 'EXPTIME') & $
  result = result + ((float(im) - bias) - dark * exptime) / nfiles & $
endfor

result = result / median(result)

sxaddpar, hdr, 'BZERO', 0.0
fits_write, 'my_flat_field.fits', result, hdr

```

Question: What does the master flat frame look like?

4. Process the Data Frames

Subtract bias and scaled dark frames from each raw data frame, then divide by the appropriate normalized master flat field frame.

```

files = dialog_pickfile(/multiple)

; Select all science target FITS files observed with the same filter as your flat field.

nfiles = n_elements(files)

fits_read, 'my_bias.fits', bias, biashdr
fits_read, 'my_dark_per_second.fits', dark, darkhdr
fits_read, 'my_flat_field.fits', flat, flathdr

```

```

for i=0, nfiles-1 do begin & $
  fits_read, files[i], im, hdr & $
  exptime = sxpar(hdr, 'EXPTIME') & $
  im = ((float(im) - bias) - dark * exptime) / flat & $
  sxaddpar, hdr, 'BZERO', 0.0 & $
  fits_write, files[i]+'proc.fits', im, hdr & $
endfor

```

Questions: What do the reduced images look like? Are they flat? What is the background level? Did bias and dark get subtracted successfully, or are there big residuals left over due to some normalization error?

V. Optional: Make a 3-color image

If you have images of a target taken through three different filters, you can combine them to make a 3-

color image using Rob Gutermuth's custom GUI task available in the ast337 version of IDL:

threecolor

VI. Optional: Creating a Mosaic

Align and coadd your calibrated data frames together to make a mosaic.

If you have more than one image taken through a given filter, you will probably want to combine them into a single, less noisy (higher signal-to-noise) image. The images may be pointed at different centers with overlapping edges; the final image assembled from these individual "tiles" is called a "mosaic".

1. The hard way:

```
files = file_search( '*.proc.fits', count=nfiles )
```

; Note the automatic file selection used here! You can use `dialog_pickfile` instead if you like clicking!

```
fits_read, files[0], master, masterhdr  
mastermask = master * 0 + 1  
offsets = fltarr( 2, nfiles )  
master_offset = [0. , 0.]
```

```
for i=1, nfiles-1 do begin & $  
fits_read, files[i], im, hdr & $  
immask = im * 0 + 1 & $  
tmpoffset = rob_reg( master, im ) & $  
offsets[* ,i] = tmpoffsets[0:1] - master_offset & $  
txoff = tmpoffset[0] & tyoff = tmpoffset[1] & $  
irac_spooffset, im, hdr, immask, kx, ky, txoff, tyoff & $  
im = poly_2d(temporary(im)*immask,kx,ky,1,missing=0) & $  
immask = poly_2d(temporary(immask),kx,ky,1,missing=0) & $  
wbd = where(mask lt .1,nbd) & $  
wgd = where(mask ge .1,ngd2) & $  
if nbd gt 0 then begin & $  
im[wbd] = 0. & $  
immask[wbd] = 0. & $  
endif & $  
if ngd2 gt 0 then im[wgd] = im[wgd] / immask[wgd] & $  
mosaic_two, master, mastermask, im, immask, txoff, tyoff, xmoft, ymoft, /autooffset & $  
master_offset = master_offset+[xmoft,yoft] & $  
endfor
```

```
fits_write, 'my_mosaic.fits', master, masterheader
```

2. The easy way:

```
mosaic = rob_assemble2( ) ; that's right, no command line inputs necessary!
```